Použití mobilních zařízení pro vzdělávání

Application of mobile devices in education

Bakalářská práce

Josef Hošna

Vedoucí bakalářské práce: RNDr. Jaroslav Icha

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

2008

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval/-a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne

Anotace

Cílem mé práce je zpracovat problematiku programovani výukových Java aplikací pro mobilní telefony. Toto programovaní bude odlišné od standartní Javy pro specificke pozadavky aplikací kvůli kompatibilite a rozdili mezi jednotlivými zařízeními. Práce obsahuje (bude obsahovat) ukázkovou aplikaci, s podrobným průvodcem jak tuto aplikaci naprogramovat krok po kroku.

Abstract

The goal of my work is to comprehend the questions of educational Java applications for cellphones. Such programming will be different from standard Java programming due to specific application requirements, compatibility issues and differences in various mobile devices. The work contains a demonstration application with detailed instructions on step-by-step programming.

Poděkování

Rád bych poděkoval svému vedoucímu RNDr. Jaroslavu Ichovi za cenné rady pro tvorbu bakalářské práce.

Dále bych rád poděkoval své rodině, přítelkyni a kamarádům za podporu ve studiu a trpělivost při tvorbě této práce.

Obsah

1 Úvod

V úvodu bylo čerpáno z [1]-[5].

Mobilní telefon, když se řekne tento pojem, každý si vybavý tu malou(nebo i větší) věcičku, kterou má někde v kapse, batohu či kabelce a kterou střeží jak oko v hlavě. Takto bedlivě si svůj telefon nestřežíme, protože tzv. mobil je nemalá investice pro každého, ale hlavně protože si v dnešní době lidé život bez mobilních telefonů už ani nedokáží představit. Mobilní telefon je každodenním pomocníkem každého z nás, pomáhá nám komunikovat s přáteli, rodinou, domlouvat si obchodní schůzky nebo rande či v krajním ale ne neobvyklém případě je přivolání pomoci v nouzi. Já osobně používám telefon několikrát denně k telefonování, psaní SMSek , jako budík nebo pro organizaci svého volného času pomocí zabudovaného kalendáře. I přes tuto dnešní samozřejmost to není tak dávno co se nám o mobilních telefonech sotva zdálo.

Jak to vlastně bylo s jejich začátky? Prvním mobilním telefonem na světě byla Motorola DynaTAC 8000X(přezdívaná cihla) uveřejněná roku 1983 s cenou 3995 dolarů, hmotností kolem 800g a rozměry 33 × 4,3 × 8,9 cm. Baterie tomuto stroji vydržela na jedno nabití pouze 30 minut hovoru, popř. 8 hodin pohotovostního režimu. Její funkční prototyp byl hotov už v roce 1972, muselo se však čekat na vybudování sítě po které se bude volat, proto tento časový rozestup. Motorola DynaTAC 8000X byla telefonem 1. generace, která byly založena na analogovém

přenosu hovorového signálu, jenž tvořily několik odlišných sítí. V té době nejrozšířenějším evropským standardem sítě byl NMT "Nordic Mobile Telephone".

V roce 1991 přichází s touto sítí do české republiky Eurotel, což se dá považovat za počátek mobilní komunikace u nás. Ceny telefonů nebyli zatím únosné pro normální lidi, jelikož se mohly vyšplhat i přes 30 000 Kč. Poplatky za volání také byli také velmi vysoké a neplatili se jen odchozí hovory ale i příchozí. Když už jste tyto peníze měli a mohli je obětovat dostali jste přístroj, ze kterého se dalo pouze volat. Není tedy divu, že se mobily vídali jen u manažerů a podnikatelů. Tyto přístroje měli pevně dané číslo, takže SIMku by jste nenašli. Ty se začali objevovat až roku 1996 ale ne dnešního typu.

Mezníkem, po kterém se začínají mobilní telefony u nás dostávat i mezi širší veřejnost, je příchod dalšího operátora, neboli vstup firmy Radiomobil se sítí Peagas na český trh. Oba operátoři spouští sít GSM, která zde drží dodnes. Tímto příchodem zapracovala na trhu zdravá konkurence a ceny přístrojů společně s poplatky začaly klesat, takže se stali dostupnými pro většinu lidí a nastal mobilní boom.

Další kroky v historii mobilní komunikace jako byl komerční start sítě Český Mobil(síť Oskar) 1. března 2000 už pro nás nejsou nejsou zajímavé. Nejdůležitějším pro nás bude ten fakt, že kolem roku 2001 se začali na trhu objevovat telefony s podporou platformy J2ME. Pro lidi, kteří rádi programují se tak otevřela možnost programovat utilitky a vychytávky pro své miláčky, které mají pořád u sebe. To znamená ze si z mobilu mohli udělat kalendář, kalkulačku či cokoliv jiného, proto se od té doby jakoby

roztrhl pytel s nejrůznějšími programy pod tímto prostředím. Šířili se programy pro usnadnění organizace času, kalkulačky atd., ale nejvíce se rozšiřovaly různé hry.

Postupem času se mobilní telefony zdokonalili na takovou úroveň, že na nich mohou běžet aplikace, na které jsme před deseti lety potřebovali celý počítač. To znamená že dnes u sebe většina lidí nosí multimediální přístroj s relativně velkou vnitřní pamětí a nemalým výpočetním výkonem. Máme ho u sebe všude např. ve frontě na úřadě, u kadeřníka, při volné hodině ve škole, ve vlaku nebo v čekárně u lékaře před preventivní prohlídkou. Co mají tyto situace společného? Společný rys těchto situací je nuda, neboť všude kde se na něco čeká je čas nevyužitý či přímo ztracený, pokud se člověk nudí. Kolikrát se vám stalo, že jste podobně čekali den před zkouškou a nemohli jste se učit, jelikož jste neměli učebnici. Nebo jste v podobné situaci zapomínali cizí jazyk místo jeho procvičování, ale neměli jste slovník. Co jste ale měli? Mobilní telefon! Proč ho tedy nevyužít k tomu co momentálně potřebujeme, využít ho ke vzdělávání, které je v dnešní době nezbytnou součástí života každého z nás. V takovýchto situacích se lidé vyskytují čím dál častěji, proto vznikl m-learning, neboli učení z mobilu.

Spousta lidí pro alespoň nějaké využití těchto a jiných volných chvil hrají na svém mobilu hry. Mě osobně takto “využitý” čas přijde spíš přímo zabitý, protože hry moc nehraji a když už, tak na počítači, kde je velká obrazovka, výkonná grafika, kvalitní reproduktory, myš atd.. Z těchto důvodů mě myšlenka vzdělávání z mobilu zaujala v takové míře, že jsem si ji vybral za téma své bakalářské práce.

Když jsem začal hledat , zjistil sem ze ve vodách českého internetu je to oblast, zatím pomalu neprozkoumaná o knižních publikacích ani nemluvě. Jedny z mála dokumentů o této problematice publikuje *Ing. Pavel ROSMAN, Ph.D.* Je sice pravda že že je to pouze podnož e-Learningu, který je omílán na každém rohu, ale žádný dokument není zaměřen takto úzce.

A kolik se toho v této práci o něm dozvíte? V první části si rozebereme vše o tom co to m-learning je, jaké má výhody a nevýhody. Jaké jsou důvody jeho vzniku, způsoby praktikování. Podíváme se na možnosti uplatnění a prostředky se kterými se dá použít. Zkráceně řečeno si podrobně probereme vše o m-Learningu.

To by ale samo o sobě nebylo lidem, kteří si čtou tuto práci a chtěli by se m-learningem zabývat, moc platné, tak si v další části dáme přídavek v podobě seznámení se s nejvíce podporovaným programovacím jazykem mezi mobilními telefony a to s jazykem “Java” a jeho odnoží pro mobilní zařízení J2ME a hlavně jejich rozdíly, protože J2ME je velmi odlišná od klasické Javy v níž hodně lidí programovat umí .

Poté si ukážeme v dnešní době asi nejpopulárnější vývojové prostředí pro tuto platformu neboli NetBeans v nové verzi 6.0. Ukážeme si jeho použití při programování aplikací pro mobilní zařízení s platformou J2SE. Vezmeme to od začátku, neboli jeho stáhnutí nainstalování a použití. Popíšeme si jeho nástroje, komponenty a různé vychytávky.

Nakonec všechny vaše čerstvě nabrané poznatky využijeme ve čtvrté a poslední části mé práce, kde si spolu krok za krokem napíšeme jednu výukovou aplikaci. V poslední části se nebudeme

zabývat pouze psaním aplikace, ale budeme si říkat jak správně program napsat od nápadu co by program měl dělat přes jeho návrh až po správnou realizaci.

2 M-Learning

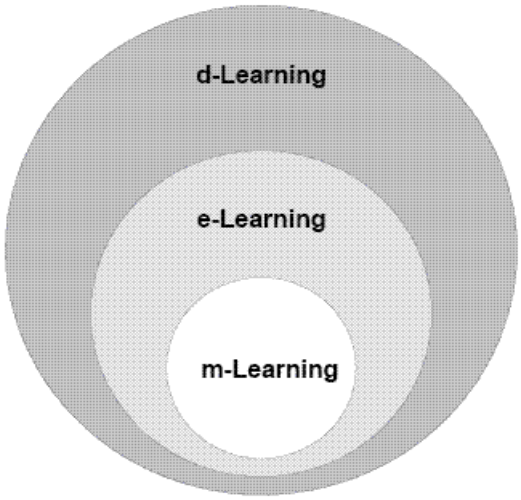
Tato kapitola byla psána za pomoci [4] až [12].

2.1 Úvod do m-Learningu

E-learning, což je moderní vzdělávání pomocí ICT, je dnes velmi známou a běžnou formou vzdělávání. Vyskytují se však situace, kdy je nutné tuto metodu více přizpůsobit novým směrům. Můžeme přitom využívat volné chvíle, kdy mnoho z nás má po ruce mobil, nebo i PDA. Jedním z řešení této situace mohou být interaktivní multimediální kurzy využívající mobilní ICT ve vzdělávacím procesu. Pro uchopení možností, které mobilní ICT skýtají se do popředí dostává pojem “m-Learning” (mobilní e-Learning). M-learning tedy využívá v procesu vzdělávání mobilní technologie, jako jsou kapesní počítače (PDA), mobilní telefony (MDA), notebooky či tablet PC. Dá se chápat jako nová, doplňková forma vzdělávání založená na aktivním samostudiu a individuální práci studujících, jejíž hlavním kouzlem je snadná dostupnost nejen doma, ale i v práci a na cestách.

2.1.1 Pozice m-Learningu

Ve zkratce se dá říct, že m-Learning je podmnožinou e-Learnigu, ve které se k vyučování používá z ICT hlavně mobilní zařízení, kde e-Learning je podmnožinou distančního nikoliv prezenčního vyučování . Tato situace je zobrazena na následujícím obrázku č. 1.



Obrázek 1 : Struktura distančního vzdělání

V předchozím textu bylo použito několik nových výrazů, proto si je zde vysvětlíme:

Prezenční studium je forma studia, při které se vyžaduje fyzická účast studentů při výuce ve třídě. Po dobu výuky jsou učitel a studující v přímém kontaktu (při studiu doma to neplatí).

Distanční vzdělávání (d-Learning) je vzdělávací proces, který umožňuje oddělit vyučujícího a studujícího v čase nebo místě, popřípadě v obojím. Komunikace mezi učitelem a studujícím probíhá v moderních distančních kurzech hlavně elektronicky (pomocí e-mailu, chatu, audia, videa, telekonference, nejčastěji v prostředí webu). Výukové materiály, jako knihy, audio a video

kazety, CD-ROM může být někdy účelné zasílat poštou, nikoliv po síti – poté mluvíme o korespondenčním (klasickém) d-Learningu.

Blended learning - v praktické výuce se nevyskytují obvykle "čisté" formy, například čistá prezenční, elektronická nebo distanční forma. V praxi se prosazuje koncepce smíšeného vzdělávání (blended learning), což je kombinace prvků prezenčního i e-Learningového(m-Learningového) vzdělávání, která má největší perspektivu použití na středních a především vysokých školách.

ICT je zkratka oboru informačních a komunikačních technologií z anglického názvu Information and Communication Technologies. ICT vzniklo z IT, když mezi sebou začaly počítače a celé počítačové sítě komunikovat ve velkém. Vrcholem této komunikace se staly mobilní telefony. ICT nejsou jen hardware, ale i software, který strojům říká, jak pracovat či zpracovávat informace podle potřeb uživatelů. Ono C znamená komunikaci mezi počítači a počítačovými sítěmi. ICT je tedy i o přenosu informací, kterému daly nový rozměr internet a mobilní sítě, po nichž neustále proudí neuvěřitelné množství dat. Komunikaci zprostředkovávají i telekomunikační sítě a satelity.

E-Learning lze velmi těžko definovat, protože na něj existuje více pohledů (pedagogický, technologický, síťový) a dodnes není ujednocená definice. E-learning v širším slova smyslu je definován zejména jako aplikace nových multimediálních technologií

a Internetu do vzdělávání za účelem zvýšení jeho kvality posílením přístupu ke zdrojům, službám, k výměně informací a ke spolupráci. Tato definice chápe e-learning jako jakékoli využívání informačních technologií multimediálního charakteru ke zlepšení kvality a efektivity vzdělávání. To znamená, že za e-learning můžeme v širším slova smyslu považovat například používání výukových CD-ROMů v rámci prezenční školní výuky. E-learning můžeme chápat jako multimediální podporu vzdělávacího procesu za použití moderních informačních a komunikačních technologií (ICT), jejichž primárním úkolem je zvýšit kvalitu a dostupnost vzdělávání. V užším slova smyslu je e-learning chápán zejména jako vzdělávání, které je podporované moderními technologiemi a které je realizováno prostřednictvím počítačových sítí – intranetu a zejména Internetu. Tato definice již popisuje e-learning, jak jej zná většina populace – jako vzdělávání po Internetu.

2.1.2 Dostupná zařízení

Mobilní učení se samozřejmě provozuje skrze mobilní zařízení. Na výběr máme z mnoha druhů od přenosných počítačů (notebooků, tablet PCs) přes kapesní počítače, smartphony, komunikátory a obyčejné mobily až po různá přenosná zařízení jako Ipod, mp3 přehrávače atd.. Jaké vychytávky mají které zařízení?

Notebook a tablet PC mají stejné parametry jako stolní PC(od PC jediná odchylka, standardní příslušenství je WiFi) s tím rozdílem, že tablet PC má dotykový display. Obojí jsou přenosná

zařízení s omezenou mobilitou. Při notebook s mobilním internetem (pomocí GSM technogie), se hranice e-Learningu a m-Learningu překrývají, proto se budeme v dalších částech věnovat spíše ostatním technologiím. Právě když skloubíme notebook a mobilní internet, můžeme ke studiu používat klasický e-learningový online kurz a už jsme opustili oblast m-learningu.

Další kategorií jsou kapesní počítače. Ty mají malé rozměry (vysoká mobilita), dotykové displaye, OS a aplikace v ROM a RAM, komunikační prostředky(bluetooth, IrDA, Wi-Fi), rozšiřující porty a možnost synchronizace s PC. Mají vlastní procesor i vnitřní paměť rozšiřitelnou pomocí paměťových karet.

Co se týče smartphonů a komunikátorů, ty mají povětšinou stejné parametry jako kapesní počítače, akorát nemívají tak veliký display, ovšem mají navíc možnost komunikovat přes GSM.

K normálním mobilům je těžko něco říkat, jelikož je více kategorií, které jsou i jinak cenově dostupné. Podle toho kolik člověk investuje může mít telefon blížící se parametry k smartphonu nebo “hloupoučký” telefon s černobílým displayem.

Ostatní mobilní přístroje mohou sloužit ve výuce spíše jenom k poslechu mluveného slova (výuka cizí řeči). Ale pokud toto zařízení podporuje videosekvence, může se z něj stát interaktivní nástroj pro názorné vysvětlení technik různých dovedností pomocí výukového videa.

2.1.3 Využití v praxi

Předem si musíme říct, že mobilním vzděláváním nechceme nahradit výuku ve třídách, ale podobně tak jako u e-Learningu dostáváme další způsob, jak tuto výuku doplnit. Tento způsob je technologicky velmi přístupný, protože mobilní přístroje jsou v dnešní době snadno pořiditelné pro širokou vrstvu lidí, jsou také zábavné a mohou tak přivést ke vzdělávání mnohem širší okruh zájemců všech věkových kategorií. Jelikož hlavně mezi mladými lidmi jsou velmi oblíbené dražší typy mobilních telefonů, mohl by takový doplněk vyučování vyvolat zálibu ve studiu i u nich. Také může pomoci ke vzdělání lidem, kteří neuspěli v klasickém výukovém systému. Nebo usnadní přístup ke vzdělání lidem nezaměstnaným (nezaměstnatelným), pro zvýšení jejich dovedností a zpřístupnění více pracovních míst.

M-Learning se snaží začlenit elektronické vzdělávání využívající tyto moderní informační a komunikační technologie jako trvalou součást každodenního života všech kdo se potřebují vzdělávat. Umožňuje tak uživatelům rychle se zorientovat v nových informacích a naučit se novým dovednostem, aby obstáli v rychle se měnícím ekonomickém i společenském prostředí.

U mobilních zařízení se neustále zvyšuje jejich výkonnost a postupně se zlepšují jednotlivé funkce, mezi nimi také video, barevná dotyková obrazovka, webový prohlížeč nebo kompatibilita s aplikacemi klasického PC, což z mobilního vzdělávání dělá věc nejen možnou, ale hlavně prakticky využitelnou. Pro tato vylepšení velké množství lidí již dnes mobilní přístupové technologie používá např. pro přístup k e-mailu, vyhledávání na internetu, organizování

svého času (kalendáře), nebo pro přístup k pracovním dokumentům.

Moderní ICT jsou schopné splnit poslání vzdělávání pro masy lidí, zmírnit krizi z nárůstu nákladů na vzdělávání a zvládnutí explodujícího nárůstu studentů toužících po vzdělání. Proto se musí vzdělávací programy postupně přiblížit k novým komunikačním technologiím a využít jejich plný potenciál.

V předchozím odstavci je velmi důležité slovíčko “toužících”, protože my můžeme udělat cokoli proto, aby jsme vzdělání co nejvíce přiblížili lidem, ale pokud potenciální student po vzdělání netouží jen stěží ho donutíme použít mobil ke vzdělávání. Pravdou sice je, že velké kouzlo m-Learningu je v jeho mobilitě, ale prostředí ve kterém máme k dispozici pouze mobil nemusí být pro studium vždy přející. V nevýhodu mobilních zařízení se v tomto případě obrací i velikost těchto zařízení, je to sice důvod proč jsou mobilní, ale bohužel jejich malé displaye, skromné klávesnice a u většiny zařízení i nevelká vnitřní paměť z nich nedělají zrovna nejpohodlnější nástroj pro studium. Z čehož vyplývají důvody, proč je m-Learning pouze doplněk ke studiu.

U nás v republice se zúčastnili dvě školy z Plzně projektu mezinárodní spolupráce TransmobiLE v rámci programu Evropské unie Leonardo II, kde se účastnili německého projektu, ve kterém pro výuku používali mobilních zařízení. Tato první etapa podle stránek škol dopadla velmi úspěšně. Naše univerzita začíná pracovat ve spolupráci s německou univerzitou na projektu studia jazyků z mobilů, bohužel tento projekt je zatím v počátcích. Na druhou stranu zde v Českých Budějovicích působí jazyková škola the Microschool, jejíž majitel mi sdělil, že před pár lety nabízeli

svým žákům jako doplněk k výuce mobilní aplikace na učení nepravidelných anglických sloves, to se neuchytilo.

Rozdíl mezi těmito dvěma pokusy byl v tom, že na plzeňské škole žáci ke studiu používali PDA zařízení, ale v the Microschool byl program určen pro mobilní telefon, který není tak pohodlný.

Můj názor je, že m-Learning je šikovná pomůcka k výuce, kterou já osobně rád používám (onen prográmek na procvičování sloves), ale je to moje subjektivní rozhodnutí. Jelikož člověk je od přírody tvor líný a pohodlný, tak pokud mu tento způsob nepřijde jako pohodlný způsob výuky používat ho nebude.

Zda má m-Learningu v dnešní době potenciál pro velké rozšíření, záleží na mnoho aspektech. Jedním z nejdůležitějších je vývoj mobilních zařízení, které by měli uživatelům v budoucnu nabízet standardně funkce, které jsou dnes nadstandardní a finančně těžko dostupné. Pokud bude v budoucnu každý mobilní telefon nabízet podobné funkce jako dnes např. Iphone, tak bude mobilní studium připadat pohodlné a zábavné každému.

2.2 M-kurz

Metod m-Learningu neboli mobilního učení je hodně. Jde o to, co chceme učit. Pokud se jedná např. o výuku cizích jazyků lze za mobilní učení považovat i poslech mluveného slova na cestách. Obecně se v mobilním učení narozdíl od e-Learningu (kde je nejrozšířenější online kurz) používá offline vzdělávání, to znamená, že si stáhnete výukovou aplikaci k sobě do mobilního zařízení a ke vzdělávání nepotřebujete být připojeni ke zdroji dat,

připojíte se pouze pro aktualizaci či doplnění výukových dat pro aplikaci. Existují samozřejmě i jiné metody, na WAPu může být obdoba klasického online kurzu (tato metoda je ovšem s dnešními poplatky docela nákladná pro používání), nebo se používá SMSkvíz.

Základem klasického vyučování je studijní kurz, který obsahuje bloky předmětů, uspořádaných tak, aby se dosáhlo požadovaných cílů vzdělávání. V e-Learningu se tento kurz používá také, ale má dvojí využití. Zprvu se lidé snažily tímto kurzem nahradit výuku a vytvořit interaktivní učebnici pro samouky, kde by si člověk učivo přečetl, mohl by si v opakovacích otázkách ověřit co se naučil a popřípadě se vrátit k látce kterou moc neovládá. A v tomto části je zakopán kámen úrazu, jelikož nikdy nevíte zda se žák vrátí k látce, nebo půjde dál, jelikož nad ním nestojí učitel, který by žáka kontroloval a nutil ho se učit. Z těchto důvodů se tato metoda neuchytila jako náhražka klasického vyučování a e-Learning se používá pouze jako doplněk klasické výuky, což znamená, že obsahuje celý studijní kurz, ale žák jej používá pouze pro doplnění či zopakování učiva.

V m-Learningu je to stejné, používá se pouze k doplnění, zopakování nebo procvičení učiva. Lze si tedy vybrat jak obsáhlá tato aplikace pro podporu výuky bude. Můžeme vytvořit celý studijní kurz (v našem případě m-Kurz), který bude obdobou interaktivní učebnice a bude studenta provázet celou výukou, nebo vytvoříme aplikaci omezenější zaměřenou pouze na procvičování určité části učiva jako je například program pro procvičování nepravidelných sloves v německém jazyce. My si zde probereme postupně jak vytvořit kompletní studijní kurz, protože pro konkrétní aplikaci si pak z něj můžete vybrat pro konkrétní případ

potřeby doplnění výuky pouze část, která bude vyhovovat vašim požadavkům. V některých případech budete potřebovat podporovat výuku pouze výkladovou částí bez testů či procvičování a někdy zase budete potřebovat výuku podpořit aplikací, kde si studenti procvičí nově nabrané znalosti. Kompletní studijní kurz charakterizují zejména tyto složky:

1. vzdělávací obsah studijních materiálů
2. elektronická distribuce vzdělávacího obsahu
3. elektronická správa m-kurzů.

Vzdělávací obsah studijních materiálů – obsah učiva se musí didakticky transformovat do formy interaktivního softwaru specificky určeného pro samostudium. Software by se proto měl skládat z těchto částí -> programované vstupní informace, učební úlohy ,zpětnovazební kontrolní informace, nezbytné řídící instrukce.

Distribuce vzdělávacího obsahu – Jelikož m-Learning je mobilní e-Learning, k distribuci se používají pouze mobilní zařízení jako kapesní počítače (PDA), mobilní telefony (MDA). Pro přenos dat se používá sítí GPRS, GSM či Wi-Fi internet, možné je i použití Bluetooth nebo IrDA, ale to spíše pro výměnu dat mezi PC a mobilním zařízením.

Elektronická správa m-kurzů – Pokud je kurz realizován formou WAPového kurzu nebo aplikací komunikující se serverem, kde jsou uloženy výukové materiály nebo se vyhodnocují studijní výsledky , zabezpečují správu SW systémy pro řízení studia (Learning Management Systems [LMS], někdy též Course Management System), jako u klasického e-Learningového online kurzu. Pokud

je ale kurz vytvořen jako program který si člověk pouze stáhne do mobilu, těžko bychom ho zpravovaly na serveru přímo v systému. V tomto případě je m-Learning ochuzen o možné přehledy a statistiky výsledků studentů či přímé změny v systému a není tedy možné tak rychle reagovat na změny v odborném obsahu, pouze tak že se stáhne aktualizovaná verze programu nebo dat.

2.2.1 Realizátoři

Pro tvorbu m-kurzu je potřeba tým lidí, jelikož je třeba se zaměřit na hodně oblastí. Základní potřebné profese jsou tři a to manažer, vývojář a tutor. Jelikož se pohybujeme v oblasti mobilních zařízení a m-Learningu, měli bychom si tyto profese obohatit o předponu “m-” neboť i pro profese spojené s e-Learningem se používají označení e-tutor atd.... Vznikl nám tedy tým skládající se z profesí m-manažera, m-vývojáře a m-tutora.

M-manažer – má za úkol návrh a koordinaci projektu, analyzovat studijní skupinu, navrhnout výukové cíle, celkovou strategii, analýzu a ocenění výuky, zabezpečení servisu, marketing.

M-vývojář – má za úkol vytvořit projekt výuky, didaktickou transformaci obsahu, posouzení obsahu, tvorbu multimedií, tvorbu grafiky, programátorské práce, pilotní ověření m-kurzu.

M-tutor – náplní jeho práce jsou aplikace LMS v m-kurzu (nesmíme zapomenout, že ne vždycky budou LMS součástí kurzu),

vedení výuky, rady ke studiu, konzultace, podpora a usnadnění studia, administrace výuky.

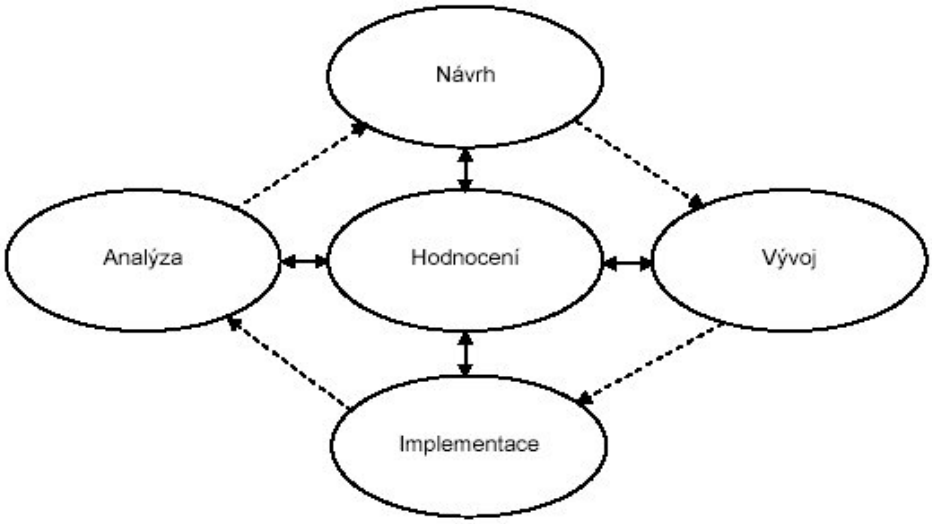
Pro každý jednotlivý úkol u těchto profesí jsou zvláštní odborníci, pokud chceme tým minimalizovat, více úkolů může zastat jeden člověk, ale profesi m-vývojáře zastávají většinou tři odborníci a to projektanta výuky (expert na pedagogiku), autora obsahu (expert na odborný obsah) a počítačového odborníka(programátor, grafik, designer).

Zde je vidět, že obsah je většinou připravován odborníky v oboru, je tedy zaručena kvalita obsahu i didaktického zpracování. To je obrovská výhoda oproti klasické výuce, kde je obsah závislá na osnovách a didaktická úroveň přímo závislá na učiteli. Někdy se bohužel stává, že výuka je zaměňována s prezentací. Výhodu m-learningu lze nalézt i v názornosti, jelikož multimediální přístroj je ideální pro názornou výuku (záleží na úrovni mobilního zařízení, do jaké míry je multimediální). Nevýhoda takovéto přípravy kurzu je nákladnost, protože náklady jsou o hodně vyšší než při přípravě klasické výuky ve škole. Tato počáteční investice ale uspoří další výlohy opakovaným užíváním, sníží také náklady na cestování za výukou a organizace školení.

2.2.2 Didaktika m-Kurzu

Z předchozích informací vyplývá, že nemůžeme nejdřív začít psát a poté řešit, jak se to bude učit. Základem práce je vytvořit program, který bude odpovídat požadavkům výuky a bude provádět akce podle didaktického návrhu. Psaní programu tedy musí předcházet didaktický návrh projektu výuky. Toto není práce

programátora(ten je ve své podstatě pouze kodér) ale tutora. Způsobů didaktického návrhu označované Instructional Design (ID) je více, jedním z nejobecnějších je postup ADDIE (Analysis – Design – Development – Implementatiton – Evaluation), který zobrazuje obrázek 2.



Obrázek 2 : struktura postupu ADDIE

Tento postup se skládá z následujících fází včetně hodnocení u kterého z obrázku plyne centrální postavení v celém procesu:

Analýza (ANALYSIS). Je částí, kde identifikujeme hlavní problém. Zde dochází ke spolupráci návrháře s odborným expertem. Seznamujeme se s problémem, provádíme analýzu potřeb, vybíráme učivo, jež budeme školit a stanovujeme měřítka výkonu či provádíme odhad nákladů. Fáze analýzy je základním stavebním kamenem vývoje výukového programu.

Návrh (DESIGN). Specifikuje cíle programu, jako trojici {činnost, měřítko výkonu a podmínky výkonu}. Cíle, obvykle hierarchie

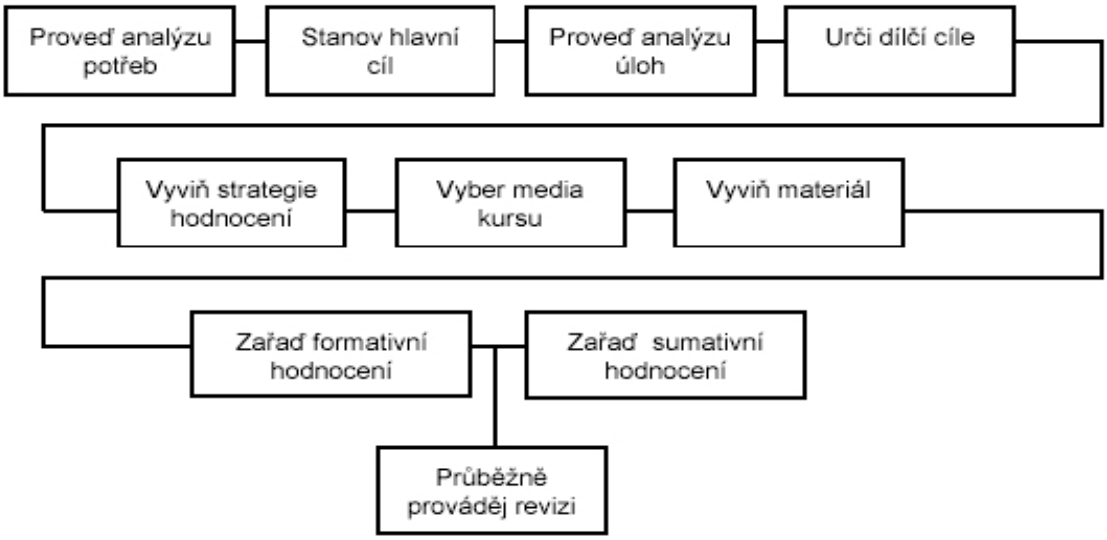
hlavního cíle a cílů dílčích, representují požadovaný výstup učení. K jeho dosažení projektujeme v této fázi jednotlivé kroky učení. Podstatnou se zde jeví zvolená teorie učení – chceme přece, aby zkonstruovaný program učil. Ta nás vede ke specifikaci strategií výuky. V této fázi navrhujeme i strukturu kursu. Výstupem fáze návrhu je tzv. model kursu, což je soustava diagramů hierarchických a vývojových a tabulek.

Vývoj (DEVELOPMENT). Vývoj zpracovává výsledku z fáze návrhu, tedy jednotlivé aktivity, vedoucí k naučení. Víme, že učení znamená změnu chování. Učení, to je akvizice nové informace, manipulace s ní a vyhodnocení, zda došlo k zapamatování. Vždy podle příslušných oblastí učení a vstříc jednotlivým stylům učení. A to nám dnešní ICT s multimédii umožňují.

Implementace (IMPLEMENTATION). Výstupem je plán řízení kursu se seznamem courseware, stanovením přípravy m-lektorů, popisem administrování kursu. Současně specifikujeme výstupní balík dle standardů, spustitelnost v prostředí LMS (Learning Management System).

Hodnocení (EVALUATION). Hodnocení prochází celým procesem tvorby kursu. Hodnocení určuje hodnotu programu, stanovuje kritická místa, odhaluje eventuální chyby. To vše s cílem nápravy. Zde rovněž prokazujeme, že program skutečně učí, a že učí v souladu s očekávanými výstupy. Opět existuje celá řada přístupů, metodik a technik, které nám napomohou k završení našeho úsilí.

Fáze z postupu ADDIE se objevují ve většině ostatních ID což si můžeme ukázat na jednom z nejpoužívanějších a to tzv. systémové navrhování, Instructional System Design (ISD).



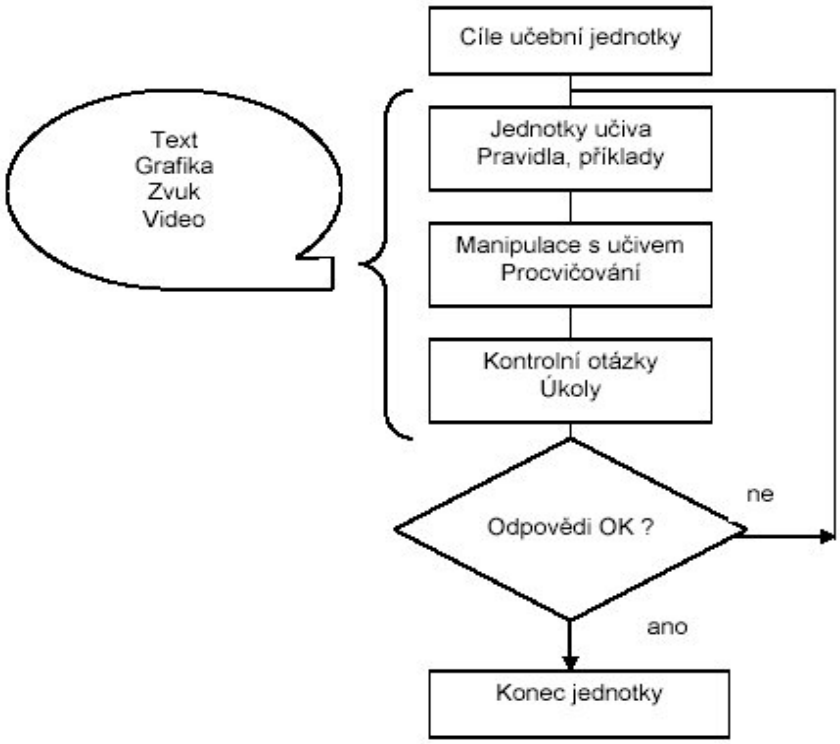
Obrázek 3 : systémové navrhování Instructional System Design

2.2.3 Zásady vývojáře

My se v další části práce budeme zabývat jak napsat vyučovací aplikaci, čili prací vývojáře. Proto si nyní trochu podrobněji probereme jeho úkoly a zásady práce.

První zásadou je konstruovat aplikaci tak, aby učivo bylo rozděleno do kratších etap, nesmí se tedy vše nahrnout na studenta najednou, jelikož nikdo nemá rád sáhodlouhé texty. V mobilním telefonu toto platí obzvláště, jelikož popojíždět tisíckrát na malém displeji mobilu by nebavilo nikoho, proto do kurzu zařadíme pouze důležité informace. Každá tato kapitola by měla být ukončena testem, aby si studenti mohli ověřit co si pamatují, popřípadě si uvědomit, že by si měli něco zopakovat

(zpětná vazba ve výuce). Schéma jednoho úseku (lekce) je znázorněn na obrázku 4.



Obrázek 4 : struktura jedné lekce.

Než začneme programovat musíme si nejdříve analyzovat co budeme tvořit, což znamená vymezit cíle výuky, pro jakou skupinu lidí budeme výuku připravovat, zjistit si zdroje a uvědomit si možné překážky. Musíme si také shromáždit, připravit, obsah výuky, který vytvoří odborník, nebo můžeme digitalizovat materiály stávající a upravit je do potřebné podoby.

Poté můžeme začít navrhovat, což znamená, že si uspořádáme obsah výuky a vytvoříme plán programu, vývojové diagramy podle kterých vytvoříme první verzi aplikace. Tu posléze

otestujeme studenty, vyladíme chyby a upravíme do šiřitelné podoby. Tím naše práce nekončí, musíme zařídit technickou podporu (webové stránky projektu) a soustředit se, zda uživatelé neobjeví nějaký skrytý problém a nezapomenout na aktualizace informací. Práci vývojáře si podrobně probereme v poslední části práce při samotném vývoji aplikace.

2.2.4 Prostředky

K vlastní výuce máme k dispozici pouze text, grafiku, obrázky, zvuky a videa. Jiné komunikační medium mezi žákem a naším kurzem k dispozici nemáme. Nyní se na jednotlivé součásti podíváme podrobněji.

Text - měl by být rozdělen do krátkých částí obsahujících nejvíce 6 nových pojmů. Tyto části je potřeba pro snadnější porozumění skládat z krátkých jednoduchých vět. Také se musí rozumně používat nadpisy, je zřejmé, že na malém displeji mobilu stačí jedna či dvě úrovně, na displayi PDA je více prostoru, ale počet úrovní by nikdy neměl přesáhnout 4. Důležité je i používání seznamů, ale opět nejvíce se dvěma úrovněmi. Pokud bude aplikace konstruována pro velmi malý display, mělo by se použít pouze jedné. Co se týče písma, pro elektronické materiály se obecně doporučuje písmo bezpatkové. V textu by měla být zvýrazněna všechna klíčová slova například tučným písmem, pokud text obsahuje i nějaké odkazy na jinou část materiálu měly by se také zvýraznit odlišností písma. Velmi důležité je také rozmístění obsahu v textu, nejdůležitější části se umísťují zásadně na začátek textu (první polovina), kde se lidé nejvíce soustředí.

Obrázky – jsou názorným doplňkem výuky, bohužel se musí brát ohledy na rozlišení displeje a počet podporovaných barev. Obrázek má mimo obsahu také určitou velikost (která hlavně u mobilních zařízení hraje důležitou roli), proto se nejčastěji používají komprimované formáty (GIF, JPEG).

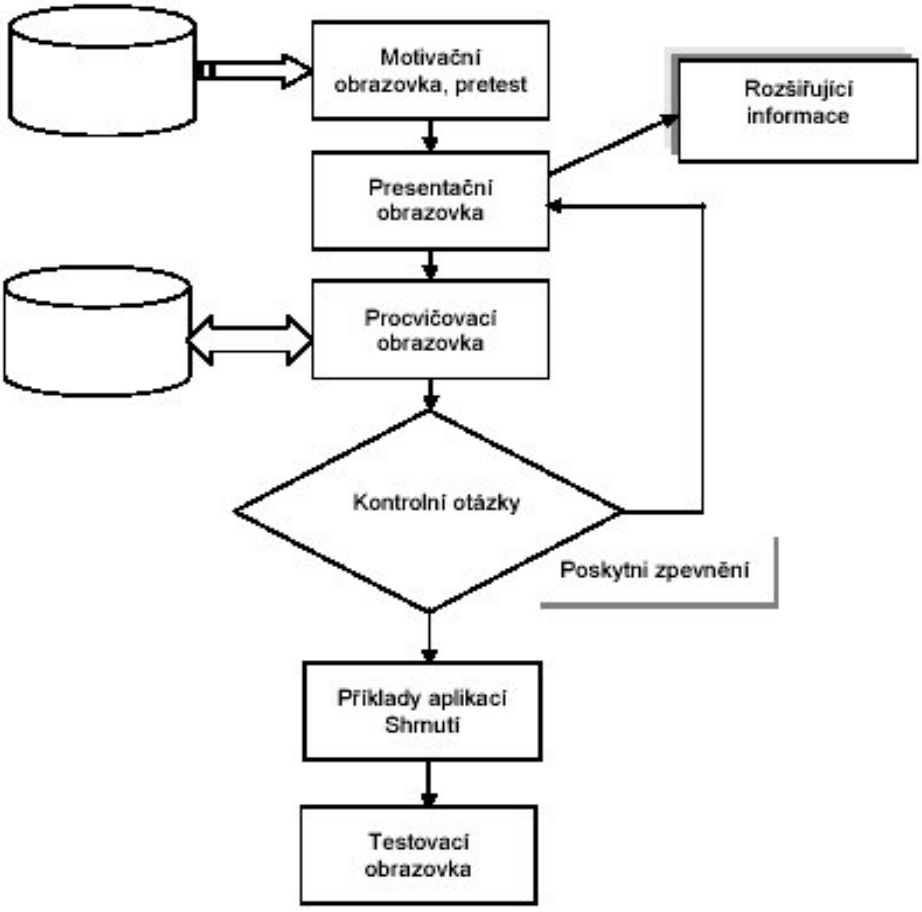
Zvuk –nezajímá nás jako hudba, i když je tu možnost pustit si ho na pozadí. Tato funkce by měla být uživatelem volitelná. Pokud je zvuk takto použit, je to atraktivní funkce programu. Nás spíš bude zajímat zvuk jako ekvivalent k textové informaci, což znamená že nám zvuk přehraje (přečte) informaci textovou. Pokud je zvuk použit takto, je to velmi efektivní pomůcka pro výuku, v některých případech přímo nevyhnutelná (výuka cizího jazyka).

Video – má ta samé omezení jako obrázky. Velmi efektivní je např. při výuce cizích jazyků použití videa v originálním znění + titulky.

Grafika – se řídí také svými pravidly. Např. jednotlivé obrazovky programu by měli být konzistentní, což znamená, že ovládací prvky budou vždy na stejném místě se stejným stylem stránky. Je důležité výrazně odlišit část pro text a pro ostatní prvky (barevně). S barvami se obecně doporučuje šetřit (např. pravidlo tří barev). Co se týče vzájemné kombinace upřednostňuje se kombinace tmavého textu se světlým pozadím před tmavým pozadím s kontrastním textem, jelikož je to méně unavující pro oči. Druhá kombinace se používá pouze tam kde je malé množství textu (mobily s malý displayem, ne PDA).

2.2.5 Struktura m-Kurzu

Výuková struktura programu v pojetí skrze jednotlivé obrazovky vypadá v nejkomplexnějším případě následovně:



Obrázek 5 : struktura výukového programu

Motivační obrazovka - motivuje žáka položením otázky, navozením problému, inspiruje použitím protikladů či porovnání dvou situací.

Prezentační obrazovka – slouží k předložení učiva. Používá tabulku pojmů podle cílů, prezentuje učivo po malých dávkách **rychlostí, kterou si řídí sám žák.** Žák musí být pro další pokračování výuky fyzicky aktivní (musí potvrdit, posunout).

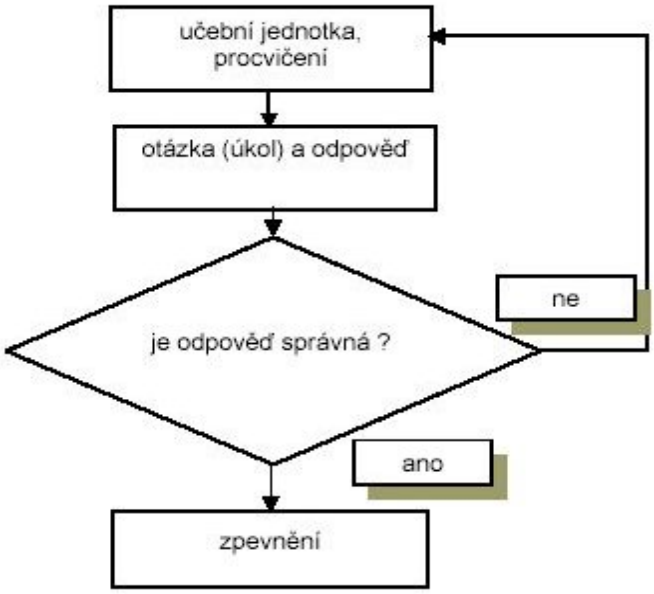
Rozšiřující informace – realizují zobrazení doplňujících informací pro náročné. Tato obrazovka ctí rozdíl “je dobré vědět – je nutné vědět”.

Procvičovací obrazovka – používá různé metody pro procvičení čerstvě nabraných znalostí či dovedností. Důležité je žáka donutit k aktivní odpovědi.

Zpevnění informace(zpětná vazba) – Je realizována skrze otázky. Nejpoužívanější typy otázek jsou :

1. s alternativní odpovědí (správně/chybně)
2. s vícenásobnou odpovědí (jeden výběr, více výběrů)
3. vzájemné přiřazení (většinou s použitím techniky přesuň)
4. volby objektu (většinou potvrzením kurzoru)
5. doplnění textem (otázka doplňovací)

Existuje několik způsobů řešení zpětné vazby, nejlepší z nich při nesprávné odpovědi na otázku zobrazí link odkazující na oblast, kterou se otázka zabývá. V opačném případě je nutné v žákovi zpevnit nabyté informace např. pochvalou, ale vždy se u ní musí zopakovat správná odpověď (pro připomenutí, zpevnění informace). Vývojový diagram zpětné vazby vypadá následovně:



Obrázek 6 : struktura zpětné vazby

Příklady aplikací a shrnutí – přenáší v příkladech teoretické poznatky na praktickou realitu a shrnuje poznatky.

Závěrečný test – neslouží k procvičení nabytých informací, ale ke zjištění kolik se toho žák naučil. Zařazuje se po několika lekcích a zkouší ze všech najednou. Měří jaké úrovně žák dosáhl. Neobsahuje přímou vazbu zpět na učivo. Je velmi důležité správně a jednoznačně postavit otázky (kritické místo programů).

2.2.6 Dodatečné informace

Do teď jsme se zabývali pouze učivem a jeho zpracování. Náš m-Kurz musíme ale doplnit i o další informace. Musíme uživatele informovat o tom pro koho je kurz určený, co se v něm dozví, možnosti navazujících/rozšiřujících informací, technické parametry. Toto vše mu ale musíme sdělit před vstupem (stažením do svého zařízení) do kurzu, nemůžeme nutit uživatele stahovat

si program za drahé poplatky (např. GSM poskytovateli), bez toho aby věděl, zda e-kurz je pro něj vhodný. Proto bychom měli informace uvádět např. na stránkách odkud si lze mobilní program stáhnout, nebo u WAP kurzu zásadně před registrací do kurzu.

3 Java a J2ME

Tato kapitola byla psána za pomoci [4] až [12].

3.1 Rozdělení jazyků

Nejspíš každý kdo se pustí do čtení této práce ví, že pro zápis programů používáme programovací jazyky. My se zde budeme zabývat jazykem Java, ale než se pustíme do jeho popisování řekneme si něco o programovacích jazycích obecně.

Programovací jazyky lze rozdělit podle toho, zda je výsledný program překládaný, interpretovaný nebo kombinací obou z nich.

Překládaný program se po napsání předá kompilátoru (překladači), který program zkompiluje (přeloží) do kódu, kterému rozumí daná platforma(rozumějme hardware počítače + OS). Poté se na této platformě může na požádání spustit. Pokud ale chceme program spustit na jiné platformě, musíme program předělat, tak aby mu rozuměla daná platforma a znovu přeložit (objem práce je většinou stejný, jako při psaní první verze).

Interpretovaný program předáme interpretu, který program prochází a zároveň provádí. Není to tak rychlé jako u předchozího způsobu, protože interpret musí přečíst kus programu, zjistit co dělá a poté až to provést. Jejich předností je ovšem to, že jim nezáleží na jaké platformě běží, pokud tam běží příslušný interpret. Dá se tedy říci, že platformou těchto programů je onen „interpret“, stačí pro novou platformu vyvinout interpret a můžou se v něm spustit všechny programy.

Mimo tyto druhy existuje ještě jedna možnost a to je jejich kombinace zvaná **„hybridní“** program . Tato skupina programů spojuje výhody obou předchozích. To znamená, že program je přeložen do jakéhosi mezijazyka, který je vytvořen pro co nejrychlejší interpretaci. Takto přeložený program poté interpretuje speciální interpret zvaný „virtuální stroj“ . Virtuální stroje jsou konstruovány tak, aby poznali co je nejlepší pro rychlý běh programu. Např. často používaný kód si dají stranou, aby se nemusel stále dokola překládat, nebo naopak ho přeloží znovu, protože může využít speciálních právě platných podmínek, za kterých to bude rychlejší. Nakonec tedy běží i rychleji než program překládaný a přesto je nezávislý na platformě.

3.2 Java

Java je velmi oblíbený a tudíž i velmi rozšířený programovací jazyk, což je zapříčiněno jejími přednostmi, kterých je hned několik.

Jako první bych uvedl, že Java je objektově orientovaný jazyk. Tento přístup je v dnešní době velmi rozšířený a používají ho jazyky jako je C#, či nové PHP. O výhodách a nevýhodách objektově orientovaného přístupu (OOP) nemá cenu se zde bavit, jelikož to není předmětem zájmu této práce a bylo o tom napsáno již hodně obsáhlých knížek.

Další výhodou je jednoduchost její základní syntaxe, kterou se lze naučit za pár hodin a další studium probíhá již jako poznávání jednotlivých knihoven, jejich tříd a metod. Tato jednoduchost ale není dána omezeností jazyka, protože v Javě se programují aplikace všech možných rozměrů i možností využití (mobilní zařízení, PC, web).

Asi poslední takto výraznou předností je multiplatformnost Javy, jelikož je Java jazyk hybridní. To znamená, že se program nejdříve přeloží do tzv. „bajt kódu“ , který následně Javovský Virtuální stroj (Java Virtual Machine) analyzuje a interpretuje. Tento virtuální stroj umožňuje, aby jeden program fungoval na různých počítačích a operačních systémech. Pro každou kombinaci hardwaru a OS lze vytvořit vlastní virtuální stroj. Dnes Java běhá pod systémy Windows, Linux, Unix, MacOS, Solaris a mnoho dalších operačních systémů, nevyjímaje ani mobilní zařízení. Tento Virtuální stroj tvoří v podstatě vlastní platformu Javy, což byl od začátku záměr autorů. Java má tedy vlastní platformu, která se dělí na tři druhy, podle toho na jakém zařízení ji chcete provozovat, neboli na JSE, JEE, JME.

Java Standard Edition (JSE) – je základní platforma pro vývoj aplikací na osobní PC a jednodušších verzí serverových aplikací.

Java Enterprise Edition (JEE) – je nadstavba platformy J2SE, rozšířená o knihovny pro tvorbu rozsáhlých distribuovaných aplikací.

Java Micro Edition (JME) – je nejmenší verze, neboli je to omezená a v některých třídách i malinko pozměněná J2SE. Je určena pro vývoj aplikací na nejmenších zařízeních, jako jsou mobilní zařízení. Tuto platformu budeme používat i my.

Pokud chcete, aby na vašem PC byli funkční programy pro Javu (JSE), musíte si stáhnout onen virtuální stroj. Pro deskopové počítače se dají stáhnout dva druhy a to JRE a JDK.

Java Runtime Environment (JRE) je prostředí pro běh programů v Javě, obsahuje jen interpret Javy a standardní knihovny. Neboli po jeho nainstalování vám budou fungovat javovské programy.

Java Development Kit (JDK) - obsahuje JRE a je rozšířena o překladač a další vývojové nástroje. Neboli potom můžete programy i vyvíjet. Aktuální verze je verze 6.x .

Tak a nyní v tom máte nepořádek, protože je Java SE a přitom JDK 6.0, tak jak to tedy s tou verzí Javy je? Skutečnost je taková, že celé označení aktuální verze je 1.6.X (kde X je aktuální update, vyladění chyb) a ne pouze 6.0. Co které z těchto čísel znamená je znázorněno na následujícím schématu.

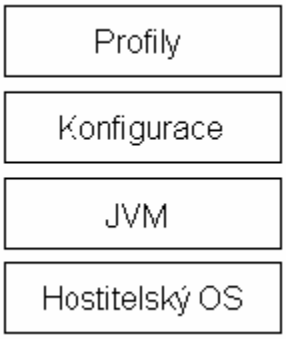
```
JDK 1.6.0
     | |  |
     | |  +-- bug fix number (větší číslo -> méně chyb)
     | +----- minor verze (mění se, dojde-li ke změnám
     |        či rozšířením v jazyce nebo knihovnách)
     +------- major verze (interně zatím vždy 1)
```

Poté co se v Javě objevil balíček Swing pro tvorbu uživatelských rozhraní, byl to takový skok kupředu, že se Java dlouhou dobu označovala jako se Second Edition neboli J2EE, J2SE a J2ME. V dnešní době se od toho již upusilo. Pro větší chaos se od **JDK 1.5** nazývá **JDK 5.0** atd..

3.3 Java ME

V minulé části jsme si řekli, že J2ME je Java platforma pro nejmenší zařízení jako jsou PDA, mobilní telefony, GPS navigace, pagery a dokonce i čipové karty. Když se podíváme do útrob této platformy zjistíme, že se skládá z několika částí a to jsou JVM (Java Virtual Machine, který je pro každé zařízení jiný), skupina knihoven + API (chovají se na každém JVM stejně, nazývají se konfigurace a profily) a nástroje pro vývoj a nastavení malých zařízení.

Co znamenají ty zmíněné profily a konfigurace? Když si představíme strukturu programů, tak základem je OS, na kterém běží JVM. Konfigurace je nad JVM a obsahuje knihovny zajišťující základní funkce. Na špici konfigurace sedí jeden či více profilů, které obsahuje další knihovny pro zajištění funkcí podobných malých zařízení. Názorně je tato struktura zobrazena na následujícím obrázku. č. 7



Obrázek 7: struktura Javy ME

3.3.1 Konfigurace

J2ME nabízí v dnešní době dvě standardní konfigurace. Jsou to Conected Device Configuration (CDC) a Connected Limited Device Configuration (CLDC). Z těchto dvou je už podle názvu jasné, že CLDC je určena pro výpočetně méně výkonné zařízení.

Connected Device Configuration - je určena pro výkonnější zařízení s častým připojením do sítě jako jsou komunikátory, GPS navigace a další. Tato konfigurace používá plnohodnotný virtuální stroj, který je velmi podobný JVM z J2SE. Pouze je přizpůsoben zařízením s nižším výpočetním výkonem. Virtuální stroj pro tuto konfiguraci je tedy **Compact Virtual Machine (CVM)** . Pro běh CDC konfigurace musí zařízení být řízeno 32-bitovým procesorem, mít 2MB a více prostoru pro běh Javy a programu v RAM, popřípadě s FLASH či ROM paměti. Zařízení může být připojitelné do nějakého druhu sítě a může mít propracované uživatelské rozhraní – není podmínkou.

Connected Limited Device Configuration - neklade takové nároky na výkon zařízení jako ta předešlá. Lze nahrát pomalu do jakéhokoli zařízení, proto je také velmi rozšířená a pro J2ME častější. Virtuální stroj, který používá tato konfigurace se nazývá **Kilobyte Virtual Machine (KVM)**. Jmenuje se tak, jelikož ke svému běhu potřebuje pouze několik kilobajtů paměti. Přesto je to kompletní Javovské prostředí pro malá zařízení. Má pouze některé odchylky dané limitujícími omezeními malých zařízení (omezené zdroje a málo paměti). Zařízení pro běh CLDC stačí pouze 160 – 512kB paměti (opět RAM případně s FLASH či ROM). Může také

být připojitelné k některé síti a mít propracované uživatelské rozhraní, ale opět to není podmínkou.

3.3.2 Profily

Zavedením profilů se definovali prostředí pro rozdílná zařízení na shodné či velmi podobné úrovni. Tvoří tedy jakousi nadstavbu konfigurací skládající se z programátorských rozhraní, které poskytují přístup programům ke specifickým vlastnostem různých zařízení. Z pohledu programátora je profil rozhraním, nad kterým staví svoje aplikace. Nyní si zde všechny postupně probereme.

MIDP (Mobile Information Device Profile) – Dnes asi nejpopulárnější a nejznámější profil určený pro mobilní telefony a pagery. Tento profil běží na konfiguraci CLDC. Obsahuje třídy pro uživatelské rozhraní, trvalé ukládání dat i práci v síti. Existují pro něj vývojová prostředí, které umožňují ukládat aplikace nazývající se MIDlety do zařízení.

PDA Profil je pro práci s CLDC, poskytuje API pro uživatelská rozhraní a API pro ukládání dat v PDA zařízeních.

Základní profil (The Foundation Profile) – je profilem nad CDC konfigurací, který kompletně využívá tuto konfiguraci, ale nedodává žádná API pro uživatelské rozhraní, také neobsahuje knihovny `java.beans`, `java.rmi` ani `java.sql`. Je velmi univerzální a tudíž dostupný všem zařízením pro CDC. Je základem

pro funkčnost ostatních tří profilů (osobní, RMI, game), což vyplývá již z názvu. Vyžaduje 1 MB ROM a 512 kB RAM.

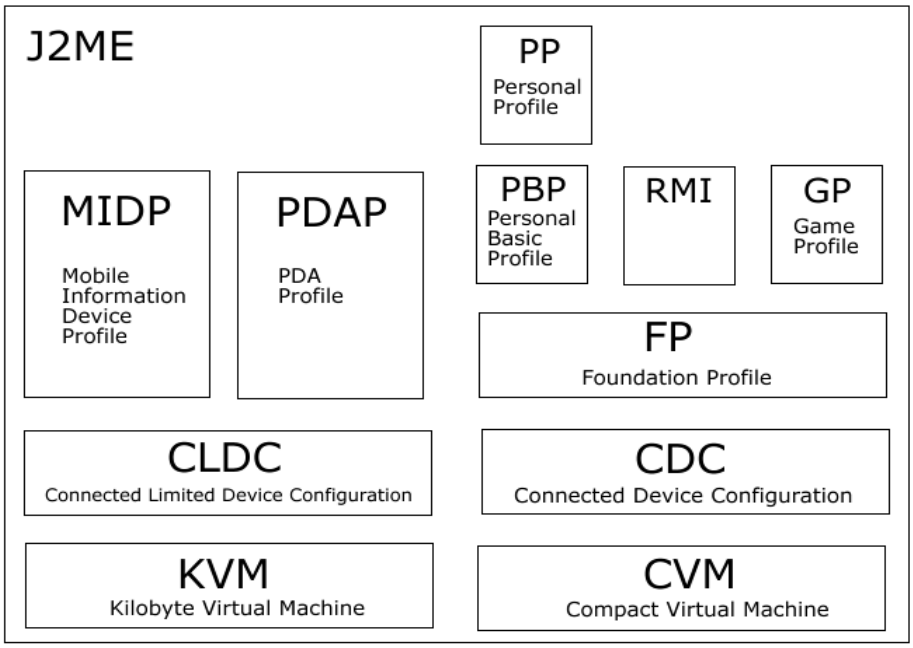
Základní osobní profil (Basic Personal Profile) - přidává základní uživatelské rozhraní, avšak použití tohoto rozhraní je omezeno pouze na jednu instanci obrazovky (screen). Přidává třídy `java.util.zip, java.util.jar`.

Osobní profil (Personal Profile) – rozšíření základního profilu pro větší zařízení jako jsou PDA nebo komunikátory. Obsahuje kompletní podporu Java AWT (Abstract Window Toolkit). Vyžaduje 2.5MB ROM a alespoň 1MB RAM.

RMI profil (Remote Method Invocation profile) – je rozšířením o RMI (volání vzdálených metod). Pro svou funkčnost také potřebuje mít nainstalován základní profil, což znamená že je pouze pro CDC.

Herní profil (Game Profile) – Jak už název napovídá, měl sloužit k psaní herního softwaru, zde se ovšem vývojáři dostali do slepé uličky a vyhodnotili tento profil jako neperspektivní a tato větev vývoje byla zastavena.

Pro snadnější představení popsané složité situace v J2ME si její strukturu ukážeme na obrázku na kterém je situace přehledně znázorněna.



Obrázek 8 : vnitřní struktura Javy ME

Podrobné informace o obou z konfigurací, či o všech profilech jsou velmi rozsáhlé, proto se v dalších částech práce budeme zabývat podrobněji pouze těmi částmi J2ME, které budou potřebné pro vývoj naší výukové aplikace.

3.3.3 Connected Limited Device Configuration

Jak jsem již v předchozí části uvedl, že CLDC je nejmenší část Javy, kterou lze použít pro různá zařízení. Přesto definuje standardní platformu s minimálními požadavky na zařízení. Specifické rysy zařízení ale neobsahuje, jelikož ty definují až profily. Základní vlastností CLDC je, že veškeré vlastnosti lze

využít ve všech zařízeních podporujících tuto konfiguraci. V dnešní době jsou k dispozici dvě verze a to CLDC 1.0 a 1.1. Nyní si blíže specifikujeme požadavky na zařízení.

160 až 512KB celkové paměti se skládá z minimálně 128 kB paměti pro JVM a knihovny obsažené v CLDC minimálně 32 kB dočasné paměti využitelné pro práci JVM (paměť RAM).

16-bitový nebo 32-bitový procesor s minimální taktovací frekvencí 25 MHz, což jsou typické procesory u malých mobilních zařízeních.

Připojitelnost k některému druhu sítě - často bezdrátových, přerušovaným připojením s omezenou šířkou pásma.

Primární napájení z baterie – zařízení mají nízkou spotřebu energie, jelikož pracují na bateriích, což se odráží v jejich výkonu.

I když má CLDC tato omezení poskytuje soubor rysů Javy a virtuálního stroje, soubor základních knihoven Javy (`java.lang.java.util`), základní vstup/výstup (`java.io`), základní podporu sítí (`javax.microedition.io`) a zabezpečení. Z našeho výčtu je zřetelné, že konfigurace (CLDC) nezasahuje do běhu aplikací, do uživatelských rozhraní, do správy událostí, či do interakce mezi uživatelem a aplikací. Tyto akce běží nad konfigurací a obstarávají je profily, např. MIDP.

Chybějící aspekty VM pro CLDC

Podpora plovoucí řádové čárky znamená, že se v programech nesmějí používat datové typy s touto plovoucí řádovou čárkou, což jsou `float` a `double`. Malé zařízení podporující CLDC neobsahují hardware pro počty s plovoucí řádovou čárkou, platí pouze pro CLDC 1.0.

Omezené možnosti zpracování chyb – obsaženy jsou pouze čtyři základní třídy pro zpracování chyb (`java.lang.Error`, `java.lang.OutOffMemoryError`, `java.lang.NoClassDefFoundError` `java.lang.VirtualMachineError`), chyby nesouvisející s během programu zpracovává zařízení podle svých možností (aplikací ukončí, nebo se restartuje).

Finalizace – v CLDC není obsažena metoda `Object.finalize()` nelze provádět závěrečné úklidové operace s objekty, dříve než je daný objekt zrušen garbage collectorem .

JNI (Java Native Interface) obsaženo ve virtuálním stroji CLDC kvůli bezpečnosti a výpočetní náročnosti

Uživatelem definované zavaděče tříd (class-loadery) nejsou definovány, jelikož je zabudován interní zavaděč tříd, který se nedá uživatelem nahradit, ani odstranit pro větší bezpečnost.

Podpora reflexe - nemohou být používány API pro reflexi, z čehož vyplývá, že chybí serializace objektů a RMI.

Slabé odkazy – na JVM pro CLDC nemohou být používány slabé odkazy.

Skupiny vláken, nebo procesy typu démon nejsou podporovány, přestože JVM pro CLDC podporuje vícevláknové zpracování. Musíme tedy pro operace uložení objektů skupiny vláken, použít skupiny objektů na úrovni aplikace.

Knihovna CLDC

Jelikož budeme s touto konfigurací pracovat v poslední kapitole, řekneme si něco i o její knihovně. Když se podíváme na paměťové požadavky CLDC a vezmeme v úvahu skutečnost, že programové rozhraní JSE má cca 20MB, musí nám být jasné, že knihovna byla pozměněna a to na minimum dostačující pro vývoj programů a profilů. Knihovna CLDC 1.0 dědí 37 tříd ve třech balíčcích z J2SE (CLDC 1.1 byla ještě rozšířena) a zbytek jsou její vlastní.

V CLDC 1.0 byli převzaty tyto třídy:

`java.lang`

Třídy — `Boolean, Byte, Character, Class, Integer, Long, Math, Object, Runtime, Short, String, StringBuffer, System, Thread, Throwable`

Rozhraní - `Runnable`

java.io

Třídy - `ByteArrayInputStream, ByteArrayOutputStream, DataInputSream, DataOutputStream, InputStream, OutputStream, InputStreamReader, Reader, Writer OutputStreamWriter, PrintStream,`

Rozhraní - `DataInput, DataOutput`

java.util

Třídy – `Calendar, Date, Hashtable, Random, Stack, Timezone, Vector`

Rozhraní – `Enumeration`

V CLDC 1.1 byla rozšířena o tyto třídy:

java.lang

Třídy – `Double, Float`

java.lang.ref

Třídy – `Reference, WeakReference`

Převzaté výjimky:

java.lang – `ArithmeticException, ArrayIndexOutOfBoundsException, ArrayStoreException, ClassCastException, ClassNotFoundException, Error, Exception,`

IllegalAccessException,
IllegalArgumentException,
IllegalMonitorException,
IllegalThreadStateException,
IndexOutOfBoundsException,
InstantiationException,
InterruptedException,
OutOfMemoryException,
NegativeArraySizeException,
NumberFormatException,
StringIndexOutOfBoundsException,
VirtualMachineError

java.io – EOFException, IOException,
InterruptedIOException,
UnsupportedEncodingException,
UTFDataFormatException

java.util – EmptyStackException,
NoSuchElementException

Z těchto tříd byly ve třídách `Math`, `String` a `StringBuffer` odstraněny metody a datové typy používající plovoucí řádovou čárku. Dále se v balíčku `java.io` nepřevzaty třídy `FileInputStream` a `FileOutputStream`, místo nichž se používají `FileReader` a `FileWriter`.

Také nejsou převzaté žádné třídy pro práci v síti, jelikož nebylo možno zachovat podobnost se třídami standardních edicí. Je tedy vytvořen balíček `javax.microedition.io` pro obecné IO operace v síti.

Jako poslední rozdílnost si uvedeme nezdědění třídy `java.util.Properties` a její nahrazení čtyřmi systémovými vlastnostmi (`encoding, platform, profiles, configuration`), které lze zjistit metodou `System.getProperty (String key)`.

3.3.4 KVM

Jelikož jsme si podrobně probrali vlastnosti CLDC, měl bych vám zde říci i něco více o jejím VM, což je jak jsme si již řekli **KVM**. Tento VM je napsán v programovacím jazyce C, neboli lze ho přenést na jakoukoli platformu s kompilátorem jazyka C. Je to pomalu plnohodnotný Javovský VM pouze s malými odchylkami pro specifické požadavky malých zařízení. Není zahrnuto několik rysů, jelikož jejich zavedení by bylo moc náročné z hlediska procesoru nebo paměti. KVM je tedy velmi minimalistický (pouze pokud nejde o rychlost) a stačí mu k provozu 6-bitové a 32-bitové CISC nebo RISC procesory a procesory s rychlostí minimálně 25 MHz a 80KB statické paměti.

Největší problém při zmenšování jeho nároků byla verifikace tříd (odstranění neplatných class souborů během běhu programu), což je náročný proces, který obvykle zabere od 35 do 110 KB pracovní paměti. Jenže KVM je konstruován pro zařízení s pamětí

od 50 do 80 KB pro VM, ale tuto funkci musí přesto VM povinně provést. Jak se to řeší?

Tvůrci to řešili tak, že většinu operací verifikace převedli na klasické PC, které všechny `class` soubory kompilují. Tomuto postupu, kdy se verifikuje na jiném zařízení, se říká **preferifikace** (předběžné ověření). Při tomto procesu do `class` souborů přidají informace pro verifikátor, který pak může pracovat s menší pamětí. Dále se do mapy zásobníku přidá atribut nesoucí informace o kritických oblastech tříd podle kterého verifikátor provádí kontrolu. Po odebrání těchto povinností z VM se verifikátor stará jen o rychlé přečtení class souborů, které zjišťuje jejich verifikovanost a zda neobsahují žádné nepovolené instrukce. Při této proceduře se snížila velikost paměti potřebná běhovým verifikátorem na cca 100B.

Po tom co jsme vysvětlili jak probíhá proces verifikace v CLDC si můžeme říci i něco o jejím zabezpečení, které se zaměřuje na dvě oblasti:

Zabezpečení na úrovni virtuálního stroje - aplikace pracující v KVM nesmí ohrozit zařízení, na kterém běží. Za to je zodpovědný verifikátor tříd, který se stará o to, aby bajtové kódy tříd neodkazovaly na neplatná místa v paměti.

Zabezpečení na úrovni aplikace - nelze upravovat správce zařízení (jako je tomu u J2SE). Zabezpečovací model KVM poskytuje zabezpečení tím, že zajišťuje běh aplikace v uzavřeném

prostředí, a tím, že aplikace mohou volat pouze třídy podporované samotným zařízením.

3.3.5 Connected Device Configuration

Konfigurace CDC je určena pro větší zařízení jako jsou PDA, komunikátory, GPS navigace a další. Nebudeme ji tedy v poslední kapitole aktivně využívat, proto se zde o ní zmíním krátce jen pro úplnost. Jelikož zařízení pro která je určen, disponují větším výkonem, může si oproti CLDC dovolit více funkcí. Obsahuje celou knihovnu pro CLDC zařízení doplněné o několik dalších tříd a metod, neboli je to její nadmnožina. Neobsahuje omezení která skýtala CLDC. Požadavky na zařízení:

2MB a více – pravé minimum pro CDC je 512kB v ROM a 256kB RAM

32-bitový procesor – opravdu je vyžadováno zařízení s 32b procesorem

Připojitelnost k některému druhu sítě - často bezdrátových, přerušovaným připojením s omezenou šířkou pásma.

Primární napájení z baterie – zařízení mají nízkou spotřebu energie, jelikož pracují na bateriích, což se odráží v jejich výkonu.

Knihovna CDC

Knihovna CDC opravdu tvoří nadmnožinu CLDC neboli obsahuje stejné třídy, pouze beze změn kvůli plovoucí řádové čárce a navíc přidává balíčky:

- `java.net`
- `java.text`
- `java.security`

3.3.6 MIDP (Mobile Information Device Profile)

Jelikož bude naše výuková aplikace určena pro mobilní telefon, budeme se podrobně zabývat jen tímto profilem. MIDP je profil J2ME pro mobilní zařízení jako jsou mobilní telefony a pagery, který definuje otevřené prostředí pro vývoj aplikací. K dispozici jsou celkem tři verze – 1.0, 2.0 a 3.0. Profil MIDP je vrcholem na pyramidě od OS přes KVM až k CLDC, což znamená, že je nejvýše a není nad ním už nic. Z toho vyplývá, že musí řešit oblasti, které neřešila CLDC ani KVM. Verze 1.0 řeší základní oblasti jako je správa průběhu aplikací či ukládání dat v zařízení. Druhá verze 2.0 řeší ty samé oblasti, ale rozšiřuje předchozí verzi např. o práci s multimédii nebo větší podporu pro vývoj her. Tento standard definuje následující požadavky na mobilní zařízení:

Display – minimální velikost 96x54px s barevnou hloubkou minimálně 1bit.

Vstupní zařízení - klávesnice pro jednu (klávesnice na mobilním telefonu) nebo pro obě ruce (QWERTZ/Y klávesnice) nebo dotykový displej .

Paměť – MIDP 1.0 – 128kB trvalé paměti pro aplikaci + 8kB pro trvale uložená data a 32kB dočasné paměti

MIDP 2.0 – 256kB trvalé paměti pro aplikaci + 8kB pro trvale uložená data a 128kB dočasné paměti

Síť – obousměrný přerušovaný síťový provoz, realizovaný většinou bezdrátovým spojením s omezenou šířkou pásma

Profil MIDP obstarává následující oblasti

Správa průběhu aplikací – zprostředkovává balíček `javax.microedition.midlet` obsahující třídy a metody pro spuštění, přerušení a vymazání aplikace z hostitelského prostředí.

Uživatelské rozhraní a události – balíček `javax.microedition.lcdui` obsahuje třídy pro tvorbu uživatelského rozhraní a práci s 2D grafikou .

Připojitelnost k síti – MIDP 1.0 - obsahuje rozhraní HttpConnection pro protokol http

MIDP 2.0 -je rozšířeno o rozhraní HttpsConnection pro šifrovaný protokol https .

Ukládání dat v zařízení – poskytuje balíček `javax.microedition.rms`, umožňující databázový systém ukládání dat.

Knihovna MIDP 1.0

javax.microedition.midlet

Třída – `MIDlet`

Výjimka – `MIDletStateChangeException`

javax.microedition.lcdui

Třídy – `Alert, AlertType, Canvas, ChoiceGroup, Command, DateField, Display, Displayable, Font, Form,Gauge, Graphics, Image, ImageItem, Item, List, Screen, StringItem, TextBox, TextField, Ticker`

Rozhraní – `Choice, CommandListener, ItemStateListener`

javax.microedition.rms

Třída – `RecordStore`

Rozhraní – `RecordComparator, RecordEnumeration, RecordFilter, RecordListener`

Výjimky – `InvalidRecordIDException, RecordStoreException, RecordStoreFullException, RecordStoreNotFoundException, RecordStoreNotOpenException`

Rozšíření CLDC

java.util

Třídy – `Timer, TimerTask`

java.lang

Výjimka – `IllegalStateException`

Rozšíření knihovny MIDP ve verzi 2.0

javax.microedition.lcdui

Třídy – `CustomItem, Spacer`

Rozhraní – `ItemComanndListener`

java.microedition.lcdui.game

Třídy – `GameCanvas, Layer, LayerManager, Sprite, TiledLayer`

java.microedition.media

Třída – `Manager`

Rozhraní – `Control, Controllable, Player, PlayerListener`

Výjimka – `MediaException`

`java.microedition.media.control`

Rozhraní – `ToneControl, VolumeControl`

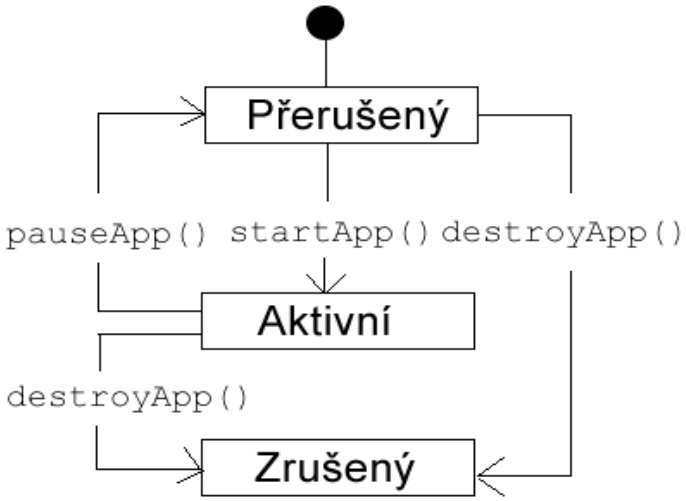
`java.microedition.pki`

Rozhraní – `Certificate`

Výjimka – `CertificateException`

3.3.7 MIDlet

MIDlet je aplikace, která běží na MIDP zařízeních. Každý MIDlet má stavy ve kterých se může nacházet. Stavy MIDletu jsou následující: zrušený, aktivní a přerušený. MIDlet musí být potomkem abstraktní třídy `javax.microedition.midlet.MIDlet` zajišťující metody pro řízení stavů MIDletu. Metody této třídy jsou: `pauseApp()`, `startApp()` a `destroyApp()`. Tyto metody volá Manažer aplikací, který řídí prostředí MIDletu. Situace je vyobrazena na následujícím obrázku, kde černý bod znázorňuje počáteční neaktivní stav MIDletu.



Obrázek 9 : Vnitřní stavy MIDletu

Manažer aplikací

Manažer aplikací se též nazývá systém řízení aplikací (Application Management System – AMS), nebo program pro řízení MIDletu (MIDlet management software). Je to předinstalovaný program na zařízení pro MIDP, vytvářející operační prostředí schopné získat soupravu MIDletů , instalovat na MIDP zařízení, spravovat jejich verze, spouštět je, zajistit operační prostředí pro KVM i všechny MIDP a CLDC třídy a nakonec je umí i vymazat.

3.3.8 Funkce, aneb jak to funguje

Každé zařízení má pro své jedinečné parametry svůj specifický manažer aplikací. Každý tento manažer musí provádět

stejné funkce pro přechod mezi stavy MIDletu. Tyto funkce si popíšeme v následujících odstavcích.

Když uživatel spustí MIDlet, vytvoří manažér aplikací novou instanci třídy MIDlet zavoláním jejího bezparametrického konstruktoru. Provede se inicializace po níž přejde MIDlet do stavu přerušení. Jestliže při vytváření instance nastane výjimka či chyba, přesune se MIDlet do stavu zrušený.

Poté co přešel MIDlet do stavu přerušený, zavolá manažer metodu `startApp()`, aby přešel do stavu aktivní.

Pak může manažer aplikací, na základě požadavku MIDletu nebo operačního prostředí, zavolat metodu `pauseApp()`, aby MIDlet přešel do stavu přerušený.

Metodu `destroyApp()` lze zavolat manažerem aplikací, aby převedla MIDlet do stavu zrušený. Metoda `destroyApp()` používá parametr typu boolean, indikující zda je či není MIDlet určen ke zrušení.

V předchozím textu byli použity několikrát názvy metod a příklady jejich použití. Pro lepší pochopení si nyní podrobně probereme funkce jednotlivých metod.

`public void startApp()` – volá se pokud MIDlet přechází ze stavu přerušeného do aktivního. Jsou inicializovány všechny potřebné objekty pro stav aktivity a nastaví se odpovídající obrazovka. Tato metoda se volá na počátku a následně při

přechodu z přerušeného stavu do aktivního. Proto pokud chceme provést nějakou akci pouze na počátku spuštění MIDletu, umístíme ji do konstruktoru třídy.

`public void pauseApp()` – volá se pokud MIDlet přechází ze stavu aktivního na stav přerušený. Přeruší se všechna vlákna, která jsou aktivní a může se nastavit obrazovka, která se objeví, když bude MIDlet opět aktivní. Data lze uložit a opět obnovit při přechodu do aktivního stavu.

`public void destroyApp(boolean unconditional)`– volá se pokud MIDlet přechází ze stavu aktivního nebo přerušeného do stavu zrušeného. Tato metoda by měla uvolnit všechny zdroje, které byly při průběhu MIDletu využívány. Dále by měla uložit data potřebná pro pozdější použití. MIDletu. Používá parametr typu `boolean`, která značí zda je MIDlet určen ke smazání.

Přechod mezi stavy MIDletu si může řídit i MIDlet sám a manažera pouze informuje o provedení změny stavu, využívá k tomu metody:

`public void notifyPause()` – Pokud MIDlet zavolá tuto metodu, sám se přepne do stavu přerušený. Metoda lze zavolat v aktivním stavu a tím informuje manažéra aplikací, že se MIDlet převede do přerušeného stavu .

`publid void resumeRequest()` – Pokud se chce MIDlet smazat zavolá tuto metodu. Je možné ji zavolat jak z aktivního tak

z přerušeného stavu. Tímto se upozorňuje manažer aplikací, že MIDlet přešel do zrušeného stavu a poté se uvolní všechny zdroje a uloží se data, která budou potřeba při dalším použití MIDletu.

`public void notifyDestroyed()` – Pokud se chce MIDlet smazat zavolá tuto metodu. Je možné ji zavolat ze dvou stavů z aktivního i z přerušeného. Cílem je upozornit manažera aplikací o přechodu MIDletu do stavu zrušeného. Poté se uvolní všechny zdroje a uloží se data, která budou potřeba při dalším použití MIDletu.

Souprava MIDletů

Soupravu MIDletů tvoří dva a více MIDletů, které jsou komprimované do JAR souboru. MIDlety ve stejné soupravě mohou používat i třídy a zdroje jiných MIDletů v JAR souboru obsažených. V JAR souboru jsou i všechny obrázky a soubory potřebné k běhu MIDletu. Vlastnosti tohoto souboru se dají zjistit dvěma způsoby.

JAR Manifest - je v adresáři META-INF v souboru JAR. Informace z manifestu používá program pro správu aplikací a to k identifikaci a instalaci soupravy MIDletů. Manifest ovšem není povinný.

Popisovač aplikací Javy JAD (Java Application Descriptor) - Je zřejmé, že používání manifestu sebou přináší problém v podobě potřebného nahrání celého JAR souboru do zařízení pro zjištění informací pro instalaci, která se ani nemusí provést, jelikož zařízení např. nemá potřebný profil. Proto vznikl JAD , což je textový soubor podobný manifestu obsahující také informace

s požadavky. Výhodou tohoto souboru je ta skutečnost, že není zabudován v JAR souboru, neboli se zařízení potřebné informace dozví ještě před nahráním JAR souboru.

Přehled popsaných vlastností obou souborů pro popis vlastností souboru JAR si ukážeme v následující tabulce, kde si uvedeme vlastnost, její popis, ukázku hodnoty a ve kterém popisovači se vyskytuje.

P - povinný údaj, V – volitelný údaj, x – údaj neobsažen

Vlastnost	**JAR**	**JAD**	**Popis vlastnosti**	**Ukázka hodnoty**
MIDlet-Name	P	P	Název MIDletu obsaženého v JAR souboru.	Test
MIDlet-Version	P	P	Číslo verze MIDletu, formát xx.yy.zz, kde xx je verze, yy podverze, zz číslo překladu, číslo překladu není povinné, je používáno pro případ instalace či upgradu soupravy	1.0.1
MIDlet-Vendor	P	P	název vývojáře	Test Inc.
MIDlet-n	P	x	název, ikona, třída	MujMIDlet, icony/moje.png , MojeTrida

MicroEdition-Profile	P	x	Profil, pro nějž je MIDlet určen. Tato hodnota se kontroluje při instalaci MIDletu na zařízení.	MIDP-1.0
MicroEdition-Configuration	P	x	Konfigurace, pro níž je MIDlet. Hodnota se kontroluje při instalaci MIDletu na zařízení.	CLDC-1.0
MIDlet-Description	V	V	Popis soupravy	Dva MIDety
MIDlet-Icon	V	V	Cesta k ikoně ve formátu PNG, která může být použita například během instalace	/icony/main.png
MIDlet-Info-URL	V	V	url popisu soupravy	http://mujmidlet.cz
MIDlet-Data-Size	V	V	minimální počet bajtů, které MIDlet potřebuje	256
MIDlet-Jar-URL	x	P	URL, kde se nachází JAR soubor s MIDletem.	http://mujmidlet.cz/download
MIDlet-Jar-Size	x	P	Velikost JAR souboru v bytech.	24354

Tabulka 1 : údaje v JAR a JAD

Grafické uživatelské rozhraní (GUI) v MIDP

Uživatelské rozhraní v MIDletu můžeme psát dvěma způsoby a to buď jako vysokoúrovňové a nebo nízkoúrovňové. Tyto dva

přístupy se liší především v použitých třídách a přenositelnosti aplikace mezi zařízeními.

Vysokoúrovňové GUI – poskytuje vysokou úroveň abstrakce, nedají se nastavit podrobné vlastnosti typu font atd.. Umožňuje tedy přenositelnost mezi různými typy mobilních zařízení. Využívá vysokoúrovňové API tvořené potomky třídy javax.microedition.lcdui.Screen.

Nízkoúrovňové GUI – takto konstruované uživatelské rozhraní používá velmi přesné údaje o umístění a kontrole grafických prvků. Používá se pro aplikace požadující náročné grafické prvky jako jsou například hry. Využívá nízkoúrovňové API tvořené třídami `javax.microedition.lcdui.Canvas` a `javax.microedition.lcdui.Graphics`.
Poskytuje pouze malou míru abstrakce. Není zaručena přenositelnost, jelikož využívá detailní informace o zařízení, které jsou specifické pro každé jednotlivé zařízení.

Aby mohl MIDlet něco zobrazit, je třeba získat přístup k jeho displeji, o který se stará třída `javax.microeditio.lcdui.Display`. Tato třída je jediným manažerem displeje a obsahuje metody, kterými se získávají informace o možnostech displeje. Dalším článkem je obrazovka (`javax.microedition.lcdui.Screen`), což je objekt spravující a organizující grafické objekty. Obrazovku je možné zobrazit metodou `setCurrent()` v již zmíněné třídě `Display`. Aplikace může obsahovat více obrazovek, v jednu chvíli je ale viditelná (zobrazena) pouze jedna.

3.3.9 Specifické požadavky aplikací

MIDP je sice vrcholem na naší pyramidě J2ME a upřesňuje požadavky a prostředky, ale i tento profil je univerzální. To znamená, že v sobě nemá obsaženy všechny technologie a specifické doplňky zařízení, které by se tedy aplikacemi nedali využít. Z tohoto důvodu existují volitelné balíčky rozšiřující základní podporu funkcí zařízení. Používají se pro specifické účely jako jsou hry, webové služby.`Tyto balíčky nejsou výrobci povinni implementovat. Tato skutečnost ale velmi narušuje přenositelnost složitějších aplikací. Nejznámější volitelné balíčky si popíšeme v následující tabulce.

Název	Označení	Popis funkcí
PDA Optional Package	JSR 75	Přístup k adresáři, kalendáři a file systému zařízení
API pro Bluetooth	JSR 82	Podpora pro bluetooth protokoly: RFCOMM, OBEX a Service Discovery protokoly.
Wireless Messaging API (WMA)	JSR 120	Podpora technologií SMS, Unstructured Supplementary Service Data (USSD) a Cell Broadcast Service (CBS)
Wireless Messaging API 2.0 (WMAPI 2.0)	JSR 205	Rozhraní pro přijímání a odesílání MMS zpráv.

Mobile Media API (MMAPI)	JSR 135	Vysoko úrovňové rozhraní k multimediálním a zvukovým schopnostem mobilního zařízení.
Jain Simple Presence	JSR 164	Podpora Session Initiation Protocol pro práci v IP sítích
J2ME Web Services	JSR 172	Základy XML, rozhraní pro tvorbu webového klienta atd.
Security and Trust Services API	JSR 117	Ochrana citlivých dat jako jsou hesla či bankovní spojení
Location API	JSR 179	Podává informace o současném fyzickém umístění Java aplikací
SIP API (Session Initiation Protocol)	JSR 180	Pro vytvoření a správu multimediálních IP relací.
Mobile 3D Graphics	JSR 184	Porpora 3D scén
Event Tracking API	JSR 190	Podpora sledování události aplikace a tyto předávání dat serveru
Advanced Graphics and User Interface	JSR 209	Část jádra z J2SE s prostředky: Swing, Image I/O, Java 2D Graphics and Imaging, Input Method Framework.
Scalable 2D Vector Graphics API	JSR 226	Podpora zobrazení 2D vektorové grafiky i SVG souborů, umožňuje zobrazit obrázek podle velikosti displeje.
Payment API	JSR 229	Slouží k zahájení platební transakce.

Advanced Multimedia Supplements	JSR 234	Přistup k fotoaparátu a rádiu v mobilním zařízení.
Mobile Telephony API	JSR 253	Umí zahájit volání, kontrolovat ho a ukončit nebo přijmout příchozí hovor

Tabulka 2 : rozšiřující Java API

Další možná rozšíření

V minulé části jsme si řekli, že přenositelnost Java aplikací není vůbec tak snadná, jak se může na první pohled zdát. Dalším fakt komplikující přenositelnost, je skutečnost, že výrobci mobilních telefonů vyvíjí své vlastní aplikační programové rozhraní (API). Může se to zdát jako nelogický krok, když to ruší přenositelnost, ale má to své důvody. Důvodem počátků vývojů vlastních API byla malá podpora vlastností zařízení v MIDP 1.0, jako např. grafické rozhraní pro hry atd.. Krokem kupředu byla verze 2.0, ale ani ona neobsahuje stěžejní funkce jako jsou přístup do file systemu, k telefonímu seznamu, fotoaparátu či odesílání SMS zpráv a jiné.

Nyní si pro názornost ukážeme jako příklad obsah a funkce dvou API a to pro telefony Siemens a Samsung:

SIEMENS

com.siemens.mp.game - ovládá vibrace, podsvětlení, pokročilé grafické funkce, práce se zvuky, přehrávání a skládání melodií, animaci grafických objektů, nastavení pozadí .

com.siemens.mp.gsm - pro čtení SMS zpráv, volání, prohlížení telefonního seznamu

com.siemens.mp.io - přístup aplikace do vlastního adresáře a podadresáře a v něm využití file systému, posílání dat pomocí infraportu, přenos dat mezi aplikacemi pomocí SMS zpráv

com.siemens.mp.io.file - lze pracovat se soubory na paměťové kartě. Lze procházet adresáře, vytvářet nové soubory, mazat soubory, kopírovat, zkracovat délku souborů a zapisovat do souborů. Pro zachování bezpečnosti se mobilní telefon při pokusu aplikace k přístupu na pamětovou kartu vždy dotáže uživatele, zda si přeje povolit aplikaci přístup.

com.siemens.mp.media - umožňuje přehrát multimediální soubory typu Windows Media Audio (WMA) a Moving Picture Editors Group (MPEG), podporované typy souborů a protokolů a přehrávání tónů jsou zahrnuty v třídách Manager a Control

com.siemens.mp.media.control - nastavení hlasitosti

com.siemens.mp.ui - pro pokročilou práci s obrázky: transparentnost, zrcadlení atd.

SAMSUNG

com.samsung.util.AudioClip – pracuje s audio klipy na zařízení play, stop, resume atd.

com.samsung.util.LCDLight – velmi jednoduchá třída pro práci s podsvícením telefonu, buď ho zapne a nebo vypne

com.samsung.util.SM – pracuje s čísly pro zpáteční hovory

com.samsung.util.SMS – umí posílat SMS

com.samsung.util.Vibration –podpora práce s vibracemi zařízení

4 JME a NetBeans

Tato kapitola byla psána za pomoci [13] až [18].

Už jme si ukázali principy m-Learningu, seznámili jsme se s jedním programovacím jazykem a jeho strukturou pro mobilní zařízení. Nyní pro psaní výukové aplikace nám již chybí pouze vývojové prostředí, neboli aplikace ve které budeme náš program vyvíjet. Vývojových prostředí pro Javu je velké množství. Zde si uvedeme několik nejznámějších:

BlueJ - je integrované prostředí určené primárně pro výuku objektově orientovaného programování v Javě.

JEdit - programátorský editor napsaný v Javě s podporou editace velkého množství programovacích jazyků a formátů. Po instalaci je vhodné doinstalovat pluginy pro snadnější práci.

JCreator - Programátorský editor pro Javu napsaný v jazyku C s nízkými nároky na technické vybavení (postačuje 32MB RAM). Existuje freeware verze a shareware verze (která navíc např. doplňuje kód). Na pomalejších počítačích je vhodným doplňkem překladač jikes, který výrazně urychluje překlad.

CodeGear (bývalý JBuilder) - úspěšné komerční vývojové prostředí pro Javu, nároky na vybavení počítače jsou přiměřené (doporučeno 512MB RAM, Pentium III, monitor s min. rozlišením

1024x768). Studenti mohou legálně a bezplatně používat Foundation verzi, která je částečně funkčně omezena.

Eclipse - rozsáhlé vývojové prostředí s rozsáhlou podporou vývojových nástrojů. Jeho vývoj je podporován firmou IBM s cílem vytvořit obecné vývojové prostředí s velkým množstvím nástrojů. Mezi vývojáři je poměrně oblíbené. Doporučené technické vybavení: 512MB RAM, Pentium III. Lze doplnit o mnoho nástrojů, mezi jinými EclipseME pro vývoj v JavěME.

NetBeans je dnes vlajkovou lodí mezi vývojovými prostředími pro Javu podporující i jiné programovací jazyky (Ruby, PHP, C/C++, …), nyní rozvíjené jako OpenSource s podporou firmy Sun. Podporuje vytváření GUI "klikáním"(grafický designer), má poměrně rozsáhlou podporu XML. Doporučené technické vybavení: 512MB RAM, Pentium III.

Z těchto prostředí si vybereme to, které podporuje psaní aplikací pro mobilní telefony, je kompletně zdarma a dnes se dostává do popředí mezi všemi vývojovými prostředími pro svou propracovanost a univerzálnost neboli **NetBeans**.

4.1 NetBeans

Je univerzální vývojové prostředí primárně určené pro Javu, ale podporuje i ostatní jazyky(C/C++, PHP, JavaScript, Ruby). Toto prostředí je kompletně napsané v Javě, funguje na většině operačních systémech (např. Linux, Windows, Solaris OS a Mac OS) a nabízí vývojářům mnoho nástrojů pro vývoj profesionálních aplikací pro PC, web i mobilní telefony. Je vyvíjen pod licencí Open Source. Zdrojový kód je dostupný pod licencí CDDL (Common

Development and Distribution License). Hlavním sponzorem a podporující firmou tohoto projektu je Sun Microsystems a dnes se vyvíjí dva produkty a to vývojové prostředí NetBeans a vývojová platforma The NetBeans Platform.

NetBeans IDE - je nástroj, ve kterém programátoři mohou psát, překládat i ladit libovolné aplikace. Existuje velké množství modulů, rozšiřujících toto vývojové prostředí o nové vlastnosti a možnosti.

NetBeans Platform - tato platforma je modulární, rozšiřitelný základ používaný při vytváření rozsáhlých aplikací. Modulární znamená, že dodavatelé softwaru nabízejí do této platformy moduly, které slouží pro vývoj jejich vlastních řešení a nástrojů.

Toto vývojové prostředí je dnes velmi oblíbené v komerční i nekomerční sféře. Je vyvíjeno celou komunitou vývojářů a má přes 100 firemních partnerů po celém světě. Je tedy docela zajímavý fakt to, že toto prostředí pochází z české republiky, proto si dále stručně povíme něco o historii vývoje.

4.2 Historie

Zárodek programu NetBeans vznikl roku 1996 v České republice jako studentský projekt na Matematicko-fyzikální fakultě Univerzity Karlovy . Cílem tohoto projektu bylo vytvořit v Javě vývojové IDE prostředí jako bylo v té době Delphi ale pro Javu. Tento cíl studenti splnili a výsledné vývojové prostředí nazvali Xelfi. To se stalo prvním vývojovým prostředím vytvořeným v Javě.

Studenty, kteří Xelfi napsali, tento projekt zaujal natolik, že se po dokončení studií rozhodli udělat ze svého výtvoru komerční produkt, většina z nich je v projektu NetBeans angažována dodnes. Postupem času se kolem Xelfi vytvořil pracovní tým lidí, který kontaktoval Roman Staněk hledající projekt hodný pro jeho investice. Společně vytvořili plán pro vývoj síťových JavaBeans komponentů. Poté se objevil i návrh změny názvu na NetBeans, aby jméno lépe vystihovalo cíle projektu. Tuto změnu inicializoval Jarda Tulach (tvůrce architektury a základů IDE). Na jaře roku 1999 vyšel NetBeans DeveloperX2 podporující rozhraní swing, který byl již kvalitním a životaschopným nástrojem. Poté se objevila Java 1.3 a tým předělal NetBeans do podoby, ve které ho známe dodnes.

V roce 1999 se také poohlížela firma Sun Microsystem po lepším vývojovém prostředí pro vývoj Java aplikací, které by se stalo vlajkovou lodí pro vývoj Java aplikací, a zajímala se též o NetBeans. Nakonec se pro něj i rozhodla a na podzim roku 1999 NetBeans koupila a přejmenovala ho na Forté for Java, čímž se název na krátký čas vytratil.

Po půl roce se společnost rozhodla uvolnit Forté jako OpenSource, který byl sponzorován pouze firmou Sun Microsystem a přejmenovat ho zpět na NetBeans. K tomuto staronovému NetBeansu byly zřízeny domovské stránky projektu a tento stav přetrvává dodnes, pouze se změnami verzí podle vývoje projektu. Aktuální verze je 6.0.

4.3 Charakteristické rysy

Základním rysem tohoto prostředí je jeho multiplatformnost. Již dříve jsme si řekli, že NetBeans běží na více platformách, což znamená, že je vytvořen pro více operačních systémů i typů počítačů. Znamená to, že se dá stáhnout ve verzích pro linux, windows, Mac OS, Solaris (spark, x86/x64) i v zip souboru nezávislém na OS. Pro každou z těchto platforem se dá stáhnout v několika verzích podle toho co uživatel požaduje. Verze jsou následující: Web & JavaEE, Mobility, Java SE Ruby, C/C++, All.

Každá z těchto verzí obsahuje vždy základní IDE moduly podporující jazyky, pro které je verze určena. Pouze verze All obsahuje moduly všechny, je i rozšířena oproti verzi Mobility o průvodce pro Mobility Web Services Connection. U NetBeansu je jedno kterou verzi si stáhnete, jelikož se chybějící moduly dají doinstalovat jako Pluginy (v menu Tools/Plugins záložka Updates).

Instalace je na každém systému obdobná, já jsem instaloval NetBean na počítačích IBM s operačním systémem Linux a Windows. U obou je to stejné, musíte si stáhnout JDK ze stránek firmy Sun (http://java.sun.com/javase/downloads) a nainstalovat, či nainstalovat z balíčkovacího systému v Linuxu a poté (jelikož se vás instalace bude dotazovat na složku s Javou) nainstalovat NetBeas, který lze také stáhnout z internetu (http://download.netbeans.org/netbeans/6.0/final/), nebo opět nainstalovat z balíčkovacího systému (podle verze Linuxu).

Důležitou součástí která se vám bude hodit je dokumentace. Její instalace není složitá a provádí se následovně. Nejprve si stáhnete (ze stejného umístění jako samotné JDK) zip soubor s dokumentací (název se odvíjí od verze Javy např. pro verzi 6 jeto

soubor jdk-6-doc.zip). Poté v NetBeans z menu Tools/Java Platforns vyberete v levé části platformu, pro kterou máte dokumentaci, a v pravé části dáte záložku Java Doc, kde tlačítkem Add Zip/Folder… otevřete okno pro výběr a následné přidání zip souboru s dokumentací.

Další typickou vlastností je zabudovaný grafický designer pro tvorbu grafického uživatelského rozhraní (GUI), který umožňuje tvorbu grafického rozhraní pouhým nataháním komponent na formulář. Je zde možnost jednotlivé komponenty ukotvit k okrajům formuláře či navzájem mezi sebou. V kódu se objeví odkazy na jednotlivé komponenty a můžeme pouze dopsat metody.

Dnes je již samozřejmostí zabudovaný debugger, který umožňuje krokovat, zastavovat a ladit aplikace. Bez tohoto šikovného nástroje se dnes neobejde žádný programátor.

Dalšími velmi využívanými funkcemi jsou automatické doplňování kódu při zadání klávesové zkratky „Ctrl+Space“, formátování nepřehledného kódu na krásně zformátovanou zkratku „Alt+Shift+F“nebo používání zkratek v kódu a jejich doplnění za kompletní výraz. Přehled těchto zkratek najdete v menu Tools/Options – Editor – záložka Code Templates, kde je kompletní seznam zkratek i nastavení kláves pro jejich doplnění.

NetBeans obsahuje také spoustu dalších doplňků jako je XML editor, možnost UML modelování nebo integrovaná podpora kontroly verzí. Kdybychom chtěli vyjmenovat všechny vlastnosti tohoto vývojového prostředí musel bych tuto práci zaměřit pouze na popis NetBeansu. Zde jsem uvedl jen několik základních vlastností a funkcí. Dále se podrobně podíváme na podporu vývoje aplikací pro mobilní zařízení.

4.4 NetBeans Mobility

Z minulého povídaní vyplývá, že pokud chcete, aby vaše instalace NetBeans podporoval tvorbu aplikací pro mobilní zařízení, musíte mít verzi Mobility/All a nebo si modul Mobility doinstalovat. Pokud máte nainstalováno, tak vaše vývojové prostředí podporuje tvorbu aplikací pro mobilní zařízení a vy můžete začít tvořit.

Pokud se rozhodnete vyvíjet aplikaci pro CDC zařízení, čeká na vás nemilé překvapení, jelikož podpora není úplná a to v oblasti emulátorů platforem. To znamená, že zde nejsou zabudovány žádné defaultní platformy pro emulaci CDC zařízení a bez doinstalování, vás nenechá průvodce vytvořením aplikace vytvořit aplikaci novou pro CDC. Tyto platformy přidáte do NetBeans tak, že si je stáhnete a nainstalujete na svůj počítač, poté v menu Tools/Java Platforms dáte Add platform, kde vyberete typ platformy a dále už NetBeans sám nalezne všechny platformy tohoto typu na vašem počítači a dá vám vybrat, kterou z nich chcete přidat do vývojového prostředí.

Tyto emulátory jsou volně ke stažení na stránkách www.netbeans.org, kde jsou i popsány postupy instalací s odkazy na stáhnutí. Velká nevýhoda tohoto procesu doinstalování je fakt, že všechny emulátory jsou v *.exe souborech, neboli na Linuxu si toho moc nedoinstalujete. Dokonce i Java Toolkit 1.0 for CDC přímo od firmy Sun Microsystems je pouze v .exe souboru, přestože ostatní součásti Javy jsou ve verzích pro více platforem.

Pokud se rozhodnete vyvíjet aplikaci pro CLDC zařízení neboli MIDlet, můžete se do toho bez starostí pustit, jelikož NetBeans obsahuje svou platformu Sun Java Wireless Toolkit 2.5.2

pro CLDC, ve které jsou na výběr hned čtyři zařízení (barevný display, černobílý display, qwerty klávesnice a kontrolní skin). Můžete tedy bez obav začít vývoj aplikace.

Pokud byste si chtěli doinstalovat platformu pro jiné zařízení např. kvůli specifickým vlastnostem, postup je stejný jako u CDC platforem. Bohužel i zde platí, že většina emulátorů jsou jako *.exe soubor.

Jelikož výuková aplikace, kterou budeme společně tvořit v poslední části práce bude určena pro ty nejmobilnější zařízení, neboli mobilní telefony, bude tedy tvořena pro zařízení CLDC. Proto si zde podrobněji probereme možnosti tvorby aplikací pro CLDC zařízení v NetBeans.

4.4.1NetBeans a jednoduchý MiIDlet

Pokud chceme vytvořit aplikaci na mobilní zařízení musíme vytvořit jeden či více MIDletů, ve kterých se budou nacházet třídy, metody atd.. Vytvoření jednoduchého MIDletu je velice snadné pomocí průvodce. V menu klikněte na File/New Project. Poté se spustí průvodce pro jeho vytvoření, kde vlevo vyberete Mobility a vpravo MIDP. Během tohoto průvodce lze zvolit mimo jména a složky pro uložení i konfiguraci a profil, pro které bude aplikace vytvořena. Po vytvoření tohoto prázdného projektu do něj máte možnost přidávat velké množství objektů. Množství těchto objektů je veliké od základních balíčků, klasických Java tříd přes MIDlety a Visual MIDlety, až po propracované pomocníky jako jsou Visual Game Designery a Java ME web klienti. Ty nejdůležitější

předdefinované objekty si následně probereme v dalších kapitolkách.

4.4.2 Visual MIDlet

Visuální návrhář pro MIDlety je novinka verze 6.0. Rozdíl mezi obyčejným a Visual MIDletem je ten, že obyčejný musíte napsat, napozicovat a ošetřit celý sami. Neboli každou věc kterou bude MIDlet obsahovat musíte vlastnoručně napsat. V dnešní době je v módě visuální části aplikace tvořit designéry, kteří po vašem naklikání objektů napíšou kód za vás. Tohoto trendu si byli tvůrci NetBeansu vědomi a vytvořili designer i pro návrh MIDletů.

Pokud vytvoříte nový Visual MIDlet, objeví se vám obrazovka skládající se ze čtyř záložek a to Source (Kód), Screen (Obrazovka), Flow (Diagramové znázornění aplikace) a Analyser.

Source – v této záložce se edituje přímo programový kód. Je vidět, že je zde již vytvořena základní kostra MIDletu s již naplněnými základními metodami pro běh MIDletu. Jsou zde také části s šedivým podkladem, které se nedají editovat, jelikož jsou vytvořeny přímo designerem. U každé této části (metodě) se ale nachází řádek před a po příkazu definovaným designerem pro váš kód.

Screen – v této části lze editovat jednotlivé obrazovky. Můžete přidávat jednotlivé prvky a měnit jejich vlastnosti podobně jako v MS Visual studiu. Například titulek formuláře nebo jeho název nastavený zde je ta část kódu, která má šedivé pozadí a nelze přímo editovat. Mimo jednotlivých objektů lze přidávat i příkazy. Je

zde i přehled přidělených příkazů. Jednotlivé obrazovky přepínáte pomocí popup menu.

Flow – tato část je srdcem designeru, jelikož zde je ten pravý prostor pro snadné vytvoření aplikace. Prostor je tu doslova, vy do tohoto prostoru pouze nataháte jednotlivé prvky jako jsou formuláře či Listy reprezentující seznam příkazů(menu příkazů na displayi a ne dole u ovládacích tlačítek). Poté stejným způsobem přidáte těmto jednotlivým prvkům příkazy(opět pouhým přetáhnutím) a ta nejkrásnější věc je skutečnost, že poté stačí tahem myši z příkazu na jiný prvek přiřadíte kam se po jeho vykonání dostanete. Tímto způsobem si zde základní kostru vaší aplikace naklikáte za pár minut.

Analyzer – zde NetBeans hlídá zda nemáte v aplikaci nepoužívané příkazy či jiné komplikace. Je to šikovná součást vývojového prostředí která vám ušetří zbytečné prohledávání kódu a hledání zbytečností.

4.4.3Mobile Game Builder

Žhavou novinkou ve verzi 6.0 je visuální pomocník pro tvorbu her pro zařízení. Pokud do projektu přidáte Visual Game design... otevře se vám přímo návrhář. V tomto objektu jsou záložky pouze dvě a to s kódem a s Buildrem. V druhé záložce je základní nabídka, kde jsou na výběr vytvořené objekty rozdělené do tří části, které jsou Sceny, Tiled Layers (vrstvy pokryté čtverečkovanou sítí, kde je jednotlivé čtverečky mají svou texturu) a Sprites (Animované objekty).

Tiled Layer – při editaci takovéto vrstvy, vytváříme mapu, neboli poklad scény. Při vytváření této scény si vyberete jaké má mít rozměry v pixelech. Pak vyberete obrázek s texturami, NetBeans obsahuje defaultně dva obrázky, které se skládají z velkého množství částí s různými texturami. Tento obrázek se po načtení rozdělí na jednotlivé části s texturami (tráva, cesta, voda, okraj). Vytvořená vrstva se také rozdělí na jednotlivé části a poté stačí pouze natahat části obrázků s texturami do částí ve vrstvě.

Sprite – pokud chcete vytvořit objekt, který se bude pohybovat, použijete tento objekt. Při jeho vytváření opět zadáte jméno, velikost v pixelech a obrázek. Tento obrázek může být shodný s obrázkem pro Tiled Layer. Po jeho vytvoření se vám objeví opět obrázek rozdělený na jednotlivé části a jednoduchý editor pro vytváření pohybujících se objektů. Tento editor pracuje na bázi filmové pásky, neboli máte k dispozici jednotlivá políčka do kterých vkládáte jednotlivé obrázky. Sami si určujete počet snímků, pořadí (máte možnost vložit i snímek mezi jiné dva bez posouvání ostatních) i čas pro přechod snímků. Po i při jeho vytvoření si lze výsledný vzhled animace prohlížet v zabudovaném přehrávači bez volání zbytečných funkcí.

Scene – v této části tvoříte výslednou scénu, do které lze vkládat a různě kombinovat jak vrstvy, tak naanimované objekty. Vrstvy lze vkládat přes sebe a určovat jejich pořadí.

4.4.4 Webové aplikace

MIDlety mohou také komunikovat s aplikacemi umístěnými mimo vaše zařízení,. Pro tuto možnost obsahuje NetBeans dva objekty a to Mobile Client to Web Application a Java ME Web Service Client.

Mobile Client to Web Application – tento objekt je klientem pro webovou aplikaci. Při vytváření zadáte jen jméno, balík, typ aplikace a služby použité pro připojení. NetBeans vygeneruje následující objekty: třída klienta v Java ME, servlet a pomocné třídy, soubor xml obsahující mapování, ukázkový MIDlet.

Java ME Web Service Client – aplikace obsahující tento objekt bude ke svému běhu používat webové služby neboli aplikaci poskytující nějaké služby. Každá tato aplikace je popsána WSDL souborem obsahujícím popis služeb. Při vytváření tohoto objektu musíte zadat adresu WSDL souboru. IDE si tento soubor stáhne a podle něj nastaví zbylé atributy a klient je hotov.

4.4.5 Blok kódu preprocesoru

U Javy existuje vlastnost, která umožňuje napsat část kódu, který bude použit pouze za určitých podmínek. Proč to tedy uvádím když je to vlastnost jazyka? Protože NetBeans má velkou podporu této vlastnosti a při psaní aplikací pro mobilní zařízení ji lze velmi efektivně využít. Tato vlastnost funguje na principu podmíněného příkazu `if /else ve` tvaru `//#if, //#else`

a `//#ednif.` Pokud bude splněna podmínka provede se část kódu za podmínkou, pokud ne provede se druhá část.

Nyní si asi říkáte proč to dělat tak složitě, když by se dalo použít klasicky příkaz `if`, jenže u naší možnosti, která se nazývá blok kódu procesoru, můžeme dát jako podmínku konfiguraci. Funguje to tak, že když je konfigurace aktivní provede se jiná část kódu, než když není.

V NetBeans takovýto blok kódu lze vytvořit tak, že kliknete do kódu na požadované místo pravým tlačítkem myši a zvolíte Preprocessor Blocks/Create Nested If/Else Block. Jako názorný příklad si můžete vyzkoušet na jednoduchém MIDletu, kde bude pouze jedna obrazovka a na ní nějaký nápis (`StringItem`) . Stačí najet do kódu, kde se vytváří tento objekt s určitým textem a vytvořit blok kódu. Jako podmínku dáte konfiguraci projektu, která je např. pro barevný telefon, a v první části bloku vytvoříte StringItem s nápisem „Barevný display“ a v druhé „Černobílý display“. Poté spustíte projekt s konfigurací pro barevný telefon, přehodíte konfiguraci na jinou pro černobílý telefon a znovu spustíte uvidíte na každém telefonu jiný nápis.

Pokud nemáte vytvořeno více konfigurací pro jeden projekt, lze to vytvořit v položce Properties v menu po kliknutí pravým tlačítkem na projekt. Zde můžete kromě Defaultní konfigurace přidat i jinou konfiguraci pro jinou platformu i zařízení a lze si vybrat, které součásti Javy ME bude mít tato konfigurace projektu k dispozici. Konfigurace je nastavení projektu, nesmíte to zaměňovat s konfigurací CLDC a CDC, pro aplikaci určenou pro mobilní telefon by tuto konfiguraci ani nebylo dost dobře možné změnit. Po přidání lze konfigurace snadno měnit v roletce na horní liště NetBeansu.

5 Popis vývoje ukázkové aplikace

Tato kapitola byla psána za pomoci [19] až [26].

Do teď jsme se zabývali pouze teorií o tom co je m-Learning, Java ME a NetBeans. Nyní konečně přichází chvíle, kdy to vše využijeme pro praktickou ukázku. V této poslední části práci si vytvoříme plnohodnotnou m-Learningovou aplikaci pro mobilní telefon. Tato aplikace bude psána v jazyce Java 2 Micro Edition a bude to aplikace pro zařízení CLDC v profilu MIDP . Téma této naší aplikace bude autoškola a příprava na závěrečné testy. Než se ale pustíme do samotného psaní kódu, musíme si rozmyslet velké množství věcí o tom, pro koho je určena její funkce, obsah a také se zaměříme na správný návrh, který bude vycházet z objektově orientovaného přístupu a jeho správného využití.

5.1 Problematika výuky v autoškole

Nejdříve si musíme upřesnit jak probíhá výuka v samotné autoškole, abychom mohli určit jak jí nejefektivněji doplnit naším programem. Když nastoupí člověk do autoškoly, projde si minimálně 6ti týdenním kurzem výuky, kde se dozví teorii předpisů, bezpečné jízdy, řešení dopravních situací, značek, údržby vozidla a první pomoci. V druhé půlce tohoto kurzu, když se žák dozví jak se správně v provozu chovat, začne absolvovat povinné hodiny přípravných jízd v určitém rozsahu podle požadované skupiny řidičského oprávnění. Po absolvování této přípravy v autoškole žák přistoupí k závěrečným testům a jízdám, kde

předvede, že je způsobilí k získání řidičského oprávnění. Výuka teorie se při tom skládá z hodin, kde se proberou jednotlivé části zákonů o provozu na pozemních komunikacích. Pro zjištění, zda žáci porozuměli probírané látce slouží soubor několika set otázek, které do hloubky prověří znalosti z dané problematiky.

Tento soubor obsahuje dohromady necelí tisíc otázek ze všech oblastí, s tím že ne každá otázka je určena pro všechny skupiny řidičského oprávnění (např. na otázku o odstředivé síle v zatáčce působící na motocyklistu, se nebude ptát žáka ucházejícího se o řidičské oprávnění na automobil). Otázky lze rozdělit do sedmi skupin a to: pravidla provozu na pozemních komunikacích, dopravní značky, dopravní situace, zásady bezpečné jízdy, předpisy o podmínkách provozu vozidel, předpisy související s provozem a zdravotnická příprava. Otázky jsou testového (výběrového) typu s dvěma nebo třemi odpovědi a každá otázka má právě jednu správnou odpověď.

Problematika závěrečného testu je dost složitá, jelikož pro každou skupinu řidičského oprávnění obsahuje v určitých částech otázky jiného zaměření. Touto problematikou se budeme ještě podrobněji zabývat později, nyní si řekneme jen základní informace. Všechny testy mají společné to že mají 25 otázek, za kterých lze získat max. 50 bodů, ale na úspěšné vypracování stačí 85 %(43 bodů), čas na vypracování je 30 min. Tyto otázky v testu jsou vybrány ze všech sedmi okruhů, pro každý okruh je určitý počet otázek a každá oblast je jinak bodově ohodnocena.

V dnešní době se testy nepíší v papírové formě jako tomu bylo dřív, ale dělají se online na stránkách ministerstva dopravy, kde se přihlásíte pod rodným číslem. Na těchto stránkách

se aplikace připojí do databáze otázek a test se vygeneruje podle dříve zmíněných pravidel. To znamená, že je test pokaždé jiný a pravděpodobnost že narazíte na stejný test dvakrát je mizivá. Z toho vyplývá, že se nemůžete dostat nikde k předtištěným testům komisařů a naučit se jen omezené množství otázek vyskytujících se v těchto několika předefinovaných testech. Systém také myslí na to, že by se vám mohla při některém zkušebním a ostrém testu vyskytnout stejná otázka a vy byste si mohli pamatovat pořadí správné odpovědi, proto zobrazuje otázky vždy s jiným pořadím odpovědí.

Co se týče zpětné vazby ve výuce teorie, přijde mi že přímo v autoškole je pomalu nulová. V každé autoškole se pouze vyloží teorie a pro zjištění, zda jste tomu porozuměli se pouze při několika posledních hodinách dělají zkušební testy, aby instruktor zjistil, jestli vás má pustit ke zkouškám nebo ne. Při výuce nezbývá čas na probírání významu či nejasností všech otázek, pouze pokud někdo sám narazí na nějakou nejasnost je mu vysvětlena. Z toho vyplývá že studium otázek je prováděno samostudiem.

5.1.1 Přístup k didaktickému materiálu

Přístup k didaktickému materiálu je jednoduchý, jelikož teorii vyučuje instruktor v autoškole a pokud zameškáte hodinu v učebnici naleznete vše co jste zameškali.

Pro přístup k otázkám pro procvičování existuje více zdrojů. Prvním je učebnice, kterou musíte mít pro každý kurz v autoškole (dnes většina učebnic přímo obsahuje otázky, nebo se kupují

zároveň s ní jako příloha). Tento zdroj ale obsahuje pouze předtištěné otázky, kde se nemění pořadí odpovědí a neexistuje rozumná cesta k náhodnému výběru (pouze listováním a vzpomínáním jestli jsem si dneska tu otázku již nečetl). Tento zdroj není nijak interaktivní a neumožňuje zpětný náhled na statistiku kolik odpovědí bylo dobře a kolik ne.

Další možnost nabízí stránky ministerstva dopravy, kde si mimo testů na ostro můžete také udělat test zkušební, anebo si pouze procvičit jednotlivé otázky podle kategorií s náhodným výběrem nebo postupně. Je zde i možnost výběru ze všech otázek, ale tato možnost mi přijde nepraktická, jelikož když jsem si jistý, že umím všechny otázky, zkusím si cvičný test a ne jednotlivé otázky. Tato možnost je interaktivnější, ale neukládá vaše výsledky, musíte si výsledky někam zapisovat, což také není nejideálnější.

Jiná možnost je také na internetu, jsou to stránky e-testy.cz, zde jsou možnosti pro zkoušení podobné jako na stránkách ministerstva. Výhodou je, že si zde můžete založit účet a budou vám tu i vedeny statistiky s tím co vám šlo a co vám nešlo a na co by jste se měli zaměřit. Zde pouze chybí rozdělení otázek týkajících se bezpečné jízdy do skupin podle požadované skupiny oprávnění.

Jistě existuje i velké množství jiných možností skrze různá komerční softwarová řešení obsahující celé studijní kurzy. Dosavadní zdroje byli nekomerční a zcela zdarma.

5.2 Analýza

V předchozí části jsme si vysvětlili průběh vyučování v autoškole, nyní tyto informace musíme analyzovat a určit jak tuto výuku vhodně doplnit. Pokud bychom chtěli udělat kompletní kurz provázející celou výuku až po testy, tak by to nebyl problém po stránce didaktické, jelikož je tato problematika zpracovaná ve velkém množství učebnic a stačilo by problematiku pouze didakticky transformovat do naší aplikace. Problém nastane když si uvědomíte, že bychom se snažili nasoukat celou učebnici o cca 300 stranách textu do zařízení s malou pamětí.

Je sice pravda, že dnes prodávané telefony mají parametry, které by obsloužili i náročné aplikace. Ale druhá stránka věci je odpověď na otázku: jaké telefony používají lidé, kteří absolvují kurz autoškoly (cílová skupina uživatelů)? Tato skupina jsou většinou lidé od 15ti do 25ti let. Podle průzkumu, který proběhl na naší universitě, mladí lidé používají většinou úplně obyčejné telefony staršího data výroby pouze pro základní funkce jako je telefonování a psaní SMS. Tomuto faktu budeme naší aplikaci muset přizpůsobit, proto jako ukázkovou aplikaci nebudeme dělat celý kurz, ale pouze jeho část.

5.2.1 Skladba výuky v autoškole

Po tomto rozhodnutí si musíme rozmyslet, jakou část výuky bude vhodné podpořit. Podle předchozích informací se výuka skládá ze tří částí a to výuka teorie, u které je další část prozkoumání a procvičení všech otázek, a jízd. Výuku teorie a jízd

provádí instruktor, pouze na probírání způsobu podání a porozumění všech otázek již moc času nezbývá a je většinou prováděno formou samostudia.

Výuku teorie zprostředkovává vždy instruktor v autoškole, kterou rozloží podle svého uvážení do celého kurzu také za pomoci vybrané učebnice.

Procvičování otázek probíhá z velké většiny samostudiem, jelikož je to pouze opakování. Je ale zřejmé, že nestačí pouhé pochopení a zapamatování si teorie, jelikož žák nemusí pochopit zadání otázky. Z tohoto důvodu je potřebné projít si všechny otázky. Je to také důvod proč existuje tolik zdrojů, kde lze otázky procvičit.

Záludná je výuka jízdy, kde je pro výuku potřeba přímo dopravní prostředek. Zde nám jen stěží pomohou knížky nebo internetové stránky, jelikož ač byste si teorii a postupy při řízení vozidel četli kolikrát chcete, potřebujete se tzv. vyjezdit, což znamená zjistit jak se dané vozidlo chová v té či oné situaci. Z těchto důvodů tuto část výuky nelze doplnit jakýmkoli softwarem.

5.2.2 Vyhodnocení

Já osobně jsem absolvoval autoškolu třikrát (skupiny AM, B+A1, A) a nikdy jsem se doma neučil teorii, pouze jsem si ji procvičoval formou otázek, čímž jsem se i zároveň připravoval na testy. I z mého okolí absolvovalo velké množství kamarádů kurz autoškoly a každý z nich se připravoval na testy pouze procvičováním otázek a ne slepím čtením zákonů, jelikož otázky jsou vztažené již ke konkrétní situaci, která může nastat.

Po zhodnocení předchozích informací je zřejmé, že jedinou částí, která je zapotřebí doplnit, je procvičování otázek. Z tohoto důvodu budeme tvořit aplikaci zaměřenou na procvičování otázek a ne na výklad teorie či výuku ovládání vozidel.

5.3 Návrh funkcí aplikace

Nyní již víme na jakou problematiku bude zaměřena naše aplikace, musíme si tedy ujasnit, které funkce bude obsahovat. Inspiraci lze nalézt ve stávajících zdrojích a aplikacích. Již zde bylo zmíněno, který zdroj má jaké výhody a nevýhody. V naší aplikaci bychom měli spojit výhody více zdrojů a vytvořit komplexní aplikaci. Funkce podporované dosavadními zdroji by se dali rozdělit do tří základních skupin – procvičování otázek, cvičné testy a statistiky pro zpětnou vazbu.

Toto rozdělení vyplývá z didaktiky, protože žák se nejdříve procvičováním naučí správně odpovídat na otázky, zde jako zpětná vazba působí statistiky o výsledcích, a poté co usoudí, že je připraven, si zkusí cvičné testy. Jelikož naše aplikace nepoběží na internetu jako dosavadní zdroje přidáme si i čtvrtou skupinu funkcí a to multimediálních, jelikož chceme udělat aplikaci také zábavnou. V další části probereme jednotlivé funkce a specifikujeme si jejich rozsah.

5.3.1Procvičování otázek

Pokud si chceme procvičovat otázky nemůžeme je uživateli dát všechny na ráz. Otázky by se měli rozdělit do skupin, nejpříhodnější bude otázky rozdělit do skupin podle 7 základních

kategorií. To znamená je rozdělit do skupin týkajících se pravidel provozu na pozemních komunikacích, dopravní značky, dopravní situace, zásady bezpečné jízdy, předpisy o podmínkách provozu vozidel, předpisy související s provozem, zdravotnická příprava. Tímto rozdělením by jsme ale končit neměli, jelikož si musíme uvědomit, že v autoškolách se nevyučuje pouze pro budoucí řidiče aut, ale všech skupin řidičského oprávnění. Pro každé toto oprávnění existují specifické otázky. Tyto specifické otázky se vyskytují ve třech skupinách a to předpisy souvisejících s provozem na pozemních komunikacích, předpisy o podmínkách provozu vozidel na pozemních komunikacích, zásady bezpečné jízdy a ovládání vozidla.

Pokud bychom chtěli rozdělit tyto kategorie podle skupiny řidičského oprávnění museli bychom k prozatímním skupinám přičíst dalších pět. Toto rozdělení budeme muset někde v zařízení zobrazit, pravděpodobně v nějakém seznamu, ale ten by měl na malé displaye moc položek a byl by velmi nepřehledný, proto bychom ho měli malinko zredukovat. Když se zamyslíme nad významem specifických otázek, mají velký význam pouze v kategorii týkající se bezpečné jízdy. Ostatní dvě kategorie se rozdělují pouze kvůli několika málo teoretickým otázkám nevalného významu. Otázky ohledně bezpečné jízdy jsou rozdílné pro skupiny A,B a C+D. Výsledné rozdělení by mělo vypadat následovně - pravidla provozu na pozemních komunikacích, dopravní značky, dopravní situace, zásady bezpečné jízdy A, zásady bezpečné jízdy B,zásady bezpečné jízdy C+D, předpisy o podmínkách provozu vozidel, předpisy související s provozem, zdravotnická příprava.

Další věcí, nad kterou je vhodné se zamyslet je, jak bude zkoušení probíhat. První záležitostí je otázka, zda by se zkoušení mělo provádět v určitém počtu otázek. Na tuto otázku je odpověď jasná: rozhodně ne! Jelikož pro procvičování otázek na mobilu uživatel bude využívat každé volné chvíle, které nemají nikdy jistou dobu trvání, měl by mít možnost ukončit procvičování kdykoli. Jednou z posledních vlastností, kterou by měla disponovat naše aplikace, je možnost vybrat si zda se budou otázky vybírat postupně či náhodně.

Naposledy bychom se měli zaměřit na funkce didaktické, kde se nám nabízí dvě. První je zpětná vazba, kterou zde těžko uskutečníme vazbou na teorii, jelikož jí naše aplikace obsahovat nebude. Jako zpětná vazba by nám zde mohlo posloužit statistické zobrazování výsledků jednotlivých zkoušení, protože si poměr správně zodpovězených ku všem otázkám bude uživatel jen stěží pamatovat. Zpevnění je užitečná didaktická funkce pomáhající uživateli v sobě informaci upevnit. Takovéto zpevnění by mělo probíhat po každá otázce.

5.3.2 Cvičné testy

O tom, zda funkci cvičné testy zařadit, nemůže být pochyb. Procvičení testů je nejvyšší meta v autoškole, jelikož po nich většinou přicházejí testy ostré. Zařazení testů, by neměl být problém, když aplikace bude otázky již obsahovat pro procvičování, tak z nich test pouze poskládáme.

Další rozhodnutí musíme udělat v otázce, pro které skupiny řidičského oprávnění testy tvořit. Je sice pravda že největší část tvoří skupina B, pak A, a zbytek tvoří jen mizivé procento,

ale pokud naše aplikace obsahuje specifické otázky, neexistuje jediný důvod proč je také nevytvořit. Pouze se bude muset ošetřit výskyt specifických otázek.

Jelikož se pohybujeme v prostředí malého displaye mobilního telefonu, bude test probíhat podobně jako procvičování otázek. Bude se postupně zobrazovat jedna otázka po druhé a pro přehlednost bez možnosti vrácení k předchozí otázce. Toto omezení je zavedeno z důvodu, že vracení se přes několik otázek (displayů) k „některé“ otázce, kde si uživatel nebyl jistý, je poněkud zmatené. Také je dokázáno, že se udělá více chyb při kontrole, než při samotném vyplňování.

Co se týče vyhodnocení, to bude složitější. Jedna věc je sdělit uživateli zda prospěl či nikoliv, ale druhá bude vyřešit zpětnou vazbu. Pokud bychom chtěli ukázat uživateli test se všemi otázkami (25 displayů) a označit zda byla zodpovězena dobře a v případě, že nebyla, označit správnou odpověď, by se uživatel do prohlížení velmi snadno zamotal. Jinou možností je pouze sdělit prospěch a dále se tím nezabývat, ale to by jsme přišli o jakoukoli zpětnou vazbu. Jako kompromis by se dalo zvolit sdělení, ve kterém bude procentuální prospěch v jednotlivých skupinách otázek. Toto bychom měli zvolit, jelikož je to možnost přehledná a přitom se uživatel dozví, které otázky mu nešli a měl by si je znovu procvičit.

5.3.3 Statistika a zpětná vazba

Jak jsem již dříve uvedl, statistika by se měla zobrazit vždy, když se uživatel rozhodne ukončit zkoušení. Co s těmito daty posléze? Určitě by to neměla být jediná statistika v aplikaci, tyto

údaje by se měli někde střádat a komletovat, aby měl uživatel možnost se před zkoušením ujistit, která kategorie otázek mu dělá největší problémy. To znamená, že by statistika měla obsahovat procentuální úspěšnost pro každou kategorii otázek.

Do těchto statistik by se měli ukládat také údaje z testů, jelikož i v testu se vyskytují otázky z jednotlivých kategorií. Statistika by také měla obsahovat i údaje o prospěšnosti v testech. Jelikož testů uživatel nebude dělat tolik jako samotných otázek, neměli by jsme uvádět procentuální úspěšnost, ale přímo počty provedených a úspěšně absolvovaných testů. Tyto údaje již nemusíme dále dělit podle typu testu, jelikož si uživatel bude zkoušet pouze jeden typ a ten určený pro svou skupinu řidičské oprávnění.

5.3.4 Multimediální funkce

Multimediální funkce existují u mobilních zařízení pouze dvě a jsou to vibrace a zvuk. Jak jsme si již řekli dříve, existují dvě skupiny knihoven, jedny jsou defaultně v základu Javy, a druhé jsou dodávané výrobci telefonů a narušují přenositelnost programu. Třídy podporující práci s vibracemi jsou ve skupině druhé, pokud bychom je chtěli použít, naše aplikace by nefungovala na více typech mobilních zařízení, nesmíme je tedy používat. Třídy podporující práci se zvukem jsou obsaženy v základní Javě, proto je v aplikaci použijeme.

5.3.5 Otázky a dopravní značení

Než začneme vytvářet aplikaci musíme si ještě ujasnit zdroje didaktických dat. Je sice pravda, že zde bylo řečeno, že jsou v každé učebnici autoškoly, ale při hlubším prozkoumání narazíte na problém se specifickými otázkami. Většina učebnic je totiž pouze pro skupiny A,B a není v nich ani rozdělení otázek do těchto kategorií. Dlouze jsem zkoumal různé knihy a internetové zdroje, ale ani zde se mi nepodařilo vyzkoumat specifičnost jednotlivých otázek, jelikož některé jsou pro všechny skupiny a jiné jsou pro jednu či více. Rozhodl jsem se proto kontaktovat pana Ing. Ondřeje Horázného provozovatele stejnojmenné pražské autoškoly a hlavně internetových stránek e-testy.cz . Pan Horázný byl tak ochotný, že za příslib výsledné verze programu k distribuci mi vyexportoval databázi otázek z e-testů i s rozdělením pro jednotlivé skupiny.

Dalším problémem jsou dopravní značky. Naše aplikace bude obsahovat obrázky několika dopravních značek, jenže je zde problém s licencí. Všechny dopravní značky na celém světě jsou chráněny licencí, kterou já nevlastním, proto se zde budou dále používat pouze ilustrační obrázky, pro demonstraci funkce programu. Ve výsledné verzi aplikace budou obsaženy obrázky skutečných značek, jelikož bude šířena pod licencí pana Horázného z Prahy.

5.4 Návrh programu

V předchozí části jsme si ujasnili, které funkce by naše aplikace měla obsahovat. Nemůžeme ale začít hned programovat. Proces vývoje programu není jednoduchá záležitost a skládá se z více částí. Tento proces je podobný procesu vývoje didaktického kurzu, jelikož se musí nejdříve analyzovat prostředí, následně vytvořit prvotní návrh programu, poté jeho realizace a při vyvstání problému se struktura poněkud upraví, naposledy se program testuje a dolaďují nepřesnosti v chodu. Nejobecnější rozdělení lze shrnout do tří částí - návrh, realizace a testování.

Návrh se vytváří z mnoha důvodů, ten hlavní je, že kvalitní návrh ušetří hodně programátorské práce. Co to znamená? Je to logické, jelikož při globálním pohledu na program při vytváření návrhu počítáte se všemi funkcemi aplikace, ale když začnete psát program z jedné strany tak na druhé straně zjistíte, že jste s něčím nepočítali nebo jste něco zapomněli a musíte dodělávat a opravovat. Naším úkolem tedy nyní bude vytvořit takovýto návrh.

5.4.1 Přehled funkcí

Abychom si mohli ujasnit strukturu programu, uvedeme si nejdříve globální přehled funkcí:

Cvičné procvičování otázek v kategoriích:

1. pravidla provozu na pozemních komunikacích
2. dopravní značky
3. dopravní situace

4. zásady bezpečné jízdy A
5. zásady bezpečné jízdy B
6. zásady bezpečné jízdy C+D
7. předpisy o podmínkách provozu vozidel
8. předpisy související s provozem
9. zdravotnická příprava

Zkoušení bude probíhat otázku po otázce. Po každé otázce se objeví vyhodnocení se zpevněním. Bude zde možnost ukončit procvičování jak ze zobrazené otázky, tak z vyhodnocení. Po ukončení se zobrazí statistika s počtem otázek a procentuální úspěšností. Otázky se budou vybírat v náhodném pořadí, ale bude zde možnost nastavit generování postupně a zase zpět.

Cvičné testy v kategoriích:

1. A
2. B
3. B+E
4. C
5. C+E
6. D
7. D+E

Zde bude zkoušení probíhat otázku po otázce, po každé otázce se nebude ukazovat vyhodnocení (jako při ostrých testech), ukáže se až při ukončení testu. Test bude omezen časem 30 min. Bude možno test ukončit kdykoli během celého testu, nebo se ukončí sám při vypršení časového limitu. Odpočet času bude ošetřen pro případ přerušení (příchozí hovor), aby se nepočítal čas během přerušení. Vyhodnocení bude probíhat při ukončení testu,

kde se vypíše prospěch (splnil\nesplnil), bodové skóre a pro zpětnou vazbu i procentuální úspěšnost v jednotlivých skupinách otázek.

Statistiky - budou obsahovat procentuální úspěšnost z jednotlivých skupin otázek + úspěšnost v testech, ale přímo v číslech všech a úspěšně složených testů, jelikož počty absolvovaných testů budou mnohonásobně menší než jednotlivých otázek. Data budou uložena v zařízení a budou trvalá tzn., že přetrvají přes vypnutí a zapnutí aplikace.

Zvuk – zvuk bude možno zapnout či vypnout. Toto nastavení bude opět trvalé, to znamená že, pokud uživatel zvuk vypne, bude vypnutý i při dalším spuštění.

Program budeme muset doplnit také o funkce zobrazující pravidla používání nejlépe rozvětvené do dvou sekcí (procvičování otázek a testy). Aplikace by také měla obsahovat informace o autorovi a verzi programu.

5.4.2 Struktura aplikace

Nyní známe přesně všechny funkce, musí z těchto poznatků složit určitou kostru programu, jak se bude chovat vůči uživateli. Poté se pustíme do definice objektů, jejich metod a řešení jednotlivých problémů.

Základem uživatelského rozhraní bude seznam základních příkazů, kde bude rozdělení na: procvičování otázek, cvičné testy, statistika, pravidla používání, nastavení, about, konec.

Procvičování otázek – odkáže uživatele na další seznam, kde si vybere z jaké kategorie chce procvičovat. Také bude obsahovat příkaz pro návrat do hlavního menu. Každý ze seznamu kategorií odstartuje proces procvičování, který se bude skládat ze dvou obrazovek pro otázku a její vyhodnocení. Na obrazovce vyhodnocení se zobrazí text v závislosti na úspěšnosti odpovědí. Z obou obrazovek se bude moct dostat na obrazovku se statistikou (pouze ukončení zkoušení), po které se uživatel dostane zpět k seznamu kategorií.

Cvičné testy – budou velmi obdobné části pro procvičování. Odkáží na seznam s jednotlivými skupinami řidičského oprávnění + odkaz zpět na základní menu. Každý ze seznamu skupin odstartuje proces testu skládající se z jedné obrazovky pro jednotlivé otázky a obrazovky pro vyhodnocení, po které se uživatel dostane zpět k seznamu skupin.

Statistika – se bude skládat pouze z jedné obrazovky pro zobrazení statistických údajů + odkaz zpět na základní menu.

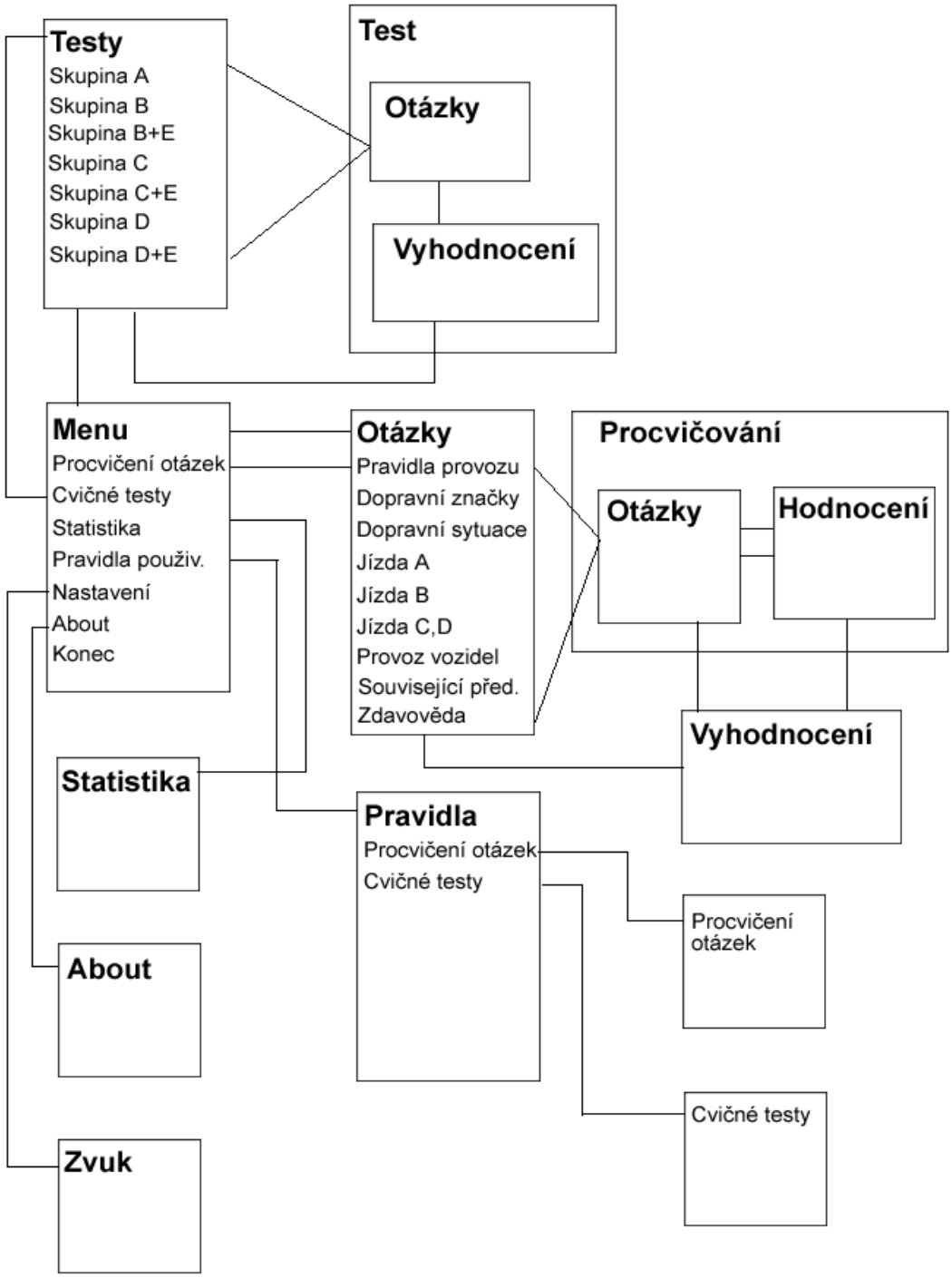
Pravidla používání – odkáže na seznam se dvěma položkami pro pravidla testů a procvičování otázek + odkaz zpět. Každá z položek zobrazí text týkající se pravidel používání té sekce + odkaz zpět.

Nastavení – zobrazí jednu obrazovku s možností nastavení aktivity zvuku + příkaz uložit, který nastavení uloží a vrátí uživatele zpět na menu.

About – zobrazí obrazovku s textem o verzi programu a informacemi o programu + odkaz zpět.

Konec – příkaz který ukončí aplikaci.

Tato struktura je popsána na následujícím obrázku:



Obrázek 10 : Struktura aplikace m-Testy

5.4.3 Návrh objektů

V této podkapitole si musíme rozhodnout jak využijeme OPP a navrhneme si jednotlivé objekty. Jasným a základním kamenem aplikace jsou jednotlivé otázky. Jedna otázka obsahuje text otázky, správnou odpověď, jednu nebo dvě špatné odpovědi, název obrázku a skupiny pro které je určena popřípadě kategorii do které patří. Proto vytvoříme objekt, který tyto informace ponese. Další důležitou vlastností otázky je pokaždé jiné pořadí odpovědí, jelikož s otázkami budeme pracovat na více místech (testy, procvičování), měli bychom kód zajišťující tuto funkci vytvořit na jediném místě kam budou přistupovat oba procesy - přímo v objektu otázka.

Otázky budeme muset někam uložit a odtud je načítat. Nabízí se také otázka jak je skladovat, zda všechny pohromadě v jednom velikém souboru nebo je nějak rozdělit. Když se podíváme zpět na funkce používající otázky, tak procvičování používá vždy pouze jednu kategorii a test sice pracuje se všemi ale také podle jednotlivých kategorií. Nabízí se tedy možnost, rozdělit si otázky do více souborů podle kategorií a nenačítat do omezené paměti telefonu celí objem dat. Tuto možnost při vývoji uplatníme.

Kolem otázek bude také hodně kódu zajišťující jejich načtení, uložení do nějaké kolekce (pro snadný přístup) a samotný přístup. Proto bychom měli vytvořit další objekt, který bude obsahovat metody, které to zajistí. Nazvěme ho třeba otázky.

Když se nám povede udělat objekt otázky s dostatečnou funkčností, bude nám to stačit pro celý proces zkoušení. Pro co nám to ale stačit nebude, jsou cvičné testy. Pokud by jsme tam chtěli přidat ještě metody a funkce pro vytváření a práci s testy,

bylo by to obsáhlé, složité a nepřehledné. Proto si vytvoříme ještě jeden objekt obsahující tyto funkce a nazveme ho Testy.

Další záležitostí, kterou naše aplikace bude muset obstarat bude ukládání, načítání a úpravu hodnot týkajících se statistiky. Další velmi podobnou funkcí bude ukládání a načítání jednoho údaje, který nám ukáže zda je zvuk zapnutý, nebo vypnutý. Vytvoříme proto další dva objekty, které budou obsahovat tuto funkčnost, nazveme je Statistika a Zvuk.

Nyní máme navržené pomocné objekty disponující metodami pro jednotlivé funkce programu. Chybí nám vytvořit samotné tělo programu. Toto tělo bude obsaženo v jedné třídě, která bude potomkem třídy MIDlet. Tato třída bude obsahovat grafické prvky a příkazy pro jejich přepínání. Až ho vytvoříme, bude stačit do akcí jednotlivých příkazů doplnit funkce, které budou obsluhovat metody z námi již nadefinovaných objektů.

5.4.4 Uložení dat

Z popsaných funkcí je jasné, že budeme potřebovat pracovat s velkým počtem otázek, které budeme muset někde „skladovat“. Jaké jsou možnosti uložení dat v MIDP zařízeních?

V MIDP zařízeních nám Java dává jedinou možnost, jak ukládat nějaká data. Tuto možnost nabízí balíček `javax.microedition.rms`, který obsahuje třída a metody pro ukládání dat databázovým způsobem. Takto uložená data ale nemusí být trvalá, záleží sice na implementaci Javy v mobilním telefonu , ale jsou to většinou data, která se vymažou při vyndání baterie ze zařízení. Z tohoto důvodu se takto ukládají pouze data

nedůležitá pro běh aplikace, jako jsou například nejvyšší skóre ve hrách, nastavení zvuků atd..

Java nám sice nenabízí jak data trvale uložit (např. do souboru), ale nabízí nám možnost je načítat a to také ze souboru, proto existují dohromady tři možnosti jak texty s otázkami uložit.

První možností je vytvořit si třídu obsahující texty těchto otázek. Možnost je to sice nejednoduší, ale nejeví se mi moc profesionální, zejména proto že kód programu slouží k určitým funkcím a ne k uchovávání velkého objemu dat. Jen si představte tu nepřehlednost kódu s tolika otázkami a odpověďmi.

Další možností je uložit si texty do souboru jako text například jako csv formát. V tomto souboru by na jednom řádku mohla být jedna celá otázka s jednotlivými údaji oddělenými středníkem. Tato možnost už nám oddělí data od kódu. Jak to bude s načítáním? Museli by jsme vytvořit algoritmus, který načte data za pomoci tříd `InputStream` a `InputStreamReader` a pak porovná jednotlivé znaky („;“ ,“ \n“), podle kterých by určil co je otázka a co jsou odpovědi. Tento algoritmus by byl dost složitý, jelikož by bylo hodně údajů na řádce musel by se provádět pokaždé, když by se pracovalo s otázkami. A to by zbytečně zatěžovalo zařízení.

Další možností je uložení do binárního souboru. Do tohoto souboru bychom postupně uložili text otázky a jednotlivé odpovědi a další potřebné údaje všech otázek postupně za sebou. Při načítání bychom pak v tom samém pořadí tyto údaje pouze načetli zpět. Tento způsob je podle mého soudu nejednodušší a nejefektivnější, proto jsem se rozhodl ho ve své práci použít.

V naší aplikaci se budou otázky již pouze načítat. Pro uložení otázek jsem si vytvořil pomocnou aplikaci. Kde jsem si vytvořil třídu `Otázka` obsahující proměnné s otázkou, odpověďmi a názvem

obrázku. Poté jsem vytvořil jednotlivé otázky jako instance této třídy a naplnil jimi hashovací tabulku. **V** programu budeme používat podobné třídy jako jsem použil zde, ale k jejich návrhu se dostaneme v další části. Algoritmus pro uložení do datového souboru vydá následovně:

```
public void ulozOtazky() {
        FileOutputStream fos = null;
        DataOutputStream dos = null;
        int p;
        try {
        // vytvoření výstupního proudu do souboru
 fos = new FileOutputStream("NazevKategorieOtazek.bin");
          // vytvoření datového výstupního proudu,
          // který data posílá dál do
          // souborového výstupního proudu
             dos = new DataOutputStream(fos);
             Enumeration enumer = otazky.keys();
            // zapsání počtu otázek
             p = otazky.size();
             dos.writeInt(p);
             while (enumer.hasMoreElements()) {
             // uložení klíče a pak jednotlivých hodnoty textotázek
             //jako text
           String key = (String) enumer.nextElement();
                 dos.writeUTF(key);
                 Otazka ot = (Otazka) otazky.get(key);
                 dos.writeByte(ot.getSkupina());
                 dos.writeUTF((String) ot.getObrazek());
                 dos.writeUTF((String) ot.getOtazka());
       dos.writeUTF((String) ot.getSpravnaOdpoved());
       dos.writeUTF((String) ot.getSpatneOdpovedi()[0]);
       dos.writeUTF((String) ot.getSpatneOdpovedi()[1]);
             }
             } catch  (Exception e) {
             e.printStackTrace();
        } finally {
             if (dos != null) {
                 // zavření datového výstupního proudu
                 try {
                     dos.close();
```

```
                } catch (Exception e) {
                }
            }
            if (fos != null) {
                // zavření souborového výstupního proudu
                try {
                    fos.close();
                } catch (Exception e) {
                }
            }
        }
    }
```

5.4.5 Třída Otázka

Tato třída bude reprezentovat otázku v testu a ponese informace o ní. Informace jsou následující: text otázky, správná odpověď, jednu nebo dvě špatné odpovědi, název obrázku a skupiny pro které je určena popřípadě kategorii do které patří. Vytvoříme proto obyčejnou Java třídu, která bude obsahovat instanční proměnné `otazka,spravnaOdpoved,image` typu `String`, další bude `skupina` typu `byte` a naposled `spatneOdpovedi`, která bude typu pole řetězců `String[]`. Tyto proměnné budou soukromé, aby k nim nemohl přistupovat nikdo bez oprávnění.

Konstruktor bude proto mít tyto všechny hodnoty jako parametry a bude obsahovat přiřazení jejich hodnoty do instančních proměnných (např. `this.image = image;`). Zde se vyskytne jedna maličkost, jelikož jsem ukládal všechny ze stejného formuláře, otázky obsahující pouze jednu špatnou odpověď mají v poli špatných odpovědí na druhém místě hodnotu prázdného stringu „“, která by nám dělala problémy při zobrazování otázky na display. Tento nedostatek vyřešíme tak, že vytvoříme nové pole `String[]` o délce 1 a do něj přiřadíme první hodnotu v poli

špatných odpovědí předaného jako parametr konstruktoru. konstrukce bude vypadat následovně:

```
if(spatneOdpovedi[1].equals(""))
   {
       String q = spatneOdpovedi[0];
       this.spatneOdpovedi = new String[1];
       this.spatneOdpovedi[0]= q ;
   }else{
   this.spatneOdpovedi = spatneOdpovedi;}
```

Dále jsme si říkali, že do této třídy umístíme metodu zajišťující přeházení pořadí odpovědí. Budeme tady muset udělat ze všech tří odpovědí jedno pole odpovědí, náhodně je tam přiřadit a uložit si index správné odpovědi . Tuto náhodnost nám obstará instance třídy Random z balíčku java.util.Random se svojí metodou nextInt (int mez), která nám vrátí náhodně vygenerované číslo od nuly včetně až po hodnotu meze předané v parametru.

Jelikož tuto instanci třídy `Random` budou používat všechny instance stejně, deklarujeme jí jako třídní (statickou) a samozřejmě opět soukromou. Pole otázek které vytvoří tato metoda musíme předat někam, kde se bude zobrazovat, proto bude metoda toto pole vracet jako návratovou hodnotu. Dále si musíme někam uložit hodnotu indexu správné odpovědi, proto pro ní vytvoříme instanční proměnou `index` typu `int`.

Algoritmus bude probíhat tak, že vytvoří nové pole řetězců jehož velikost je odvozena od počtu špatných odpovědí, vloží do něj na náhodnou pozici správnou odpověď a uloží si tuto pozici. Poté jen spustí cyklus, které vloží spatné odpovědi na zbývající místa poli. Konstrukce vypadá následovně:

```
public String[] prehazOdpovedi()
 {   String[] odpovedi;
     int p = spatneOdpovedi.length+1;
     odpovedi = new String [p];
     this.index = rnd.nextInt(p);
     odpovedi[index] = spravnaOdpoved;
     for(int e = p-2;e>-1;)
     {       int i=rnd.nextInt(p);
              if (odpovedi[i]==null)
               { odpovedi[i]=spatneOdpovedi[e];
                    e--;
               }
     }
     return odpovedi;
 }
```

Jako poslední stačí vytvořit přístupové metody (get.....) pro zbývající proměnné image, skupina, otázka a skupina.

5.4.6 Třída Zvuk

Jak bude fungovat nastavování zvuku? Jednoduše, celé nastavení zvuku bude postaveno na jedné hodnotě typu `boolean`, která ukazuje, zda je zvuk zapnutý nebo ne. Podle této hodnoty určí aplikace, jestli se má zvuk přehrát. Jak bylo řečeno v popisu funkcí bude tato hodnota trvalá, bude se tedy ukládat do zařízení, aby přetrvala přes vypnutí a zapnutí aplikace. Třída zvuk bude tedy uchovávat informaci, zda je zvuk zapnut. Měla by také být schopna tuto hodnotu změnit, uložit a načíst.

Třída tedy bude obsahovat jednu instanční proměnou typu boolean. Dále by měla obsahovat veřejnou přístupovou a nastavovací metodu pro přístup ostatních tříd, čili metody `void setZvuk(boolean zvuk)` a `boolean getZvuk()`. Jako další

bude obsahovat soukromé metody pro načtení a uložení proměnné.

Aby se nemusela při několika násobném přístupu k proměnné `zvuk` bez její změny vždy hodnota načítat, provedeme metodu provádějící načtení již v konstruktoru a hodnotu uložíme do instanční proměnné. Další načítání se provádět nemusí, jelikož při změně hodnoty změníme hodnotu v instanční proměnné a také jí uložíme. Tímto způsobem máme aktuální hodnu jak uloženou v zařízení, tak i instanční proměnné a nemusíme nic načítat.

Ukládání hodnoty bude probíhat pomocí balíčku `javax.microedition.rms`, kde využijeme třídu `RecordStore`. Tuto třídu si můžeme představit jako databázovou tabulku, do které pomocí metody `addRecord` přidáváme jednotivé řádky. Vkládat tyto „řádky“ lze pouze jako pole bytů, proto má tato metoda tři parametry. První jsou data reprezentovaná polem bytů, druhý parametr udává počáteční byte a poslední vstupní parametr určuje délku oblasti.

Pro získání odkazu na `RecordStore` musíme použít statickou metodu za stejné třídy `openRecordStore`, která má dva parametry. První typu `String` určující jméno `RecordStoru`, který se má otevřít a druhý typu `boolen`, který udává, zda se při nenalezení tohoto `RecordStore` má vytvořit nový či neprovádět nic.

Pro zapisování musíme vytvořit `ByteArrayOutputStream` a `DataOutputStream`, u kterého dáme jako parametr konstruktoru právě vytvořený `ByteArrayOutputStream`. Samotné zapsání probíhá tak, že potřebnou hodnotu zapíšeme do `DataOutputStream`, poté získáme data jako pole bytů metodou `toByteArray()` z `ByteArrayOutputStream`. Tyto data pak vložíme do `RecordStore` metodou `add`.

Musíme si zde pouze ošetřit rozdíl mezi nově vytvořeným (pokud ještě nebyl vytvořen, nebo byl smazán) a již existujícím `RecodrStore`. Tento rozdíl poznáme tak, že nově vytvořený neobsahuje žádná data a splní podmínku `zvukStore.getNumRecords() == 0`. Do nově vytvořeného `RecodrStore` budeme data vkládat metodou `addRecord`, ale v existujícím je budeme měnit metodou `setRecord` s indexem 1, jelikož jsme vkládali pouze jeden záznam a číslování začíná záznamem na pozici č. 1.

V této proceduře musíme zachytit výjimky jako v celku a také ještě při samotném zapisování do `DataOutputStream`. Co se týče načítání to probíhá obdobným způsobem, pouze v opačném pořadí. Tuto problematiku jsem zde zmínil trochu podrobněji abychom snadněji porozuměli procedůrám v dalším objektu.

5.4.7 Třída Statistika

Třída Statistika bude mít podobnou funkci jako Zvuk. Rozdíl bude v tom, že bude záznamů uchovávat 10 a v každém záznamu nebude jedna hodnota ale hned dvě. Deset záznamů je z toho důvodu, že se bude uchovávat úspěšnost pro všechny kategorie otázek, které máme rozděleny do 9ti skupin. Poslední záznam bude uchovávat informace o úspěšnosti v testech (uživatel bude dělat vždy pouze jeden typ testu). V každém záznamu bude údaj, udávající z kolika otázek byl uživatel z určité skupiny dohromady zkoušen a druhý bude obsahovat počet správně zodpovězených otázek, aby se dala snadno a přesně spočítat procentuální úspěšnost.

Abychom nezaměnili pořadí záznamů uložíme si indexy jednotlivých záznamů do statických konstant (`private static final int PRAVIDLA_PROVOZU_INDEX = 1, …`). Na rozdíl od třídy zvuk zde nemůžeme přímo při zapisování ověřit zda není `RecordStore` prázdný a zaměnit metodu vložení dat, protože by se data nemuseli vložit na námi požadovanou pozici. Z tohoto důvodu provedeme toto ověření přímo v konstruktoru třídy, pokud bude prázdný zavoláme metodu, která nám ho naplní startujícími daty, samými nulami, což nám zajistí že budeme moc data vkládat na požadovanou pozici.

Další metoda bude zajišťovat změnu určitého údaje. Pokud budeme potřebovat přidat do statistik, že uživatel byl zkoušen z 12ti otázek z pravidel provozu a na deset odpověděl správně, metoda načte oba údaje ze záznamu 1 a ke každému přičte danou hodnotu a uloží je zpět do záznamu. Tato metoda by měla být univerzální pro všechny záznamy, proto jí v parametrech předáme index záznamu který chceme změnit a oba údaje které chceme přičíst. Metodu nazvěme `napln`.

Jak ale zajistíme abychom nezaměnili pořadí záznamů? Uděláme tuto metodu jako soukromou a vytvoříme sadu veřejných metod `setPravidlaProvozu`, `setZnacky` atd. s dvěma parametry reprezentující počty otázek, které vždy zavolají metodu `napln`, vždy s dalším parametrem udávajícím index záznamu, který chceme změnit. Tento třetí parametr bude reprezentován námi nadefinovanými statickými konstantami.

Poslední funkcí, kterou by tato třída měla obsahovat, bude vracení všech uložených záznamů pro zobrazení celkové statistiky v zařízení. Tuto metodu nazveme `getStatistiky` a bude nám vracet pole hodnot typu `int`, ve kterém budou uloženy jednotlivé

záznamy. Do pole nebudeme ukládat jednotlivé počty otázek, ale již přímo spočítanou procentuální úspěšnost, kterou spočítáme tak, že počet správně zodpovězených otázek či splněných testů vynásobíme stem a vydělíme celkovým počtem.

Proto si na začátku metody vytvoříme pole typu `int` o délce 10, do kterého uložíme postupně načtené a spočítané údaje. Toto pole na konci metody vrátíme příkazem `return`.

5.4.8 Třída Otazky

Třída `Otazky` bude zabezpečovat následující dvě funkce – načtení otázek z externího binárního souboru do nějaké kolekce a předávání jednotlivých otázek buď v náhodně generovaném, nebo postupném pořadí. Tyto otázky samozřejmě musí odpovídat požadované skupině řidičského oprávnění. Do jaké kolekce otázky ukládat? odpověď je velmi jednoduchá a to do hashovací tabulky (`Hashtable`), protože v MIDP profilu žádná jiná kolekce není obsažena. V souboru je jako první obsažen údaj o počtu otázek a dále je zde vždy pořadové číslo otázky a jednotlivá data otázky.

Načítání otázek bude probíhat pomocí `InputStream` a `DataInputStream`, ze kterého budeme metodami `readInt()`, `readByte()`, `readUTF()` načítat jednotlivá data. Nejdříve načteme údaj o počtu uložených otázek, který využijeme v následujícím cyklu načítající jednotlivé otázky pro udání počtu opakování. Data jednotlivých otázek se budou načítat v pořadí: pořadové číslo, skupina, obrázek, otázka, správná odpověď a dvě špatné odpovědi. všechny tyto hodnoty uložíme do lokálních proměnných a posléze je použijeme pro vytvoření otázky, kterou vložíme do tabulky. Jako klíč použijeme pořadové číslo otázky. Klíč je uložen jako text proto

se načítá metodou `readUTF()` a ne `readInt()`, jak by jste čekali. Klíč je v tomto formátu z důvodu, že jako klíč v tabulce nesmíme použít číselný typ, proto nemusíme nic přetypovávat a rovnou můžeme vložit otázku do tabulky. Na začátku metody nesmíme zapomenout smazat všechny otázky z kolekce kvůli předchozímu možnému naplnění (předchozí zkoušení otázek jiné či stejné kategorie). Tato metoda by měla být opět univerzální pro všechny kategorie otázek, proto název souboru bude jako parametr.

Dále budeme muset otázky předávat zpět hlavní třídě a to buď v náhodném pořadí nebo postupně. Pro potřeby této funkce budeme potřebovat instanční proměnnou typu `boolean` pro určení v jakém pořadí se budou otázky vracet. Pro postupné předávání bude potřeba proměnná typu `int` pro ukládání pozice naposledy předané otázky. Pro náhodné generování budeme potřebovat jako instanční proměnnou instanci třídy `Random`, která obsahuje metody pro náhodné generování čísel. Jako poslední proměnnou budeme potřebovat proměnnou typu `byte` pro uložení hodnoty skupiny pro kterou je třeba otázky předávat.

Samotné předávání bude probíhat, tak že se otestuje proměnná udávající pořadí generování otázek a poté se buď vygeneruje náhodné číslo z intervalu 0-počet otázek v kolekci nebo se použije proměnná udávající pořadí otázky v kolekci. Dále se vybere otázka z kolekce s požadovaným klíčem a otestuje se zda vyhovuje dané skupině otázek. Pokud odpovídá otázka se předá jako návratová hodnota. Pokud ne metoda se rekurzivně sama zavolá a jako návratovou hodnotu předá otázku vrácenou takto zavolanou metodou.

V této metodě musíme také ošetřit zda hodnota udávající pořadí otázky při postupném předávání nepřesáhla počet otázek

v kolekci a pokud ano, tak jí nastavit zpět na počáteční hodnotu, kterou je č. 1. Parametr nebude mít metoda žádný, jelikož není zapotřebí. Pro nastavení skupiny otázek a způsobu pořadí předávání otázek vytvoříme dvě veřejné metody set... .

5.4.9 Třída Testy

Než se pustíme do psaní třídy pro vytváření testů musíme si nejdříve vysvětlit pravidla podle kterých test vzniká a jak se hodnotí. V testu je obsaženo všech sedm kategorií otázek každá v jiném rozsahu a jinak bodově ohodnocená. V některých kategoriích se vyskytují i specifické otázky, které určují, pro kterou skupinu řidičského oprávnění je určen. Struktura testu je popsána v následující tabulce:

Kategorie	**Počet otázek**	**Počet bodů za jednu otázku**	**Výskyt specifických A,C,D,E otázek**
pravidla provozu na pozemních komunikacích a jejich užití řidičem	10	2	
znalost dopravních značek.....	3	1	
řešení dopravních situací	3	4	
zdravotnická příprava	1	1	
předpisy o podmínkách provozu vozidel na pozemních komunikacích	2	1	x
předpisy souvisejících s provozem na pozemních komunikacích	2	2	x
zásady bezpečné jízdy a ovládání vozidla	4	2	x

Z této tabulky je vidět, že všechny testy mají společný základ skládající se ze 17ti otázek ze čtyř kategorií a ve zbytku se liší výskytem specifických otázek.

Třída Testy se bude od třídy Otazky lišit v několika podstatných faktech. Prvním je, že nebudeme pracovat pouze s jednou kategorií otázek, ale hned se všemi najednou. A druhým důležitým aspektem je, že budeme muset někde uchovávat prospěch odpovědí, aby jsme na konci testu mohli vyhodnotit, zda uživatel prospěl či nikoliv.

Celý proces vytváření testu bude vypadat, tak že vždy načteme celou jednu kategorii otázek uložíme si jí do pomocné kolekce. Z této pomocné kolekce si náhodně vybereme určitý počet otázek vyhovující dané kategorii a uložíme si je do hlavní kolekce uchovávající otázky pro test. Poté pomocnou kolekci vyčistíme načteme do ní další kategorii a tak postupujeme do té doby než nenaplníme hlavní kolekci všemi 25ti otázkami.

Z předchozích informací vyplývá, že budeme potřebovat metodu která nám bude načítat otázky ze souboru. Tato metoda bude úplně stejná jako ve třídě `Otazky`. Můžete mi namítnout, že to je duplikace. Ano je to tak, ale třída `Otazky` nám vrací pouze jednu náhodnou odpověď a nemůžeme s ní pracovat tak, jak bychom potřebovali. Navíc se buď pracuje se třídou `Testy` a nebo `Otazky`, takže se nám jeden kód dvakrát neprovádí, provádí se vždy pouze jeden.

Další metoda nám z této pomocné kolekce vybere určitý počet otázek odpovídající požadované skupině. Tato metoda se tedy bude moct zavolat jak pro přidání jedné nebo více specifických otázek, tak pro přidání více nespecifických. To

znamená, že metoda bude mít dva parametry, jeden bude určovat počet přidávaných otázek a druhý odpovídající skupinu. V těle metod vytvoříme cyklus, kde bude počet opakování určený počtem přidávaných otázek. V těle cyklu vždy vygenerujeme výhodné číslo (stejným způsobem jako v minulé třídě) z rozmezí 0-počet otázek v pomocné kolekci a poté přidáme otázku do hlavní kolekce. Zní to jednoduše, ale má to háček. Může nastat situace, kdy se nám vygeneruje stejné číslo dvakrát. Proto si v metodě vytvoříme pole typu `int[]` kam si budeme ukládat vygenerovaná čísla otázek a před každým přidáním otázky do hlavní kolekce otestujeme, zda klíč otázky již není obsažen v tomto poli, pokud ano otázka se nepřidá. Nesmíme zapomenout na správné přičítání počtu opakování cyklu, protože počet opakování se musí změnit pouze v případě, že se otázka opravdu přidá, jinak by se nepřidalo požadované množství otázek.

Pro kontrolu, zda se prvek již v poli vyskytuje vytvoříme další soukromou metodu, která pouze projde pole předané v parametru a testuje, zda se jednotlivé prvky neshodují s prvkem, který je předán jako druhý parametr. Pokud se prvek v seznamu vyskytuje, metoda vrátí `true` a v opačném případě `false`.

Nyní máme vytvořeno vše potřebné a můžeme se pustit do vytváření testů. To bude probíhat tak, že se vždy v metodě pro vytváření testů zavolá metoda pro načtení určité kategorie otázek a následně metoda pro přidání určitého počtu otázek pro potřebnou kategorii. Tento děj se bude opakovat pro všechny kategorie. Budeme tedy muset vytvořit metody pro každý druh testu zvlášť, jelikož se v každém testu budou volat tyto metody s jinou skupinou, aby se zaručil výskyt specifických otázek. Tyto metody by ale měli společné volání prvních čtyř kategorií, proto si

tento kód napíšeme do metody `vytvorZakladTestu()`, která naplní hlavní kolekci otázkami z prvních čtyř kategorií a všechny metody vytvářející test budou na počátku volat tuto metodu.

Dále musíme vytvořit metodu, která bude opět vracet otázky jednu po druhé. Abychom věděli pořadí otázky, které máme předat musíme si vytvořit instanční proměnou typu `int` a do ní si pořadí ukládat.

Také musíme vytvořit metodu, která se nám bude starat o ukládání prospěchu. To bude probíhat tak, že si vytvoříme jako instanční proměnnou pole typu `boolen[]` a metodu `pridejProspech`, která přidá do pole na pozici, které odpovídá pořadí předané otázky, údaj zda uživatel odpověděl správně či nikoli. Tuto hodnotu metodě předá hlavní třída jako parametr.

Poslední metody vytvoříme typu get, které budou vracet celé pole s prospěchem (pro vyhodnocení testu) a pořadí předávané otázky, aby si mohla hlavní třída číslovat zobrazené otázky.

5.4.10 Hlavní třída MIDlet Mtesty

Nyní máme připravené veškeré potřebné pomocné objekty a můžeme se vrhnout na to pravé tvoření objektu. Hlavní třída bude potomkem třídy MIDlet a my využijeme možnosti NetBeansu a vytvoříme si Visual MIDlet, který nám ušetří spoustu práce.

Po vytvoření tohoto prázdného MIDletu se přepněte do záložky Flow. Zde si vytvoříme strukturu aplikace z pohledu jednotlivých seznamů příkazů, obrazovek a přepínání mezi nimi. Zde budeme používat dva základní prvky, a to `Form` a `List`.

Objekt `Form` je formulář přes celou obrazovku, který má svůj titulek. Do tohoto formu můžeme vkládat další prvky jako jsou

Choice Group, Image Item, String Item, Text Box , Command a mnoho dalších. My budeme v naší aplikaci využívat pouze String Item pro vypsání textů, Choice Group pro zobrazení otázky a Command pro příkazy.

U Choice Group existuje Label(popisek), do něj budeme vkládat text otázky, pak jsou Choice Elements, do kterých můžeme přidat kolik chceme, my budeme přidávat dvě nebo tři pro odpovědi. Dále nás u tohoto objektu budou zajímat dvě vlastnosti. První vlastnost Type, nastavuje kolik bude možné vybrat elementů buď jeden (EXCLUZIV), a nebo více(MULTIPLE). Je zde i možnost nastavit POPUP, ale to by se nám komponenta přeměnila ve vyjíždějící menu a to nechceme. Druhá vlastnost je index vybrané možnosti, ke které se přistupuje pomocí metody getSelectedIndex. Tuto vlastnost budeme využívat při vyhodnocování zda uživatel odpověděl správně.

Co se týče String Item, tak je to jedna z možností, jak přidat na obrazovku nějaký text, ale snazší je text do formuláře přidat přímo metodou append a jako parametr dát požadovaný text.

Posledními a nejdůležitějšími komponentami jsou Commands neboli příkazy. Pokud chceme do MIDletu přidat nějaký příkaz, musí tento MIDlet implementovat rozhraní CommandListener a obsahovat metodu commandAction, ve které jsou definovány akce jednotlivých příkazů. Toto vše za nás udělá NetBeans sám a my už pouze definujeme v kódu akce. Většinou slouží k přepnutí zobrazené obrazovky. Příkazy se přiřazují automaticky ovládacím tlačítkům zařízení, které jsou většinou dvě. Pokud přiřadíme formuláři více než dva příkazy, vytvoří se u jednoho tlačítka

položka menu, ve které se bude nacházet seznam příkazů. Pořadí příkazů určuje jejich priorita, což je jedna z vlastností příkazů.

Pokud chceme na jednu obrazovku přiřadit více příkazů a nechcete aby byli ve skrytém menu, ale přímo na obrazovce musíte použít komponentu List. Tato komponenta je seznam příkazů, které se zobrazí na displayi pod sebou (klasické menu známé z mobilních telefonů). Položek tohoto seznamu můžete přidat libovolné množství. Akce těchto příkazů jsou definovány v metodě s názvem „název komponenty+Action“, kterou za nás opět vytvoří Netbeans.

Ten za nás také vytvoří veškeré přístupové metody typu `get` ke všem komponentám, které do aplikace vložíme. Pokud chceme nějakému příkazu přiřadit, aby nám přepnul zobrazenou obrazovku, lze to udělat jednoduše a to tahem myši z příkazu na požadovanou obrazovku.

Tímto způsobem strukturu aplikace vytvoříme pomocí myši za několik okamžiků. Struktura bude shodná se schématem na posledním obrázku. První obrazovka bude List se základním menu aplikace, další tři Listy budou jako seznam kategorií otázek, seznam typů testů a seznam pravidel pro používání. Použijeme Třikrát i `Choice Group`, pro obě obrazovky zobrazující otázky, jak při zkoušení, tak při testech a výběr vypnout/zapnout zvuk, zbytek budou již pouze Formy s texty či otázkami. Musíme ve vlastnostech jednotlivých prvků vše řádně pojmenovat, aby kód aplikace byl přehledný.

Když máme vytvořeno můžeme se pustit do upravování kódu a definice akcí jednotlivých příkazů. Jako první si popíšeme zobrazení statistiky, protože je to velmi jednoduché. Stačí v kódu najít definici akce příkazu pro zobrazení statistiky (já si úvodní

seznam nazval `mTesty`, a proto je tato definice v metodě `mTestyAction`). Do definice přidáme kód ve kterém získáme odkaz na formulář, kde se má statistika zobrazit a jako první z něj vše smažeme (při znovu zobrazení statistiky by nám zde mohli zůstat data z minulého zobrazení), poté si vytvoříme novou instanci třídy `Statistika` a metodou `getStatistika` získáme pole s procentuální úspěšností pro jednotlivé kategorie otázek. Nyní stačí pouze do formuláře pro zobrazení statistiky metodou `append` přidat jednotlivé texty typu „Značky: “ a za tento text vložíme hodnotu ze získaného pole úspěšností odpovídající značkám.

Další jednoduchý případ je zobrazení nastavení a samotné nastavení zvuku. Než se ale pustíme do zobrazování, musíme nejdříve zjistit, jak je zvuk nastaven a to potřebujeme vědět hned při startu aplikace. Vytvoříme tedy instanční proměnou typu `Zvuk` a typu `boolen`, která ponese zda je zvuk zapnut. V konstruktoru MIDletu přiřadíme do proměnné `Zvuk` novou instanci třídy `Zvuk` a metodou `getZvuk` přiřadíme do proměnné zvuk hodnotu nastavení zvuku.

Při zobrazení nastavení bude práce odlišná pouze v tom, že budeme pracovat s `Choice Group`, která je na Formu pro zobrazení nastavení zvuku (texty na ní jsou nastaveny již z designeru, celý formulář je tedy již připraven). Opět stačí nalézt definici akce pro příkaz „Nastavení“. Zde dáme pouze podmínku, kde se ověří nastavení zvuku a podle něj se nastaví v `Choice Group` jako vybraná příslušná možnost.

Při ukládání napíšeme do akce příkazu pro uložení jednoduchý kód, který zjistí z `Choice Group`, zda je vybraná možnost pro zapnutí zvuku metodou `isSelected`, která vrací

hodnotu typu `boolean`, tuto hodnotu uložíme do proměnné zvuk a také ji metodou `setZvuk` ze třídy `Zvuk` uložíme do zařízení.

5.4.11 Proces procvičování otázek

Proces zkoušení otázek nebude vytvářet otázku na jednom místě, protože se bude skládat z několika částí. Nejdříve budeme muset načíst správnou kategorii otázek, pak jednotlivé otázky zobrazit vyhodnotit odpověď a při ukončení procvičování zobrazit procentuální úspěšnost a tyto výsledky také uložit do statistik. V celém procesu tedy budeme muset ukládat a jednotlivé výsledky, proto si vytvoříme dvě instanční proměnné typu `int` a do jedné budeme ukládat celkový počet otázek a do druhé počet správně zodpovězených.

Pro načtení otázek stačí vytvořit instanci třídy `Otazky`, kterou si uložíme do instanční proměnné pro následné několikanásobné volání metody pro předání další otázky, a v této instanci zavoláme metodu `nactiOtázky` pro naplnění kolekce otázkami. Toto provedeme v definici akce každého z příkazů pro začátek procvičování otázek určité kategorie, ale pokaždé s jiným názvem souboru.

Dále budeme muset otázku zobrazit na displayi, jelikož to budeme provádět často umístíme si tento kód do zvláštní metody. Tato metoda by měla zobrazovat otázku, jak pro proces procvičování, tak pro testy, ale přitom bude pracovat vždy s jiným formulářem a jinou `Choice Group`. Jak to tedy udělat? Řešení je jednoduché, vytvoříme si instanční proměnné typu `Form` a typu `Choice Group` a před začátkem zkoušení či testu si do nich vložíme ty správné objekty a metoda bude pracovat již pouze s těmito

proměnnými. Proto si vytvoříme další metodu která bude toto přiřazení provádět a bude se volat vždy před spuštěním procesu. Další problémem je to že při testu se otázky budou předávat ze třídy `Testy` a při zkoušení ze třídy `Otazky`. Toto vyřešíme tak, že si vytvoříme další instanční proměnou typu `boolean` a do ní uložíme zda probíhá proces procvičování a nebo testu.

Metoda na začátku ověří o který proces se jedná a podle toho načte příslušnou otázku a uloží jí do instanční proměnné typu `Otazka`, jelikož s ní budeme pracovat ještě při vyhodnocení. Pak smaže všechny elementy z obou proměnných a pustí se do práce s otázkou. Nejdříve musí ověřit zda k otázce nepatří nějaký obrázek. To provede podmínkou `otazkecka.getObrazek().equals("")`, pokud ano provede se jeho zobrazení.

Zobrazení obrázku lze rozdělit do několika částí. Nejdříve vytvoříme zdroj dat pomocí `InputStream`, kde jako parametr vložíme název obrázku. Pomocí tohoto zdroje vytvoříme obrázek jako instanci třídy `Image` pomocí třídní metody `createImega,` kde jako zdroj obrázku vložíme vytvořený `InputSream`. Abychom mohli vložit tento obrázek do Formuláře musíme vytvořit prvek formuláře `ImageItem` jako instanci této třídy, kde jsou parametry popisek, obrázek, nastavení pozice a alternativní text. Tento `ImageItem` poté přidá do formuláře metodou `append.`

Dále metoda pouze vloží text otázky, popis `ChoiceGroup` a odpovědi z pole, které vrátí metoda `prehazOdpovedi` z otázky, a přidá je jako elementy pro výběr stejnou metodou jako do formuláře. Jako poslední metoda zkontroluje, zda se provádí test, a když ne musí rozhodnout, který ze dvojice příkazů Postupně a Náhodně zobrazit na displayi.

Nyní máme otázku zobrazenu a musíme definovat její vyhodnocení, které se bude nacházet v akci příslušného příkazu. Toto vyhodnocení bude probíhat tak, že se zkontroluje, zda se shoduje index správné odpovědi z otázky s indexem vybrané možnosti v `ChoiceGroup` zobrazující otázku. Pokud ano zobrazí text o tom jak uživatel odpověděl na otázku správně a pro zpevnění zopakuje otázku a odpověď. Pokud ne sdělí to uživateli a zobrazí znovu otázku se špatnou i se správnou odpovědí.

Zde je taky ta část kódu, kde se bude využívat instanční proměnná reprezentující zapnutí zvuku, jelikož zde je prostor pro přehrání zvuku. Pokud je zvuk zapnutý přehraje se zde melodie vítězná či opačná. Pro toto přehrání vytvoříme samostatnou metodu, která bude mít parametr typu `boolean`, který bude reprezentovat, zda se má přehrát melodie pro úspěch nebo opačný případ. Pro samotné přehrání musíme vytvořit instanci třídy `Player` z balíčku `javax.microedition.media`, kde ne nacházejí funkce pro zacházení s mediálními funkcemi. Tuto instanci vytvoříme třídní metodou `createPlayer` ze třídy Manager, kde jako první parametr vložíme `InputSream` ze zvukového souboru a druhý parametr, který určuje jaký formát zvuku se bude přehrávat, bude „`audio/midi`“. Zvuk spustíme metodou start z instance třídy `Player`. Je zde i možnost nastavit počet opakování zvuku metodou `setLoopCount.` V metodě zobrazující otázku se tato metoda zavolá s `true` nebo `false` podle prospěchu, pokud se splní `podmíka` zapnutého zvuku.

Musíme také definovat události, které nastanou při ukončení procvičování. Ukončit procvičování můžeme ze dvou obrazovek, to znamená dvěma příkazy a proto si vytvoříme jednu metodu, kde bude akce definována a ta se bude volat v akci obou

z příkazů. V této metodě nebude nijak složitý kód, pouze z formuláře pro zobrazení této lokální statistiky se vymažou všechny elementy a poté se za pomoci textů vypíší údaje o proběhlém procvičování. Údajů pro vypsání je několik a to počet zodpovězených otázek, počet správně zodpovězených otázek a procentuální úspěšnost spočítaná z předchozích údajů.

Jako poslední musíme zařídit, aby se údaje o procvičování uložili do celkových statistik. To provedeme v akci a příkazu na potvrzení zobrazené lokální statistiky a přechod zpět na výběr kategorií otázek. Zde vytvoříme instanci třídy `Statistika` a podle kategorie procvičovaných otázek zavoláme metodu která údaje uloží.

5.4.12 Proces cvičných testů

Proces cvičného testu bude velmi podobný procesu procvičování otázek. Z předchozího procesu máme připraveno většinu funkcí, které budeme používat. Odlišnost je zde v omezeném počtu otázek, ve stylu vyhodnocování a v omezeném čase. S omezením počtu otázek nebude problém, jelikož ve třídě `Testy` máme připravenou metodu, které nám vrací hodnotu typu `boolean` reprezentující zda jsme nevyčerpali otázky. Při vyhodnocení se nevyhneme jinému procesu, jelikož jsou otázky rozdílně ohodnocené.

Pro čas si budeme muset vytvořit funkce, které nám to zajistí. Bude se jednat rovnou o dvě časové funkce a to ukončení testu při vypršení limitu a měnění zbytku času pro vypracování testu. Pokud chceme v MIDletu uskutečnit nějakou akci v určitém čase, slouží k tomu třídy `Timer` a `TimerTask`. Celý proces probíhá

tak, že si vytvoříme potomka třídy `TimerTask` a v ní metodu `run`, která bude provádět akci, kterou chceme provádět v určitém čase. Pak vytvoříme instanci třídy `Timer`, ze kterého budeme používat metodu `schedule`, která bude mít dva nebo tři parametry. Prvním parametrem je vždy nějaká instance třídy `TimerTask`(reprezentující akci, která se má provést), druhým je čas v milisekundách od spuštění kdy se akce má poprvé provést, pokud má metoda i třetí paramtr je to čas pauzy mezi jednotlivými opakováními akce.

Pro naše potřeby si tady vytvoříme dvě třídy poděděné od třídy Timertask nejlépe jako vnořené třídy. V jedné třídě umístíme do metody run kód, který nám spustí metodu pro ukončení testu, přepne zobrazený Form na Form pro zobrazení vyhodnocení testu a přidá do něj text, že byl test ukončen z důvodu vypršení časového limitu. V druhé třídě vložíme do metody jednoduchý kód, který odečte 1 z instanční proměnné reprezentující zbývající čas. K volání metody `schedule` se dostaneme později.

Než začneme probírat celý proces vytvoříme si ještě jednu metodu a to bude obdoba metody `nastavProcvicovani` ale pro testy. V této metodě se také jako v té pro procvičování přiřadí správné hodnoty do proměnných reprezentujících `Form` a `ChoiceGroup` pro testy. Do této metody umístíme ještě nastavení hodnoty 30 do proměnné reprezentující počet zbývajících minut, vytvoření instancí tříd Timer a TimerTask a dvoje volání metody `schedule`. Poprvé jí budeme volat s instancí pro ukončení testu a časem 1800000(30min). Podruhé s instancí pro změnu hodnoty zbývajícího času a dvojím časem 60000(1min) jednou pro první provedení a podruhé pro každé následující.

Celý proces testu začne v akcích příkazů pro jednotlivé testy. Zde se pokaždé zavolá metoda `nastavTest`, nastaví se hodnota proměnné, která bude určovat do jaké skupiny bezpečné jízdy se má uložit statistika, vytvoří se instance třídy `Testy` a zavolá se metoda pro vytvoření příslušného testu. Poté se již jen zavolá metoda pro zobrazení otázky, čímž se odstartuje proces testu.

Metodu pro zobrazení otázky budeme muset trochu upřesnit. Odlišnost v zobrazené otázce bude spočívat v tom, že nad otázkou v testu bude zobrazeno její pořadí a zbývající čas. To znamená, že do připravené podmínky zda se jedná o test pro načtení otázky ze správné třídy, vložíme přidání tohoto textu do formuláře. Údaj o pořadí otázky nám předá metoda `getIndex` ze třídy Testy. K tomuto údaji musíme ještě přičíst 1, jelikož hodnota indexu začíná na č. 0 ale otázky na č. 1. Mimo tohoto doplnění bude zobrazení otázky shodné.

Nyní musíme vyhodnotit zda byla otázka odpovězena správně. To provedeme v akci příkazu pro zobrazení další otázky, test správnosti odpovědi proběhne stejným způsobem jako u procvičování. Pokud bude zodpovězena správně zavoláme metodu `pridejProspech` ze třídy Testy s `true` a pokud ne tak s `false`. Dále musíme zjistit zda nejsme na konci testu, pokud ano ukončíme test a zobrazíme formulář s výsledky a pokud ne zobrazíme další otázku. Co se týče ukončení testu, to proběhne buď po zodpovězení všech otázek, vypršení časového limitu nebo ukončení testu uživatelem v průběhu testu. Z tohoto důvodu si proceduru ukončení testu vložíme do samostatné metody.

Metoda pro ukončení testu bude muset vyhodnotit, zobrazit a uložit výsledky. V jejím těle Získáme nejdříve pole s prospěchem, poté ho projdeme ale ne najednou, jelikož každá kategorie otázek

je jinak bodově ohodnocena a také musíme každou kategorii zvlášť uložit do statistik. Proto si na začátku vytvoříme lokální proměnnou typu `int[]` pro ukládání procentuálních úspěšností jednotlivých kategorií. Dále vytvoříme proměnné typu `int` pro uložení celkového počtu získaných bodů, pozice v poli úspěšností, pozice v poli prospěchů a pomocnou pro ukládání počtu správně zodpovězených otázek z jednotlivých kategorií. Poté projdeme pole prospěchů po jednotlivých kategoriích, spočítáme správně zodpovězené otázky, uložíme čísla do statistik, spočítáme počet získaných bodů, které přičteme k celkovému skóre, a spočítáme procentuální úspěšnost, kterou vložíme na příslušnou pozici v poli úspěšností. Na konci metody porovnáme zda uživatel uspěl či nikoli a pokud je zapnutý zvuk, přehrajeme příslušnou melodii. Na konec vypíšeme texty informující uživatele o celkovém prospěchu a procentuálním prospěchu v jednotlivých kategoriích.

Nyní jsme prošli a vysvětlili všechny funkce naší výukové aplikace.

6 Závěr

Účelem mé bakalářské práce bylo popsat a vysvětlit problematiku M-Learningu, popsat programovací jazyk Java ME, ukázat vývojové prostředí NetBeans a vše využít při vývoji ukázkové aplikace.

Myslím, že jsem tento účel splnil, jelikož v první kapitole jsme si vysvětlili co to m-Learning je, popsali jsme si jeho výhody, nevýhody i princip vývoje kompletního M-Kurzu s didaktickými pravidly.

Ve druhé kapitole jsme si popsali vlastnosti programovacího jazyka Java ME, strukturu a rozdělení knihoven pro jednotlivá zařízení.

Ve třetí kapitole jsme si ukázali možnosti vývojového prostředí NetBeans a jeho využití při vývoji aplikací pro mobilní zařízení.

Ve čtvrté a nejdůležitější části práce jsme vytvořili a popsali vývoj krok po kroku plnohodnotné m-Learningové aplikace, která nalezne v praxi své využití.

Já osobně jsem v myšlence m-Learningu našel velké zalíbení, jelikož možnost studovat ve frontě na úřadě a mít volné odpoledne pro své záliby je velmi lákavá. Je ale fakt, že většina lidí mobilní zařízení zatím pro tyto účely nevyžívá, což je velká škoda. Tato skutečnost byla jedním z důvodů proč jsem si vybral toto téma.

Ve své práci jsem se snažil osvětlit možnosti využití mobilních zařízení pro výuku a využití volných chvil. Jelikož jsme během práce probrali vše potřebné k pochopení principů m-Learningu a ukázali jsme si jak lehké je vytvořit plnohodnotnou výukovou aplikaci pro mobilní zařízení, mohla by tato práce

přispět k rozšíření vývoje podobných aplikací a omezení zabíjení času hraním nudných a nezáživných her na mobilních telefonech.

Úplným závěrem bych chtěl říci, že M-learning je šikovná myšlenka, ale otázkou zůstává zda se rozvine a uspěje v dnešním pohodlném světě a doufám, že má práce přispěla pouze k jejímu rozvoji.

Reference

[1]KUBÍK, Martin. *Vývoj mobilních telefonů : 1. díl* [online]. 2006 [cit. 2008-02-02]. Dostupný z WWW: <http://www.galaxie.name/index.php?clanek=vyvoj-mobilnich-telefonu-1-dil>.

[2]KNĚŽÍK, Martin. Telefony, které psaly historii českého mobilního trhu : část první. *Http://clanky.katalogmobilu.cz/* [online]. 2006 [cit. 2008-02-02]. Dostupný z WWW: <http://clanky.katalogmobilu.cz/zajimavosti/348-telefony-ktere-psaly-historii-ceskeho-mobilniho-trhu/>.

[3**]**TÁBORSKÝ, Jan. První mobilní telefon - Motorola DynaTAC 8000X. *Http://vseohw.net* [online]. 2006, roč. 1. [cit. 2008-02-02]. Dostupný z WWW: <http://vseohw.net/clanky/historie/prvni-mobilni-telefon-motorola-dynatac-8000x>.

[4]ROSMMAN, Pavel. M-LEARNING – NOVÉ PARADIGMA VZDĚLÁVÁNÍ POMOCÍ ICT. *Everest.natur.cuni.cz/konference/2007/prezentace/* [online]. 2007 [cit. 2008-02-10]. Dostupný z WWW: <everest.natur.cuni.cz/konference/2007/prezentace/rosman.ppt>.

[5] ROSMMAN, Pavel. M-LEARNING – NOVÉ PARADIGMA VZDĚLÁVÁNÍ POMOCÍ ICT. *Http://everest.natur.cuni.cz/konference/2007/prispevekv* [online]. 2007, roč. 2007 [cit. 2008-02-10], s. 5. Dostupný z WWW: <http://everest.natur.cuni.cz/konference/2007/prispevek/rosman.pdf>.

[6]*PAVLÍČEK, Jiří. ZÁKLADY E-DIDAKTIKY PRO E-TUTORY : Studijní materiály pro distanční kurz Dovednosti e-tutora*. 1. vyd. Ostravská univerzita v Ostravě : Ediční středisko CIT OU, 2003. 74 s. Dostupný z WWW: <http://www.osu.cz/fpd/kik/dokumenty/itvv/edidak_etuto.pdf>.

[7]*Http://www.zkusit.cz : Co je ICT?* [online]. c2007 [cit. 2008-01-20]. Dostupný z WWW: <http://www.zkusit.cz/proc-zkusit/co-je-ict.php>.

[8]KOPECKÝ , Kamil. E-learning. *Http://edo.upol.cz/* [online]. 2006, roč. 1 [cit. 2008-02-02]. Dostupný z WWW:

<http://edo.upol.cz/documents.php?sid=02e2dcbfbe5ae028c782f5ff7cebb8e2&tid=elearning>.

[9]FOJTÍK, Rostislav. M-learning. *Www.osu.cz/icte/2005/* [online]. 2005 [cit. 2008-02-20]. Dostupný z WWW: <www.osu.cz/icte/2005/23.ppt>.

[10]CTDA, TRIBAL Group. *Http://www.m-learning.org/* [online]. c2005 [cit. 2008-02-22]. Dostupný z WWW: <http://www.m-learning.org/>.

[11]*Http://www.nerudovka.cz/ : Projekt Leonardo* [online]. [2005] [cit. 2008-02-10]. Dostupný z WWW: <http://www.nerudovka.cz/modules.php?name=Content&pa=showpage&pid=5>.

[12]*Http://www.transmobile.info : Údaje o zapojení ČR do projektu Leonardo* [online]. [2005] [cit. 2008-02-15]. Dostupný z WWW: <http://www.transmobile.info/cms/index.php?option=content&task=view&id=58&Itemid=80>.

[13]PECINOVSKÝ, Rudolf. Myslíme objektově v jazyku Java 5.0 : knihovna programátora. [s.l.] : [s.n.], 2004. 604 s.

[14]VAŠÁK, Petr. Programování v jazyce Java : lekce 1. Slajdy pro přednášky k předmětu \"Programování v jazyce Java\" [online]. 2007 [cit. 2008-02-20]. Dostupný z WWW: <www.boldar.cz/java/lekce01.ppt>.

[15]HENRYCH, Radim. Visualizace XML. [s.l.], 2007. 46 s. Vedoucí bakalářské práce Ing. Petr Chmelař. Dostupný z WWW: <http://www.fit.vutbr.cz/study/DP/rpfile.php?id=4297>.

[16]NetBeans 6.0 CLDC/MIDP Development Quick Start Guide. NetBeans IDE 6.0 Mobility Documentation [online]. 2007 [cit. 2008-02-18]. Dostupný z WWW: <http://www.netbeans.org/kb/60/mobility/index.html>.

[17]A Brief History of NetBeans. Www.netbeans.org [online]. 2007 [cit. 2008-02-18]. Dostupný z WWW: <http://www.netbeans.org/about/history.html>.

[18]TOMAN, Petr . Programovací jazyk Java [online]. c2007 [cit. 2008-02-10]. Dostupný z WWW: <http://dione.zcu.cz/java/>.

[**13**]*ŠEDA, Jan. Co umožňuje Java v mobilních telefonech?. Zajímavosti : články na stránkách arti.cz* [online]. 2003 [cit. 2008-02-28]. Dostupný z WWW: <http://www.arti.cz/zajimavosti/java/co_umi_java.htm>.

[**14**]*HRONÍK, Jiří. MIDP : seminární práce. VŠE Praha kat. IT : Seminární prce* [online]. 2002 [cit. 2008-03-10]. Dostupný z WWW: <http://nb.vse.cz/~zelenyj/it380/eseje/xhroj14/midp.htm>.

[**15**]*BITTNEROVÁ, Lucie. Co vás zajímá o J2ME, ale báli jste se zeptat. Interval.cz : články o J2ME* [online]. 2002 [cit. 2008-03-03]. Dostupný z WWW: <http://interval.cz/clanky/co-vas-zajima-o-j2me-ale-bali-jste-se-zeptat/>.

[**16**]*KOLAŘÍK, Radim. MIDP : seminární práce. VŠE Praha kat. IT : Seminární prce* [online]. 2002 [cit. 2008-03-10]. Dostupný z WWW: <http://nb.vse.cz/~zelenyj/it380/eseje/xkolr03/midp.htm>.

[**17**]Package javax.microedition.lcdui.game : popis Game API na strankach www.j2medev.com od firmi Sun Microsystems. Version 2.0 of MID Profile Specification [online]. 2002 [cit. 2008-03-05]. Dostupný z WWW: <http://www.j2medev.com/api/midp/javax/microedition/lcdui/game/package-summary.html>.

[**18**]*BUREŠ, Tomáš. Java 2 MicroEdition. Přednáška o J2ME z předmětu Vybrané partie z jazyka Java : Faculty of Mathematics and Physics Charles University* [online]. 2006 [cit. 2008-03-12]. Dostupný z WWW: <http://dsrg.mff.cuni.cz/~bures/teaching/vsjava/slides2006/java11b.pdf>.

[**19**]*SAMSUNG. Stránky pro vývojáře softwaru pro mobilní telefony Samsung* [online]. c2006 [cit. 2008-03-07]. Dostupný z WWW: <http://developer.samsungmobile.com/Developer/index.jsp>.

[**20**]*GRUNT, Tomáš. Programování Java aplikací pro mobilní telefony : hra Minesweper*. [s.l.], 2007. 53 s. Vedoucí bakalářské práce Doc. RNDr. Kolář Josef, CSc. Dostupný z WWW: <https://dip.felk.cvut.cz/browse/pdfcache/gruntt1_2007bach.pdf>.

[**21**]Hnětynka, *Petr. Java 2 MicroEdition. Přednáška o J2ME z předmětu Vybrané partie z jazyka Java : Faculty of Mathematics and Physics Charles University* [online]. 2005 [cit. 2008-03-11]. Dostupný z WWW: <http://dsrg.mff.cuni.cz/~bures/teaching/vsjava/slides2006/java11b.pdf>.

[**22**]*GÜTTHANS, Václav. Java implementace hry sudoku pro prostředí mobilních telefonů*. [s.l.], 2007. 48 s. Bakalářská práce. Dostupný z WWW: <https://dip.felk.cvut.cz/browse/pdfcache/gotthv1_2007bach.pdf>.

[23]BITTNEROVÁ, Lucie. Kdo si J2ME, nezlobí : texty. *Kdo si J2ME, nezlobí* [online]. 2005 [cit. 2008-04-01]. Dostupný z WWW: <http://interval.cz/clanky/kdo-si-j2me-nezlobi-texty/>.

[24]HODEK, Václav. Programujeme v J2ME : (8.): RMS . *Programujeme v J2ME* [online]. 2004 [cit. 2008-04-03]. Dostupný z WWW: <http://www.pcsvet.cz/art/article.php?id=4769>.

[25]HORÁZNÝ, Ondřej. *E-testy : podpora domácího studia pro žáky autoškol* [online]. 2007 [cit. 2008-04-05]. Dostupný z WWW: <http://e-testy.cz/>.

[26]ETesty ministerstva dopravy České republiky: *zkoušky uchazečů o řidičské oprávnění* [online]. 2006 [cit. 2008-04-02]. Dostupný z WWW: <http://etesty.mdcr.cz/>.

Abecední rejstřík

Seznam ilustrací