Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra Informatiky

BAKALÁŘSKÁ PRÁCE

2013 Martin Hrdlička

Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra Informatiky

Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra Informatiky

Bakalářská práce

Využití myšlenkových map při výuce a učení programování

Vypracoval: Martin Hrdlička
Vedoucí práce: RNDr. Hana Havelková

České Budějovice, 2013

Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

University of South Bohemia in Českých Budějovicích
Faculty of Education
Department of Informatics

Bakalářská práce

Using Mind Maps in Programming Teaching and Learning

Author: Martin Hrdlička
Supervisor: RNDr. Hana Havelková

České Budějovice, 2013

Bibliografická identifikace

Jméno a příjmení autora: Martin Hrdlička

Název bakalářské práce: Využití myšlenkových map při výuce a učení programování

Pracoviště: Katedra Informatiky, Pedagogická fakulta, Jihočeská univerzita v Českých Budějovicích

Vedoucí bakalářské práce: RNDr. Hana Havelková

Rok obhajoby bakalářské práce: 2013

Abstrakt:

Bakalářská práce se zabývá využitím myšlenkových map při výuce programování, a to pomocí veřejně dostupného softwaru a za pomoci vytvořených technik tvorby pro tuto činnost. Postupy budou demonstrovány v objektově orientovaném programovacím jazyce Java (platforma Java SE). Práce s objekty bude zahrnuta do myšlenkových map. Tento programovací jazyk využívá velké množství knihoven, proto zde budou uvedeny často využívané knihovny a zpracovány pomocí myšlenkových map.

Hlavním cílem a praktickou částí je navrhnout metodiku pro snadnou tvorbu myšlenkových map. Dále se práce zabývá širší využitelností v oboru algoritmizace, jak uplatnit návrhy těchto map pro daná témata. Práce je zaměřena na porovnání veřejně dostupného softwaru, kde budou zahrnuty obecné požadavky na tento druh programů. Následujícím cílem je zpracování myšlenkových map, včetně takzvaných Topic map a zpracování videí pro názornou ukázku. Modelovací jazyk UML je v některých částech podobný myšlenkovým mapám. Bude posuzováno, zda jdou tyto diagramy překreslit do myšlenkových map.

Klíčová slova: OOP, myšlenkové mapy, topic mapy, pojmové mapy.

Bibliographic identification

Name and Surname: Martin Hrdlička

Title of Bachelor Thesis: Using Mind Maps in Programming Teaching and Learning

Department: Information technology in education, Pedagogicial faculty, University of South Bohemia in České Budějovice

Supervisor: RNDr. Hana Havelková

The year of presentation: 2013

Abstract:

This bachelor thesis deals with the use of mind maps in teaching programming, and at the same time using the publicly available software and techniques created with the help of production for this activity. The procedures will be demonstrated in the Java programming language, Java SE platform, which is object-oriented. Working with objects will be included in a mind map. This programming language uses a large number of libraries, so there will often use the library and processed using a mind map.

The main objective and the practical part is to propose a methodology for the design and readability entered as data. This thesis deals with wider applicability in the field of algorithms. How to implement these draft maps for these topics. I will focus on a comparison of publicly available software, which will be included in the general requirements for this type of programs. The next objective is to elaborate collection of examples of mind maps, including the so-called Topic maps and video processing for demonstration. UML have some parts like mind, will therefore be assessed whether these diagrams are going to repaint a mind map.

Keywords: OOP, mind maps, topic maps, conceptual maps.

Prohlašuji, že jsem svoji Bakalářskou práci „Využití myšlenkových map při výuce a učení programování“ vypracoval samostatně pod odborným dohledem RNDr. Hany Havelkové, 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 fakultou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích 27. 4. 2013 …….

Martin Hrdlička

Poděkování

Rád bych věnoval poděkování RNDr. Haně Havelkové za podporu při psaní této bakalářské práce a za její cenné rady.

Obsah

1 Úvod

Jsme v období, kdy přístroje využíváme v každodenním životě a požadujeme, aby nám ulehčovaly co nejvíce práci v mnoha směrech. Tudíž se vyskytuje otázka, jak docílit toho, aby přístroj pracoval podle zadaných instrukcí. Programovací jazyky slouží jako komunikační kanál mezi přístrojem a uživatelem. Neexistuje pouze jediný programovací jazyk, tak jako lidé nevyužívají pouze jeden světový jazyk. Je jich velké množství a každý má svá pravidla. Bylo vydáno mnoho publikací, jak správně jednotlivé programovací jazyky psát a využívat. Pokud chceme toto téma ovládnout, tak jsou začátky pro většinu lidí dosti náročné. Tudíž se hledají varianty, jak toto téma lépe uchopit k poznání a jeho následnému využívaní. Jedním z prostředků je možná varianta využití myšlenkových map. Je to nástroj pro snazší zapamatování a utřídění myšlenek, které nám pomohou k dosažení vytyčených cílů. Jsou vhodné pro předávání myšlenek dalším osobám a tvoření asociací.

Myšlenkové mapy jsou moderní nástroj pro zapisování dat a vytváření asociací mezi jednotlivými pojmy s využitím grafické podoby. Využívají se v mnoha oborech od vzdělávání až po byznys. Cílem je rozšířit uplatnění i pro výuku programování a algoritmizace, pro kterou není sepsána metodika jak postupovat. Pro grafický návrh objektů a tvorbu vazeb se při vytváření projektu využívá programovacích technik s využitím modelovacího jazyka UML, který využívá podobné prvky jako myšlenkové mapy.

Cílem práce je navrhnout metodiku pro tvorbu myšlenkových map při výuce programování a algoritmizace. Poté zpracovat sbírku příkladů z jednotlivých témat do uvedených map v grafické podobě a několika videí, ve kterých bude ukázáno, jak tyto mapy procházet, pro správné dekódování zanesených informací. Dále porovnat a vyhodnotit možnosti jednotlivých veřejně dostupných programů pro tvorbu těchto map. Diagramy z modelovacího jazyka UML budou porovnány, zda jdou zanést do myšlenkových map, aniž by ztratily svoji přehlednost a srozumitelnost.

Hlavním zdrojem informací pro tuto práci byla z velké části doporučená literatura. Dalším zdrojem informací byly odborné články a webové stránky zabývající se tímto tématem.

2 Úvod do programování

2.1 Programovací jazyky

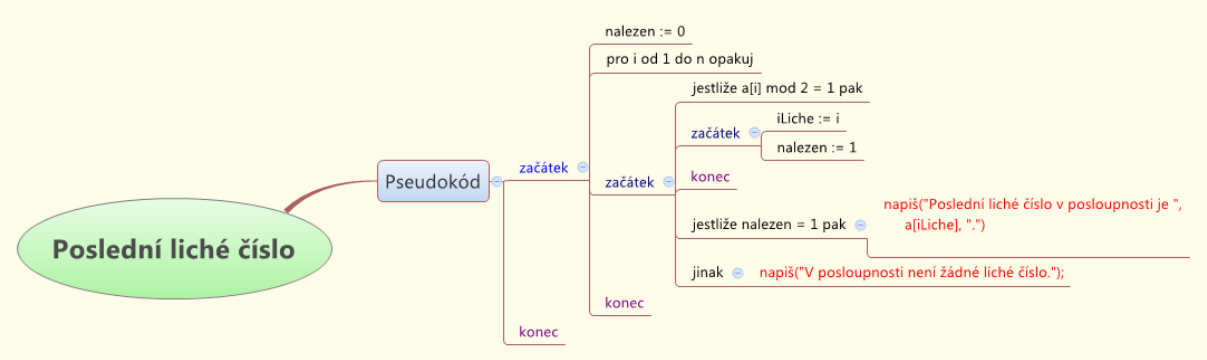
Pro zápis programů využíváme nejrůznější programovací jazyky. Program zapsaný v programovacím jazyku následně musíme nějakým způsobem převést do podoby, které rozumí počítač. Programy dělíme na překládané a interpretované, podle způsobu jakým postupují.[1]

Pokud chceme zůstat nezávislí na programovacím jazyce, tak je k tomu využíván pseudokód. Ten je určen především pro člověka nikoli pro počítač. Pseudokód se snaží odstranit složitá syntaktická pravidla, která se využívají v jednotlivých programech. V tomto druhu zápisu je vhodné využívat komentáře. Pro vysvětlení významu parametrů a specifikovat jednotlivé vstupy (input) a výstupy (output).

Nezávislost na programovacím jazyku je pouze relativní. Z konstrukce programovacích jazyků vycházejí základní řídící konstrukce pseudokódu.

Zápis v pseudokódu je kratší a stručnější, protože se pracuje na vyšší abstraktní úrovni. To znamená, že se nemusí dodržovat přesná syntaktická pravidla, jako u programovacích jazyků. Lze využívat určité pojmy, které jsou blízké dané problémové oblasti (například vstupní - výstupní informace).

U pseudokódu je předpoklad, že nebude strojově zpracováván. Tím je možné zanést jistý osobní přístup řešitele. Z tohoto důvodu může pseudokód ztratit na čitelnosti, proto se pro publikaci využívají konkrétní programovací jazyky.[2]



obr.: 1 Pseudokód

[1] PECINOVSKÝ, Rudolf. *Myslíme objektově v jazyku Java 5.0*. Vyd. 1. Praha: Grada, 2004, s. 34-35. ISBN 80-247-0941-4.
[2] Algoritmy a datové struktury. *Algoritmy a datové struktury* [online]. [20. 7. 2002] [cit. 2013-03-20]. Dostupné z: http://www.alg.webzdarma.cz/diplomka/kap2/pseudokod.html

3 Objektově orientované programování - OOP

Tvorbu složitějších programů umožnilo zavedení vyšších programovacích jazyků. Modulární programování ukazovalo, že kvalitu i rychlost vývoje výsledného programu zvýšíme, když vhodně rozdělíme velký projekt do skupiny menších, rozumně samostatných modulů.

Se stále vzrůstající složitostí vyvíjených programů se začalo ukazovat, že jednou z největších překážek v efektivní tvorbě programů je tzv. sémantická mezera mezi tím, co máme k dispozici a tím, co chceme vytvořit. Kvalita programu a rychlost jeho vyvíjení je úzce svázána s hladinou abstrakce, kterou při tvorbě využíváme.

Objektově orientované programování se proto využívá ve vyšších programovacích jazycích a ukázalo, že vhodným zvýšením abstrakce dokážeme rychleji a spolehlivěji navrhovat a vyvíjet rozsáhlejší projekty.

OOP přichází s výrazovými způsoby, které nám umožňují zvýšit hladinu abstrakce na maximum a tím zmenšit onu sémantickou mezeru mezi tím, co bychom potřebovali a tím co máme k dispozici.[3]

3.1 Principy OOP

Objektově orientované programování vychází z myšlenky, že všechny programy, jsou simulací skutečného nebo námi vymyšleného virtuálního světa. Čím bude tato podoba přesnější, tím bude konečný program dokonalejší.

Všechny tyto simulace jsou ve skutečnosti světy objektů. Objekty mají různé vlastnosti a nějakým způsobem mezi sebou komunikují. Touto komunikací se nemyslí jen komunikace pomocí řeči, ale i obecnější vzájemná interakce.

Pro co nejvěrnější převedení dané situace při vytváření projektu, by měly být schopny objektově orientované jazyky modelovat obecné objekty spolu s jejich specifickými vlastnostmi a schopnostmi a současně zachytit i jejich vzájemnou komunikaci. Čím bude tento model věrnější, tím budou programy přehlednější, ale také spolehlivější a snáze udržovatelné.

Máme-li rychle vytvářet kvalitní programy, měli bychom využívat programovací jazyk, jehož jazyková stavba nám umožní se vyjadřovat co nejpřirozeněji a co nejméně se nechat omezovat počítačem, na kterém má výsledný program běžet. Měl by umožnit co

[3] PECINOVSKÝ, Rudolf. *Myslíme objektově v jazyku Java 5.0*. Vyd. 1. Praha: Grada, 2004, s. 31-32. ISBN 80-247-0941-4.

nejlepší míru abstrakce, pro vyjádření modelované situace i jednotlivých vztahů mezi objekty.[4]

[4] PECINOVSKÝ, Rudolf. *Myslíme objektově v jazyku Java 5.0*. Vyd. 1. Praha: Grada, 2004, s. 32-33. ISBN 80-247-0941-4.

4 Java

4.1 Java a její specifikace

O dodržení zásad objektově orientovaného programování usilují tzv. objektově orientované jazyky. Mezi nimi je v současné době velmi populární programovací jazyk Java, který byl zaregistrován v roce 1995 a jeho vývoj započal v roce 1991 společností Sun Microsystems. Programovací jazyk Java je zpravidla využívaný jako vstupní jazyk pro výuku programování.[5]

4.2 Důležité vlastnosti Javy

4.2.1 Objektově orientovaná

Java podporuje objektově orientované programování. Platí obecná zásada, že Java nedovoluje zapsat program, který by nebyl alespoň z části objektově orientovaný. Tvůrci do ní začlenili převážnou většinu pravidel pro objektově orientované programování.[6]

4.2.2 Jednoduchá

Základy syntaxe může jedinec, který umí jiný programovací jazyk, zvládnout během několika hodin. Poté se již může zaměřit na poznání a pochopení funkcí klíčových knihoven a na jejich efektivní využití.

Jednoduchost tohoto jazyka neznamená jeho omezenost. Java je jedním z nejpoužívanějších programovacích jazyků současnosti. Vytvářejí se v ní aplikace pro široké uplatnění. Od programů pro čipové karty. přes programy pro mobilní telefony a různá zabudovaná zařízení, desktopové aplikace až po projekty rozmístěné na řadě vzájemně komunikujících zařízení.[7]

4.2.3 Multiplatformní

Java je současně překládaná i interpretovaná, to jí řadí mezi hybridní jazyky. Programy se překládají do speciálního tvaru nazývaného bajtkód, který pak zpracovává a interpretuje speciální program (interpret) nazývaný *virtuální stroj Javy*.

Virtuální stroj umožňuje, aby program běžel na různém zařízení a rozdílných operačních systémech. Pro každou konfiguraci hardwaru a operačního systému je nutné

[5] PECINOVSKÝ, Rudolf. *Myslíme objektově v jazyku Java 5.0*. Vyd. 1. Praha: Grada, 2004, s. 36. ISBN 80-247-0941-4.
[6] PECINOVSKÝ, Rudolf. *Myslíme objektově v jazyku Java 5.0*. Vyd. 1. Praha: Grada, 2004, s. 36. ISBN 80-247-0941-4.
[7] PECINOVSKÝ, Rudolf. *Myslíme objektově v jazyku Java 5.0*. Vyd. 1. Praha: Grada, 2004, s. 36-37. ISBN 80-247-0941-4.

vytvořit a definovat vlastní virtuální stroj. Ten zprostředkovává správný běh programu. Program ani uživatel se nemusí starat o to, na jakém hardwaru a operačním systému zrovna běží.

Java běhá pod systémy Windows, Linux, Unix, MacOS a řadou dalších operačních systémů. Programy napsané v Javě je možné spouštět na řadě mobilních zařízení, libovolném počítači se zavedeným operačním systémem i ovládat různé tipy čipových karet.[8]

4.2.4 Java je jazyk i platforma

Jedním z hlavních záměrů autorů Javy bylo vytvořit nejenom jazyk, ale celou platformu. Tuto platformu realizuje virtuální stroj Javy spolu s knihovnou nejpoužívanějších funkcí. Java se rozděluje na 4 platformy:

J2SE (Java 2 Standart Edition) je základní platforma určená pro vývoj desktopových aplikací a jednodušších serverových aplikací

J2EE (Java 2 Enterprise Edition) je nadstavbou J2SE obsahuje více knihoven specializovaných pro tvorbu rozsáhlých distribuovaných projektů.

J2ME (Java 2 Micro Edition) zjednodušená verze pro vývoj aplikací na malá zařízení. Hlavními představiteli této skupiny jsou mobilní telefony.

Java Card je platforma využívaná při vývoji aplikací pro čipové karty. Někdy bývá zařazována pod J2ME .[9]

4.2.5 Vývojové sady

Programátoři rozlišují dvě verze programových balíků:

- JRE (Java Runtime Environment - běhové prostředí Javy) obsahuje nástroje potřebné pro běh programů.
- JDK (Java Development Kit - sada pro vývoj v Javě) označována také jako SDK (Software Development Kit - sada JRE doplněná o základní vývojové nástroje) obsahuje překladač, generátor dokumentace, ladící program a další.[10]

[8] PECINOVSKÝ, Rudolf. *Myslíme objektově v jazyku Java 5.0*. Vyd. 1. Praha: Grada, 2004, s. 37. ISBN 80-247-0941-4.
[9] PECINOVSKÝ, Rudolf. *Myslíme objektově v jazyku Java 5.0*. Vyd. 1. Praha: Grada, 2004, s. 37. ISBN 80-247-0941-4.
[10] PECINOVSKÝ, Rudolf. *Myslíme objektově v jazyku Java 5.0*. Vyd. 1. Praha: Grada, 2004, s. 38. ISBN 80-247-0941-4.

5 Algoritmy

Při vytváření počítačového programu pravděpodobně implementujeme metodu, která již dříve byla pro řešení takového problému vymyšlena. Metoda je často nezávislá na platformě, a tedy pravděpodobně je vhodná pro různé počítače a odlišné programovací jazyky. Pojem *algoritmus* v informatice popisuje metodu vhodnou pro použití v počítačovém programu, který bude sloužit k řešení určitého problému. Algoritmy jsou základními prostředky informatiky. Objevují se v mnoha, ne-li ve všech oblastech této disciplíny.

Algoritmy z velké části zahrnují metody organizace dat v rámci výpočetních procesů. Objekty takto vytvořené se nazývají *datové struktury* a jsou velmi významnými objekty studia v informatice. Algoritmy a datové struktury jsou si velice blízké. Datové struktury jsou vedlejší nebo hlavní produkt algoritmů. Vznik složitých datových struktur mohou způsobit i jednoduché algoritmy, a naopak jednoduché datové struktury mohou být využívány složitými algoritmy.

Při použití počítače k řešení úloh, se běžně setkáváme s mnoha možnými, odlišnými přístupy. Malé úlohy lze řešit bez obav na zvolený přístup, pokud tento přístup dokáže vyřešit zadaný problém. V rozsáhlých úlohách, kde musíme řešit velké množství jednoduchých problémů, je žádoucí použití takových metod, které efektivně využívají čas popřípadě volné místo v paměti.

Tato disciplína nabízí potenciál k získání významných úspor, a dále umožní provádět úkoly, které by jinak nebylo možné provést. V aplikacích, kde zpracováváme velké množství dat lze docílit, aby kvůli správně navrženému algoritmu běžel program i milionkrát rychleji. Zrychlení programu dosažené zvýšením výkonu počítače nabízí oproti tomu potenciál s faktorem přibližně 10 nebo 100. Proto velmi efektivní součástí řešení rozsáhlých úloh v jakékoliv oblasti je pečlivá tvorba algoritmu.

Při vývoji velké nebo složité počítačové aplikace, je důležité věnovat mnoho úsilí definici a pochopení řešeného zadání, řízení jeho složitosti a dekompozici na menší části, které mohou být snadněji využity. Po dekompozici aplikace na menší části jsou mnohé z algoritmů zřetelně jednodušší a jsou snadněji implementovány. Existují ovšem i algoritmy, jejichž volba je nebezpečná, protože po jejich spuštění bude využívána převážná část systémových zdrojů.

Sdílení programů v počítačových systémech je stále více populární. Využití jednoduchých verzí algoritmů nám pomůže snáze je pochopit a efektivněji pak využívat pokročilé verze. Často se i ukazuje potřeba reimplementace základních algoritmů. Hlavní

příčinou je, že se provádí přechod na nové počítačové prostředí s novými vlastnostmi, jejichž původní implementace je nemusí využívat zrovna nejvýhodněji. Další z příčin pro reimplementaci základních algoritmů je, že mechanizmus sdílení softwaru není dostatečně výkonný v počítačovém systému, aby bylo možné přizpůsobit standardní programy tak, aby pracovali v požadovaných úlohách efektivně. Proto někdy bývá snazší provést novou implementaci.

Počítačové programy jsou obvykle *přeoptimalizovány*. Není vždy nutné ujišťovat se, zda je použitá implementace algoritmu opravdu nejefektivnější možná, ani v jaké četnosti se algoritmus využívá. V prvé řadě je pečlivá a relativně snadná implementace postačující. Dostaneme určitou jistotu, že konečný produkt bude pracovat, v horším případě desetkrát pomaleji než nejlepší řešení, což ve finále může znamenat několik vteřin navíc. Prvořadá snaha o výběr správného algoritmu může způsobit rozdíl s faktorem 100 nebo 1000 případně i více, což se v délce běhu programu může projevit jako minuty nebo hodiny.

Zvolení nejlepšího algoritmu pro určitou úlohu je obtížnou záležitostí, jež často zahrnuje i matematickou analýzu. Studiem těchto otázek se zabývá *analýza algoritmů*. U mnoha studovaných algoritmů, potvrdila analýza skvělou výkonnost, u jiných se potvrdilo, že pracují dobře jen experimentálně. Neměli bychom používat algoritmus bez konkrétní představy o tom, jaké bude spotřebovávat zdroje.[11]

5.1 Principy analýzy algoritmů

Analýza je prostředkem k dostatečnému pochopení algoritmů, takže je následně budeme moci efektivně aplikovat v potřebných úlohách. Dále hraje roli v procesu návrhu a implementace algoritmů. Na počátku můžeme správnou volbou algoritmu dosáhnout výrazných úspor v řádech tisíců nebo milionů. Při použití více efektivních algoritmů, se může stát problémem zvolit jeden z nich, a proto musíme znát jejich vlastnosti.[12]

5.2 Empirická analýza

Pro efektivní využití algoritmu, musí být známé jeho výkonové charakteristiky. Cílem algoritmické analýzy je rozvíjení potřebného porozumění. Jedním z prvních kroků je *empirická analýza*. Pokud k řešení stejného problému máme dva algoritmy. Spustíme oba dva a vyhodnotíme, který z uvedených algoritmů je rychlejší. Tento koncept ve srovnávacích studiích algoritmů se objevuje velmi často. Pro velké nebo složité algoritmy je empirická

[11] SEDGEWICK, Robert. *Algoritmy v C*. Praha: SoftPress, 2003, s. 20-22. ISBN 80-86497-56-9.
[12] SEDGEWICK, Robert. *Algoritmy v C*. Praha: SoftPress, 2003, s. 43-44. ISBN 80-86497-56-9.

analýza nevhodná z důvodu značného množství času, přichází tedy na řadu matematická analýza.

Empirická analýza zahrnuje určení povahy vstupních dat i dalších faktorů, které ovlivňují výsledek testu. Vstupní data lze rozdělit do tří skupin: skutečná data, náhodná data nebo zkomolená data. Skutečná data pomáhají přesně určit parametry programu při použití. Náhodná data slouží k prověření, že prováděné testy opravdu zkoušejí algoritmus a ne data. Zkomolená data slouží k testování stability programu.[13]

5.3 Analýza algoritmů

Pro některé úkoly může být vyhovující empirická analýza, ale matematická analýza může přinést mnohem více informací. Důvody proč je prováděna matematická analýza algoritmů:

- Porovnání rozdílných algoritmů pro stejnou úlohu.
- Odhad výkonnosti pro nové prostředí.
- Změna hodnot parametrů algoritmu.

Úkolem analýzy programů je, aby získala co nejvíce informací o výkonnosti algoritmu, kolik je možné, a na programátorovi je pak tyto získané informace použít při výběru algoritmu pro vytvářenou aplikaci.

Důležitým krokem při analýze algoritmů je identifikace abstraktních operací, na kterých se algoritmus zakládá, tím se oddělí analýza od implementace. Při analýze se odděluje studium toho, kolikrát jedna z implementací *sjednocení-hledání* provede část kódu a další kolik času může být vyžadováno k provedení této části kódu na daném počítači. Pro vypočtení délky běhu programu je nutné znát oba uvedené údaje. Dělí se na vlastnosti algoritmu a vlastnosti počítače. Uvedené rozdělení umožňuje porovnat algoritmy technikou, která je nezávislá na specifické implementaci konkrétního počítače.

Počet operací v algoritmu nemusí být přímo úměrný s výkonností algoritmu. Tato výkonnost je principiálně závislá na několika veličinách a zpravidla jsou tyto veličiny pro potřebnou analýzu lehce identifikovatelné. Jednou z možností jejich identifikace je profilovací mechanismus (ukazuje množství výskytu instrukcí), který zobrazí nejpoužívanější části programu pro testované vzorky. Je možné použít empirickými studie ve spojení s přibližnými matematickými výsledky, aby bylo možné předem přesně stanovit výkonnost.

Je rovněž důležité analyzovat data a modelovat vstupy algoritmu. Zpravidla bývá zvolen jeden z následujících postupů k analýze:

[13] SEDGEWICK, Robert. *Algoritmy v C*. Praha: SoftPress, 2003, s. 44-48. ISBN 80-86497-56-9.

- Předpokládat, že vstup je náhodný, a zkoumat výkonnost programu pro možný průměr.
- Pracovat se zkomolenými daty a zjišťovat nejhorší možnou výkonnost algoritmu.

Obě tyto analýzy přinášejí v mnoha případech užitečné informace o výkonnosti.[14]

5.4 Základní datové struktury

Pro vývoj počítačového programu je velmi důležitá organizace dat. Pro většinu aplikací je volba správné datové struktury nejdůležitějším rozhodnutím celé implementace. Pro stejná data potřebují některé struktury rozdílnou velikost paměti než jiné. Zvolení správné datové struktury a algoritmu získáme cestu k úspoře času i místa v paměti.

Datová struktura je aktivním objektem a nebývá pouze pasivní. Přihlídneme-li k operacím, které nad ní bývají vykonány (a na algoritmy využívané pro tyto operace). Tento koncept je zahrnut v pojmu datový typ[15].

Programy zpracovávají informace odvozené od popisů reálného nebo virtuálního světa, ať již matematických, nebo popisů v přirozené řeči. Výpočetní systém požaduje poskytnutí informací o základních prvcích takových popisů, tj. na čísla a znaky. V programovacích jazycích Java a C jsou všechny naše data zadávána pomocí několika základních datových typů:

- Celé číslo (integer)
- Číslo s pohyblivou desetinnou čárkou (float)
- znak (character)

Znaky jsou ve vyšší úrovni abstrakce nejčastěji využívány pro tvorbu slov a vět.[16]

[14] SEDGEWICK, Robert. *Algoritmy v C*. Praha: SoftPress, 2003, s. 48-51. ISBN 80-86497-56-9.
[15] Datový typ je množina hodnot a soubor operací nad těmito hodnotami.
[16] SEDGEWICK, Robert. *Algoritmy v C*. Praha: SoftPress, 2003, s. 79-91. ISBN 80-86497-56-9.

6 Myšlenkové mapy

Lidský mozek nemyslí stejně jako počítač pomocí panelu nástrojů a nabídek. Myšlení je tvořeno organicky, ve strukturách shodných s přírodními formami, jako jsou větve stromu nebo náš nervový systém. Pro správné myšlení je potřeba nástrojů, které odrážejí přírodní procesy. Myšlenková mapa je pro to vhodný nástroj. Představuje krok od lineárního, přes laterální (dvourozměrné) až k paprskovému myšlení.[17]

6.1 Definice myšlenkové mapy

Myšlenková mapa je vizuálním nástrojem pro paprskovité myšlení. Jde o proces, jakým lidský mozek přichází s nápady. Zachycením a znázorněním nápadů a myšlenek je vytvářena myšlenková mapa, odrážející to, co se děje v našem mozku. Ve své podstatě myšlenkové mapy napodobují paprskovité myšlení, a to posiluje přirozené fungování mozku a zlepšuje jeho sílu a výkonnost.[18]

6.2 Charakteristika myšlenkové mapy

Myšlenková mapa je obrazovým nástrojem pro celistvé myšlení. Podporuje především funkce mozku jako paměť, učení, kreativitu a veškeré přemýšlení.

1. Hlavní objekt pozornosti je zachycen ve středu mapy - například při psaní a plánování knihy, umístíme její obrázek do středu.
2. Ze středu mapy vycházejí různé větve. Na začátku se jedná o hlavní témata, která úzce souvisejí s centrálním prvkem (obrázkem), následně se rozkládají na podrobnější motivy.
3. Na každé větvi jsou zaznamenána klíčová slova nebo obrázky a ilustrace. Lze použít i kombinaci obou prvků.

Při tvorbě je zapotřebí uplatnit co největší kreativitu. Výhodou je obohatit myšlenkovou mapu barvami, obrázky a někdy i perspektivou (iluzí třetího rozměru). Lze použít asociace pro odkazování mezi jednotlivými větvemi. Po aplikování těchto doporučení se zpravidla zvýší její účinek.[19]

[17] BUZAN, Tony a Barry BUZAN. *Myšlenkové mapy: probuďte svou kreativitu, zlepšete svou paměť, změňte svůj život*. Vyd. 1. Brno: Computer Press, 2011, s. 41. ISBN 978-80-251-2910-4.
[18] BUZAN, Tony a Barry BUZAN. *Myšlenkové mapy: probuďte svou kreativitu, zlepšete svou paměť, změňte svůj život*. Vyd. 1. Brno: Computer Press, 2011, s. 42-46. ISBN 978-80-251-2910-4.
[19] BUZAN, Tony a Barry BUZAN. *Myšlenkové mapy: probuďte svou kreativitu, zlepšete svou paměť, změňte svůj život*. Vyd. 1. Brno: Computer Press, 2011, s. 42. ISBN 978-80-251-2910-4.

6.3 Práce s obrazy

Celá staletí se lidé domnívali, že přemýšlíme především slovy. Nyní se ale zjišťuje, že myšlenky mají nejběžněji podobu obrazců a jejich asociací. Slova jsou jen komunikačním kanálem jak tyto obrazce přenášet.

Časopis *Scientific American* publikoval výsledky jednoho experimentu, který provedl psycholog a expert na vizuální vnímání Ralph Haber. Promítl účastníkům experimentu sérii 2560 fotografií. Čas na zapamatování 1 fotografie byl 10 sekund. Cílem výzkumu bylo zjistit, jak si účastníci obrázky zapamatují.

Každému jedinci bylo předloženo 2560 fotografií. V této kolekci byly zahrnuty snímky, které již nebyly promítané. V průměru byla přesnost rozpoznávání účastníků experimentu mezi 85 a 95 procenty.

Podobný experiment provedl i další vědec, R. S. Nickerson v časopisu *Canadian Journal of Psychology*. Účastníkům promítl 600 fotografií. Čas na zapamatování jedné fotografie byla 1 sekunda. Po ukončení promítání testoval, zda účastnící dané fotografie poznají. Úspěšnost tohoto experimentu byla 98 procent.

Obrazy vzbuzují v paměti řadu asociací a podporují kreativní myšlení a paměť. Je potřeba vytvořit rovnováhu mezi používanými slovy a obrazy. Počítače, mobilní telefony a další elektronická zařízení jsou již podle toho navrženy. Využívají obrázků i slov v ikonách programů. Stejné je to s grafickým uživatelským rozhraním světoznámých webů Facebook, YouTube i ve virtuálních světech.[20]

6.4 Uživatelská příručka k myšlenkovým mapám

Pokud jste začínajícím uživatelem myšlenkových map, určitě si osvojte následující jednoduchá pravidla příručky, která mají vaše schopnosti rozšířit, nikoli je omezovat.

Pro správné pochopení myšlenkových map se uplatňují 3 fáze. Jsou nazývány jako *tři P myšlenkových map*.

1. „Přijměte" značí, dát stranou vlastní předsudky o duševních omezeních. Držet se pravidel a napodobovat obrázky.
2. „Použijte" znamená nakreslit přibližně stovku myšlenkových map podle doporučení a pravidel.

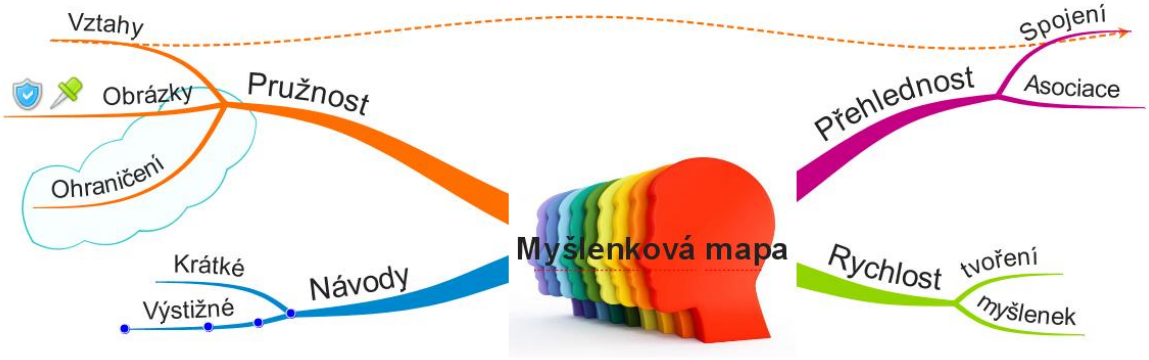
[20] BUZAN, Tony a Barry BUZAN. *Myšlenkové mapy: probuďte svou kreativitu, zlepšete svou paměť, změňte svůj život*. Vyd. 1. Brno: Computer Press, 2011, s. 51-56. ISBN 978-80-251-2910-4.

3. „Přizpůsobte" označuje rozvoj vlastních dovedností. Přichází po fázi dva, kdy přichází čas pro zapojení kreativity.[21]

6.4.1 Zvýrazňování

Pro zlepšování paměti je velmi důležitou schopností a tvůrčí schopností umění klást důraz. Všechny používané techniky zvýrazňování jsou užitečné při tvorbě asociací a naopak.

- Vždy použijte centrální obrázek. Obraz vede k zaměření pozornosti a vede k řadě asociací. Zároveň je účinný jako pomůcka pro paměť. V případě, že je nutné použít ve středu mapy slovo, je vhodné z něj vytvořit obraz přidáním barev, vhodného písma a perspektivy.
- Práce s obrazy na všech místech mapy vytvoří stimulující rovnováhu mezi slovními i obrazovými funkcemi mozkové kůry. Také se tím vylepší zrakové vnímání.
- Barvy pozitivně stimulují paměť i kreativitu. Dodávají obrazům nový podnět, proto je vhodné pro centrální obrázek použít tři i více barev.
- Použití perspektivy u slov i ilustrací způsobí snazší zapamatování. Nejdůležitější části je tedy vhodné převést do třetího rozměru.
- Pokud je to možné, je vhodné v myšlenkových mapách použít taková slova nebo obrazy, které zároveň stimulují smysly - zrak, čich, sluch chuť i hmat.[22]



obr.: 2 Obecná myšlenková mapa

6.4.2 Využití prostoru

Pro vhodné využití prostoru, který je k dispozici při tvorbě myšlenkové mapy, je vhodné dodržovat následující pravidla a doporučení:

[21] BUZAN, Tony a Barry BUZAN. *Myšlenkové mapy: probuďte svou kreativitu, zlepšete svou paměť, změňte svůj život*. Vyd. 1. Brno: Computer Press, 2011, s. 63-64. ISBN 978-80-251-2910-4.
[22] BUZAN, Tony a Barry BUZAN. *Myšlenkové mapy: probuďte svou kreativitu, zlepšete svou paměť, změňte svůj život*. Vyd. 1. Brno: Computer Press, 2011, s. 64-76. ISBN 978-80-251-2910-4.

- Důležitým mnemotechnickým nástrojem je střídání velikosti textu, čar i obrazů. Umožňuje vyznačit odlišný důraz a důležitost položek v dané hierarchii.
- Organizace rozmístění větví. Přehlednost výsledné mapy zvyšují promyšlené odstupy mezi větvemi. Zároveň napomáhají při používání hierarchií a kategorií.
- Ponecháním přiměřené mezery mezi prvky, získávají mapy lepší strukturu a pořádek. Mezery mezi položkami mohou být stejně důležité jako samotné položky.[23]

6.4.3 Asociace

Pro zlepšení paměti a kreativity jsou důležitým faktorem asociace. Lidský mozek pomocí asociací spojuje získané informace. Jde o schopnost spojit získanou informaci s již naučeným. Všechny techniky asociování mohou být využity i ke zvýrazňování.

- Pro vytvoření spojení uvnitř i vně se využívají šipky. Spojují jednu část myšlenkové mapy s druhou. Typy šipek mohou být libovolné jednosměrné i obousměrné, libovolného tvaru a velikosti.
- Barvy jsou jedním z nejužitečnějších nástrojů pro zapamatování a tvůrčí schopnost. Rozlišování významu pomocí barev, umožňuje rychlejší přístup k informacím a rozšiřuje množství nových nápadů. Zlepšují organizaci, odlišnost kategorií, uspořádání a třídění dat. Navíc stimulují větší množství mozkových buněk a usnadňují zapamatování.
- Používání různých kódů má stejný význam jako používání šipek. Dva a více prvků v myšlenkové mapě lze propojit pomocí kódu. Mohou mít jakoukoliv podobu, od jednoduchých tvarů jako trojúhelník, křížek, kolečko, čtvereček, až po propracovanější grafické prvky. Lze používat skupiny kódů pro jednotlivé oblasti myšlenkové mapy. Posilují a zlepšují práci s kategoriemi a hierarchickým uspořádání. Jako další možnost je možné využívat různé kódy k odkazování na zdroje.
- Důležitým pravidlem je srozumitelnost. Pokud nejsou informace takto zaznamenány, stěžuje se vnímání a rozvíjení celé myšlenkové mapy. Proto se využívají jasné a srozumitelné informace, které pomáhají udržet proud asociací.
- Srozumitelnost informací se zvyšuje používáním jednoho klíčového slova na každou větev. Zároveň toto pravidlo napomáhá vzniku nových spojení a asociací, protože každé slovo má miliardy možných asociací.

[23] BUZAN, Tony a Barry BUZAN. *Myšlenkové mapy: probuďte svou kreativitu, zlepšete svou paměť, změňte svůj život*. Vyd. 1. Brno: Computer Press, 2011, s. 66-68. ISBN 978-80-251-2910-4.

- Psaní tiskacích slov vede ke stručnosti a napomáhá naznačit důležitost jednotlivých slov, stejně jako velikost písmen. Slova psaná velkými písmeny jsou pro paměť snáze zapamatovatelné a napomáhají hledání asociací.
- Vhodným pravidlem je tvorba linek stejné délky, jako mají slova. Napomáhá to hledat nové asociace a myšlenková mapa pojme více informací.
- Značný význam je kladen na použití silných a přírodně tvarovaných linek, které jsou kresleny ze středu mapy. Dodávají důležitost a napomáhají při tvorbě hierarchie, zejména pokud je dané téma teprve prozkoumáváno. Organicky zakroucené tvary zvyšují pozornost.
- Pro elegantní a snadnou práci jsou účelné jednoduché myšlenkové mapy. Proto se využívají co nejjednodušší myšlenkové mapy, jak je to jen možné.
- Pokud je při tvorbě myšlenkových map používán papír, je pravidlem umístit papír do horizontální polohy. Dává více prostoru pro kreslení myšlenkových map, než u druhé varianty. Mapa vytvořená na šířku se také lépe čte.
- Při zapisování do myšlenkové mapy je důležitým pravidlem, umísťování textu do horizontální polohy. Úhel by nikdy neměl překročit 45 stupňů. V opačném případě se mapa stane nepřehlednou a těžko čitelnou.[24]

obr.: 3 Využití myšlenkové mapy

[24] BUZAN, Tony a Barry BUZAN. *Myšlenkové mapy: probuďte svou kreativitu, zlepšete svou paměť, změňte svůj život*. Vyd. 1. Brno: Computer Press, 2011, s. 68-74. ISBN 978-80-251-2910-4.

6.5 Nebezpečné oblasti při navrhování myšlenkových map

Při tvorbě myšlenkových map je nutné se vyvarovat několika oblastem, které by mohly znehodnotit celé dílo.

1. Myšlenkové mapy, které nejsou skutečnými myšlenkovými mapami.
2. Používání dlouhých frází.
3. Víra, že neuspořádaná myšlenková mapa je k ničemu.
4. Negativní reakce na vlastní myšlenkové mapy.

Do první oblasti je možné zařadit vývojové diagramy, rozvětvené osy, mapy konceptů a další schémata co vypadají jako myšlenkové mapy. Ve výsledku se myšlenkovým mapám mohou podobat, ale nerespektují jejich základní principy. Lze zde najít několik rozdílů. Rozrůstající útvary jsou stále více monotónní. Zpravidla jsou myšlenky redukovány na stejnou úroveň a netvoří žádné nové asociace.

Ve druhé části se musíme vyvarovat používání dlouhých frází. Toto pravidlo osvobozuje kognitivní i další druhy inteligence, navíc napomáhá k tvorbě nových asociací. Při používání frází je možné opomenout velké množství užitečných informací.

Do posledních dvou oblastí spadají myšlenkové mapy, které byly tvořeny v časovém presu nebo jsou záznamem chaotické přednášky. Neznamená to, že jsou špatné. Odrážejí dění v mysli nebo informace, které byli právě přijaty. Tyto mapy většinou postrádají přehlednost i estetické kvality. Je nutné je následně upravit, proto jsou chápány jako prvotní návrh, který je následně vylepšován.[25]

6.6 Využití myšlenkových map v programování a algoritmizaci

Pro různé oblasti programování se uplatnění myšlenkových map odlišuje. Při datové analýze projektu je možné tyto mapy využít, nevznikají zde žádné problémy. Rozdíl může však být u výuky syntaxe programovacího jazyka. Kdy při zapisování do myšlenkové mapy jdeme od obecného po konkrétní, to se nemusí shodovat se zápisem u jednotlivých programovacích jazyků. Například při deklaraci proměnných, kdy v Javě by vypadal zápis `int pocet = 0;` došlo by ke změně na *pocet int = 0*, protože nejobecnější částí v tomto případě je název proměnné. Konkrétnější informací je přiřazení datového typu k tomuto názvu a nakonec jeho hodnoty. Proto je navrhnutá metodika dále testována na osobách, aby byl zjištěn případný dopad těchto rozdílů. Při uplatnění myšlenkových map v algoritmizaci, dosáhneme přehledného grafického zobrazení jednotlivých příkladů, kdy následně lze lépe

[25] BUZAN, Tony a Barry BUZAN. *Myšlenkové mapy: probuďte svou kreativitu, zlepšete svou paměť, změňte svůj život*. Vyd. 1. Brno: Computer Press, 2011, s. 75-80. ISBN 978-80-251-2910-4.

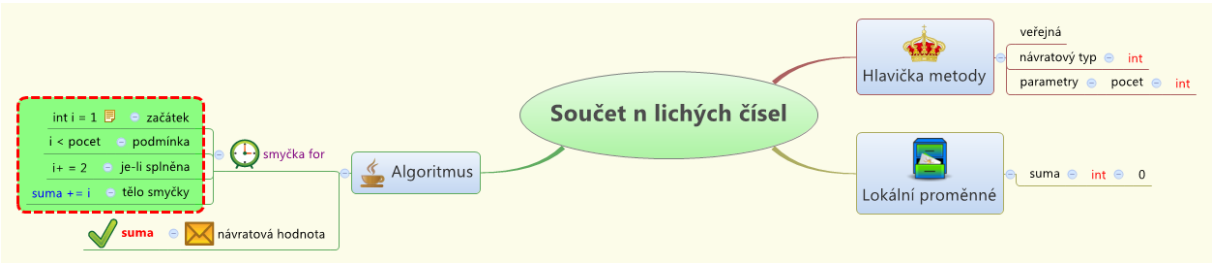
pochopit jednotlivé vnořování cyklů, podmínek… do sebe. Pro testování projektů, lze využít myšlenkové mapy. Ty pomáhají v tomto směru odhalit slabá místa projektu, případně nové možnosti jeho testování. Další možností, kde myšlenkové mapy využít je při funkční analýze. Jejich uplatnění se však vztahuje na obecné využití těchto map. U datového modelování lze ve dvou případech využít myšlenkovou mapu. Více o tomto případu užití je popsáno v desáté kapitole (UML diagramy v myšlenkových mapách).

Pro využití myšlenkových map v oboru programování a algoritmizace je nutné zachovat co nejvíce pravidel a doporučení, která zaručují přehlednost a čitelnost celé myšlenkové mapy. Programování a algoritmizace jsou specifické vědní obory, pro je nutné začlenit i některá nová pravidla.

Na začátku je nutné navrhnout, jak budeme tvořit myšlenkovou mapu. Vhodným kandidátem jsou softwarové programy, kde je možné využít neomezený prostor. Pro menší mapy lze použít nelinkovaný arch papíru. U algoritmizace bývá složitější odhadnout velikost výsledné myšlenkové mapy, protože i jednoduchá zadání mohou mít rozsáhlá řešení.

Po zvolení, kde bude myšlenková mapa tvořena, je nutné navrhnout a zakreslit *centrální prvek*. V myšlenkových mapách je pravidlem zakreslit tento prvek graficky. Pro většinu témat z programování a algoritmizace je vybrání grafického centrálního prvku velmi obtížné, až nemožné. Navíc obrázek může postrádat důležité informace, které by vedli ke špatnému řešení, již od začátku. Proto není toto pravidlo uplatněno. Možností zůstává obohacení textu o některý grafický prvek (obrázek, klipart, …).

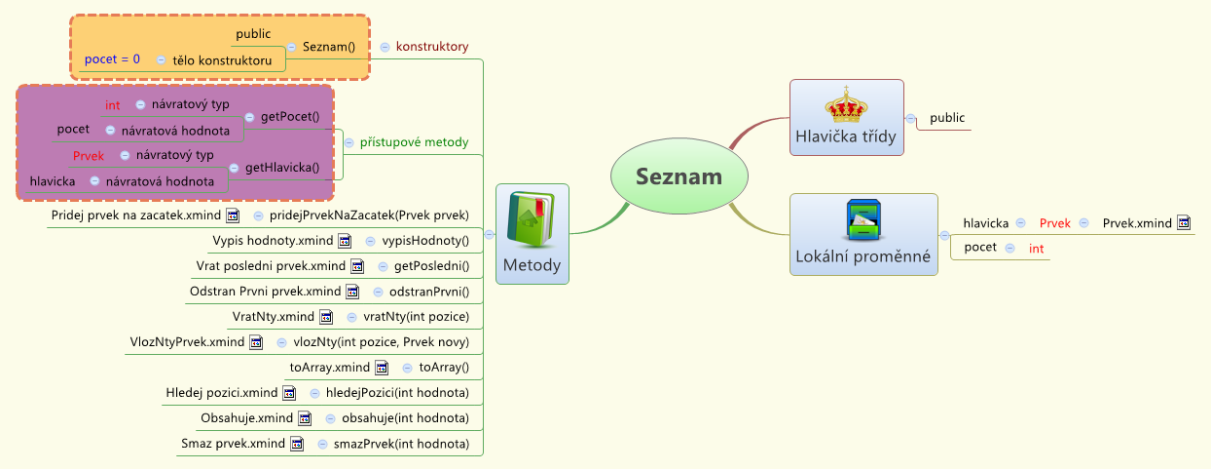
Z centrálního prvku vedou úzce související témata. V případě kdy navrhujeme pouze jednu metodu, se zaměříme na její strukturu. Postupujeme hierarchicky od hlavičky metody, přes lokální proměnné až po samotný algoritmus. Na závěr je doporučení zakreslit větev pro případné importované knihovny a balíčky, pokud o nich máme přehled.



obr.: 4 Struktura metody

Na druhou stranu při práci s větším množstvím informací je nutné zaznamenávat data hierarchicky od obecného po konkrétní. Jedná se zpravidla o objekty, které obsahují více metod nebo další objekty. Z centrálního prvku vedeme názvy jednotlivých metod (je možno doplnit informacemi o vstupních parametrech a návratové hodnotě), přístupových metod

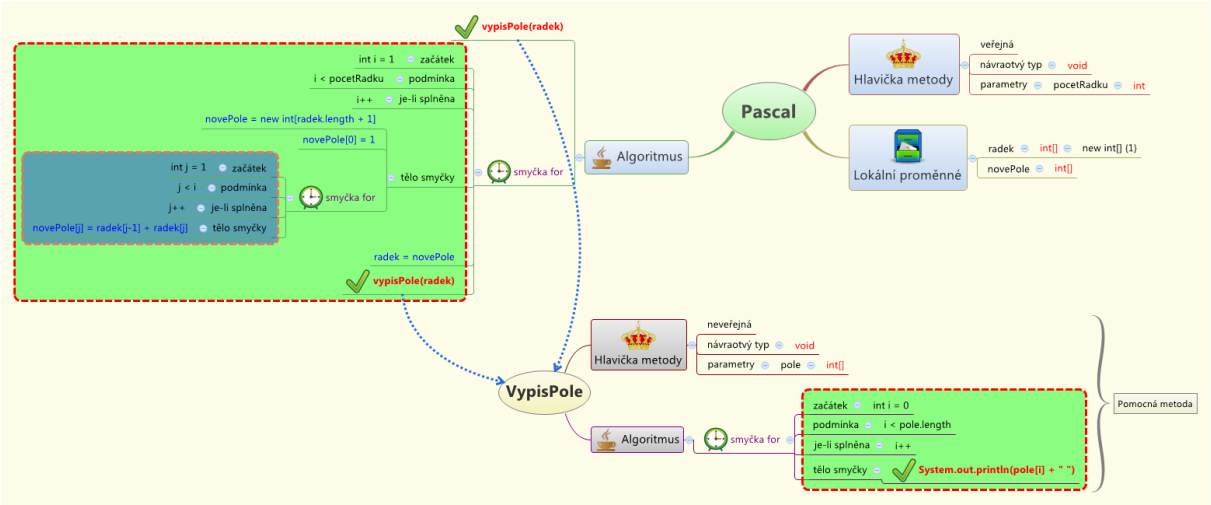
a případné konstruktory. Pokud chceme jednotlivé metody zaznamenat do detailu a je jich značné množství, je nutné využít *topic mapy*. Na uzel lze umístit odkaz na další myšlenkovou mapu (tato možnost je pouze v softwarových programech), kde již postupujeme podle jednoho z těchto postupů.



obr.: 5 Struktura třídy

Pokud dodržíme výše zmíněné postupy pro velké a malé objemy dat, získáme přehlednější myšlenkové mapy.

Pro metody, kde jsou volány další metody, je možné vytvořit vedlejší centrální prvek, kde bude popsán zmíněný kód. Například pro výpis do terminálového okna. Poté je nutné provést asociaci pomocí šipek nebo symbolů, jak na sebe data navazují. Tato situace je vyobrazena na následujícím obrázku.



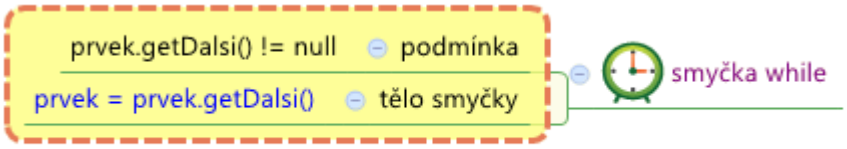
obr.: 6 Pomocná metoda

Pokud zapisujeme informace, je pravidlem dodržovat jedno klíčové slovo na každou větev. Při zápisu kódu, by však toto pravidlo způsobilo problém, proto je nutné jej upravit. Na každou větev je možné zaznamenat maximálně jeden řádek zdrojového kódu, který je zpravidla v Javě označován středníkem.

Pravidla pro grafickou podobu větví a doplnění myšlenkové mapy o obrázky jsou i v tomto oboru důležitým prvkem, který by neměl být opomenut. Pokud jsou tvořeny mapy pro výuku, měla by být zvolena sada obrázků a barev, která bude použita u všech myšlenkových map shodně.

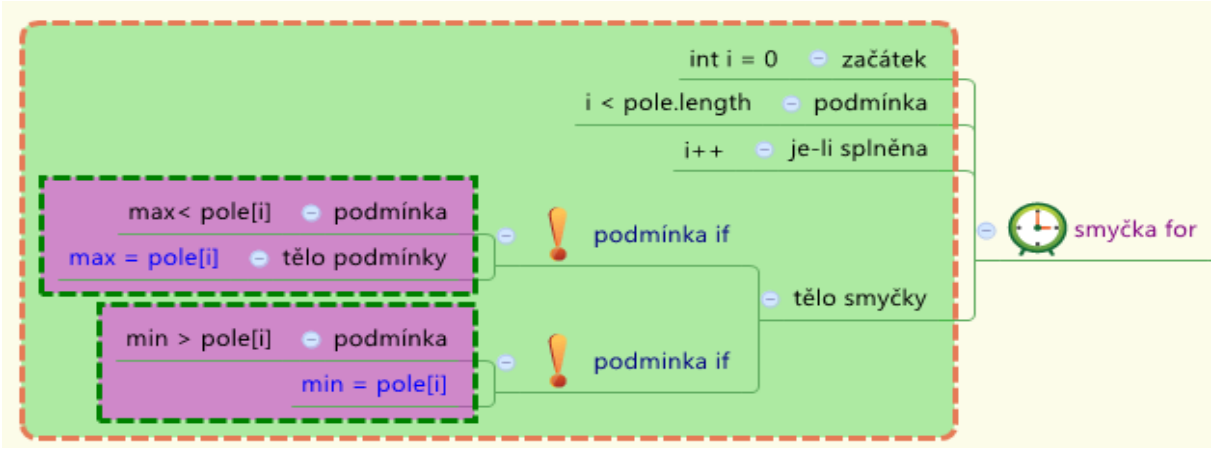
6.6.1 Podmínky, cykly a výjimky v myšlenkových mapách

Pro zaznamenání podmínek a cyklů do myšlenkové mapy je nutné postupovat podle následujícího pravidla. Každý cyklus a podmínku lze rozdělit na dvě části: první, kde je obsažena podmínka pro vykonání kódu, a na část, která se dá označit jako tělo podmínky. Tato situace je vyobrazena na následujícím obrázku. Velmi podobným způsobem lze zakreslit i výjimky. Část, kde je kód obsažený v bloku *try* a na část následného zachycení a zpracování výjimky.



obr.: 7 Smyčka while

Tímto pravidlem lze snadno a přehledně vnořovat více podmínek a smyček do sebe.



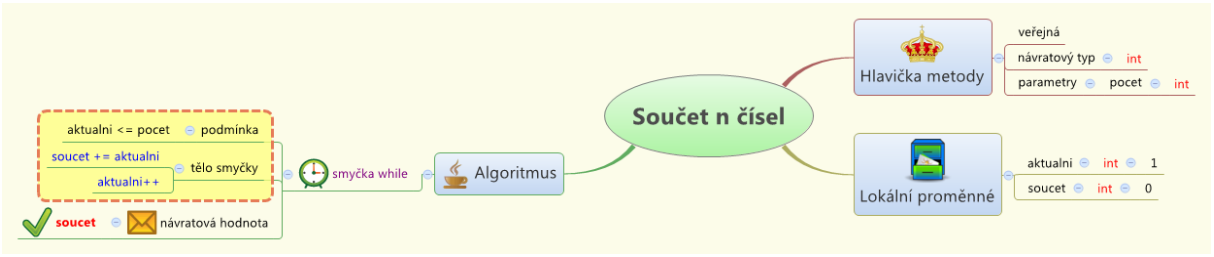
obr.: 8 Podmínky ve smyčce

6.7 *Příklady v myšlenkových mapách*

Zpracované příklady lze rozdělit do několika skupin z oborů algoritmizace a programování. Jedná se o příklady, se kterými se začínající programátoři mají možnost setkat a pochopit na nich lépe principy programování.

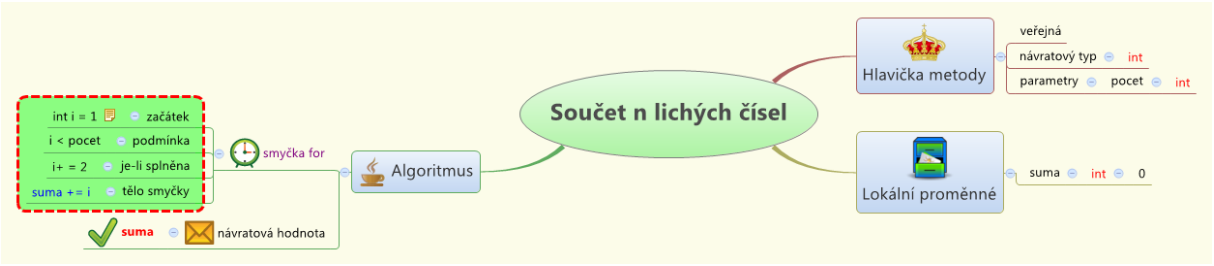
6.7.1 Sbírka příkladů

1. Metoda vrátí součet čísel od 0 do zadaného počtu.



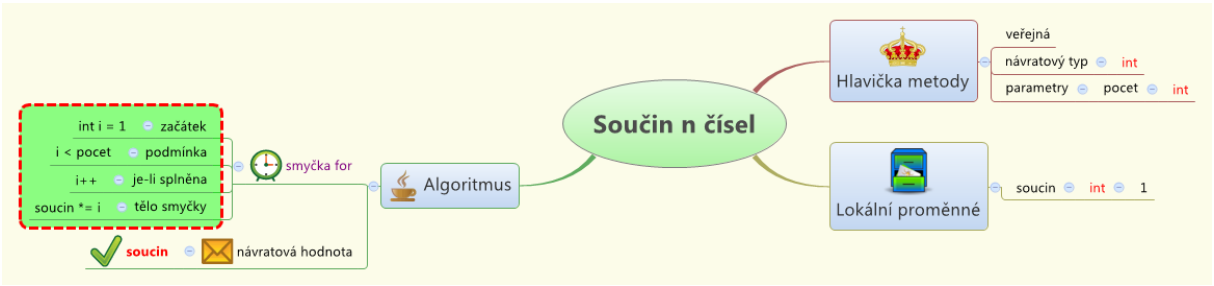
obr.: 9 Součet n čísel

2. Metoda vrátí součet lichých čísel od 0 do zadaného počtu.



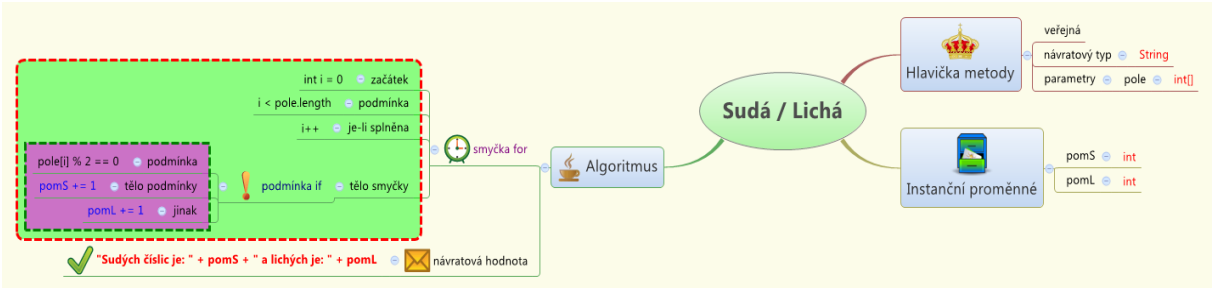
obr.: 10 Součet n lichých čísel

3. Metoda vrátí součin čísel od 0 do zadaného počtu.



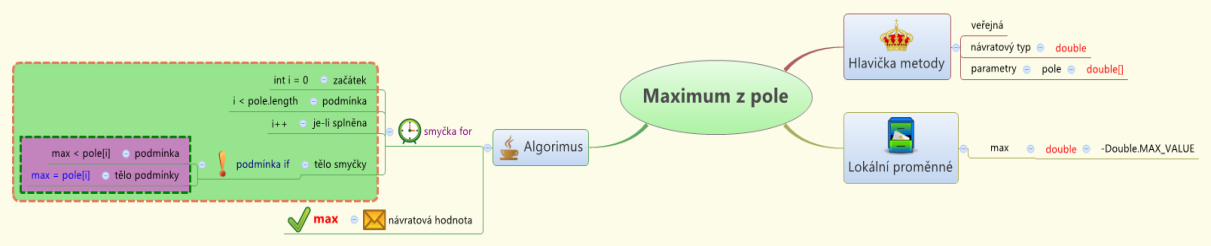
obr.: 11 Součin n čísel

4. Metoda zjistí počet kladných a sudých čísel v poli.



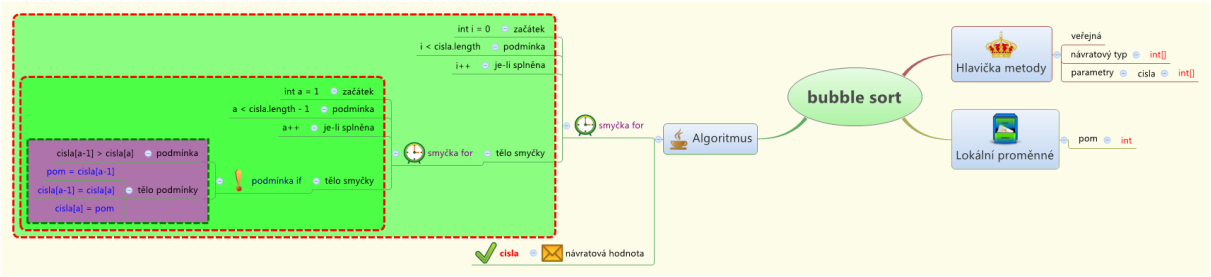
obr.: 12 Sudá / Lichá

5. Metoda vrátí maximální hodnotu čísla ze zadaného pole.



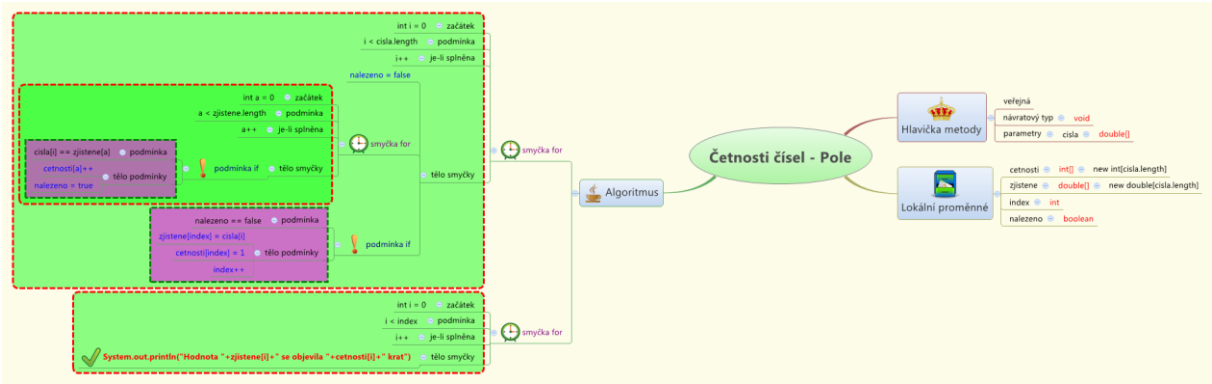
obr.: 13 Maximum z pole

6. Metoda seřadí zadané pole od nejmenšího čísla po největší.



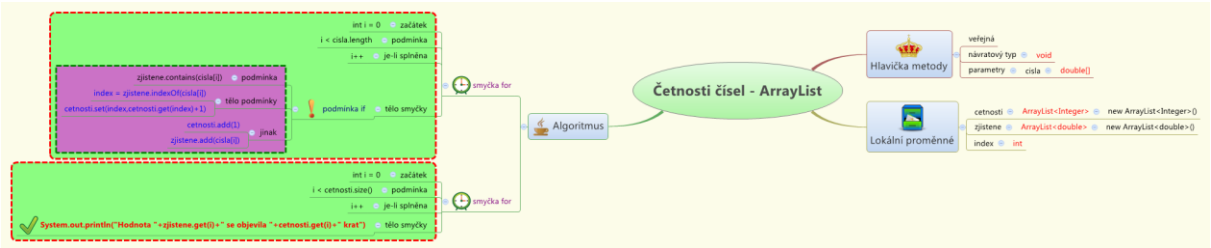
obr.: 14 Bubble sort

7. Metoda zjistí četnosti čísel ze zadaného pole. Pro počítání četnosti čísel jsou použita pole.



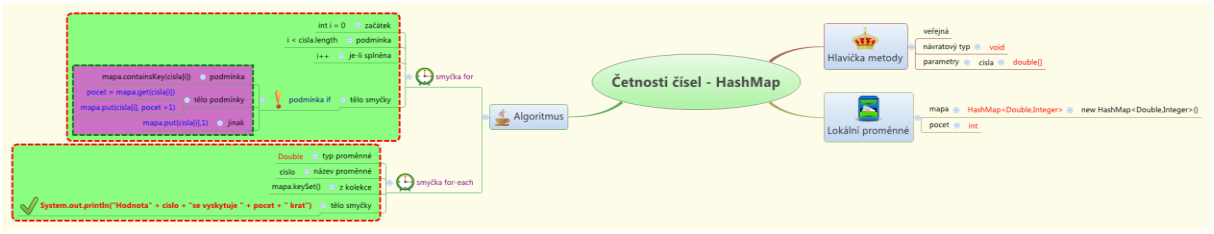
obr.: 15 Četnosti čísel - Pole

8. Metoda zjistí četnosti čísel ze zadaného pole. Pro počítání četnosti čísel je použita třída ArrayList<E> z balíčku Java.util.



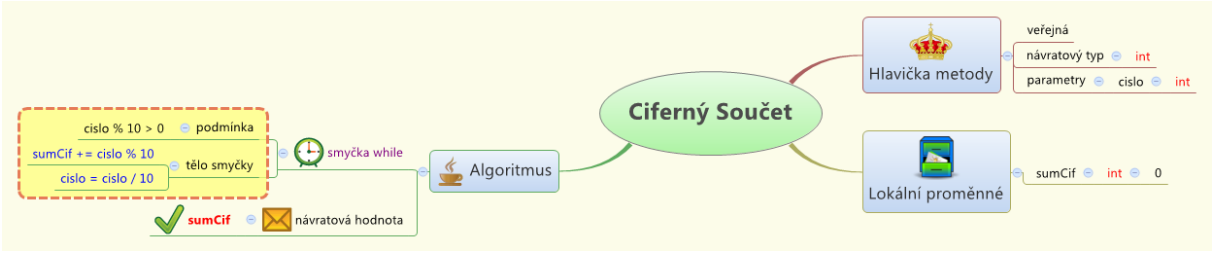
obr.: 16 Četnosti čísel - ArrayList

9. Metoda zjistí četnosti čísel ze zadaného pole. Pro počítání četnosti čísel je použita třída HashMap<K,V> z balíčku Java.util.



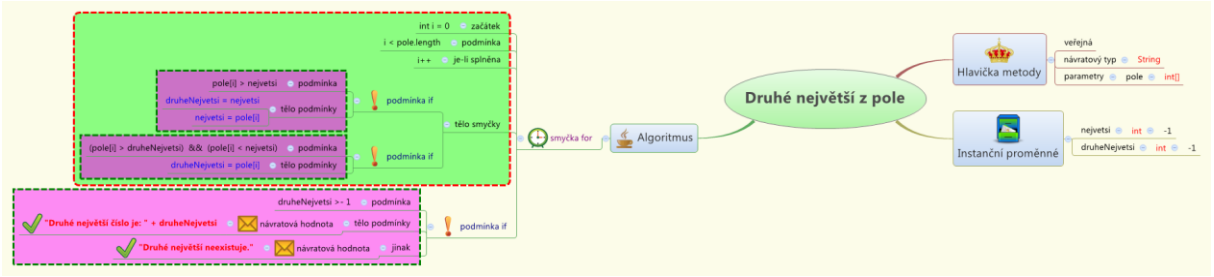
obr.: 17 Četnosti čísel - HashMap

10. Metoda vrátí ciferný součet ze zadaného čísla. Metoda obsahuje jeden parametr typu int.



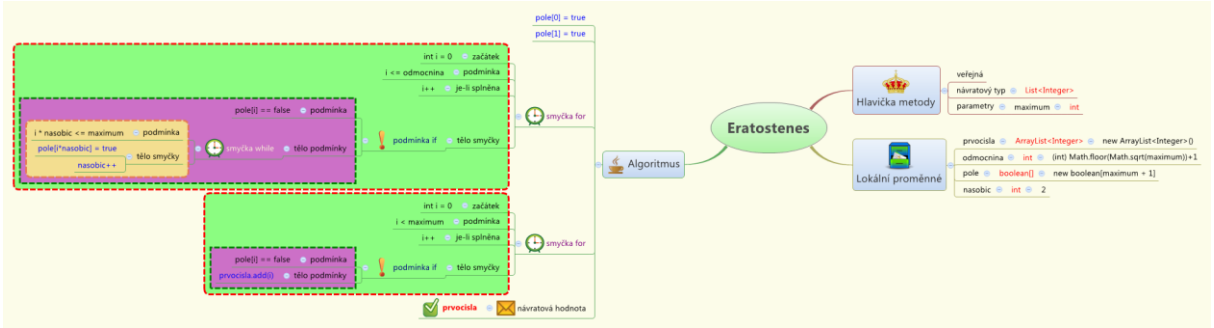
obr.: 18 Ciferný součet

11. Metoda zjistí a vypíše do terminálového okna druhé největší kladné číslo ze zadaného pole. Pokud druhé největší číslo neexistuje je vypsána chybová hláška.



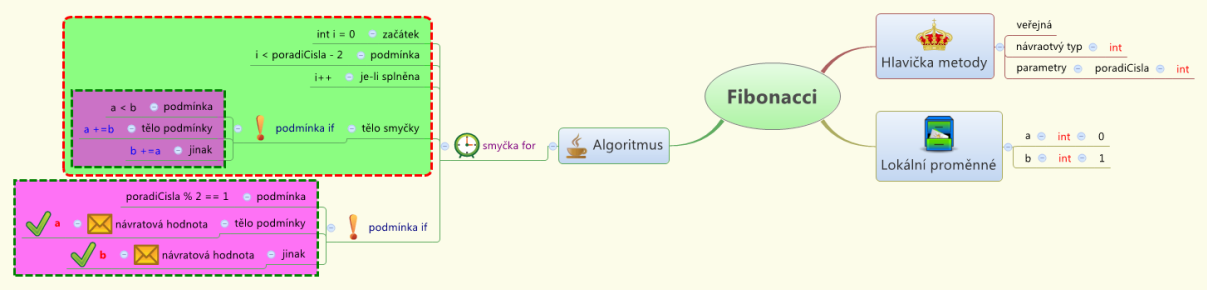
obr.: 19 Druhé největší číslo z pole

12. Metoda vyhledá všechna prvočísla menší než zadaná horní mez a vrátí jejich seznam.



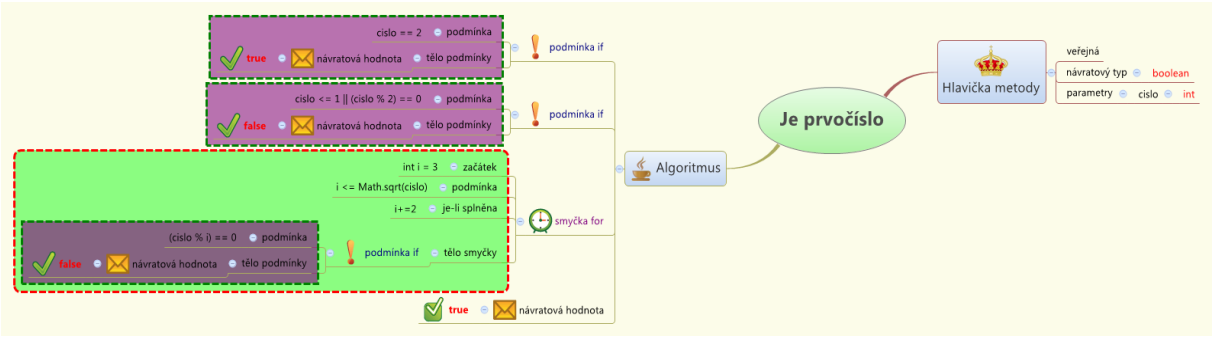
obr.: 20 Eratostenes

13. Metoda vyjadřující Fibonacciho posloupnost.



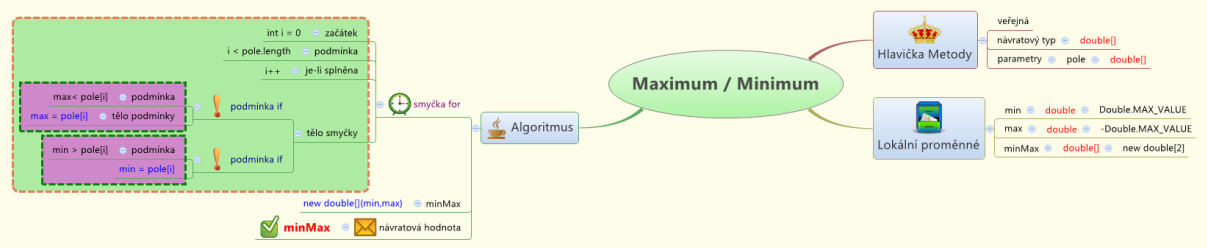
obr.: 21 Fibonacci

14. Metoda zjistí, zda zadané číslo je prvočíslem. Pokud ano, je vrácena hodnota *true*.



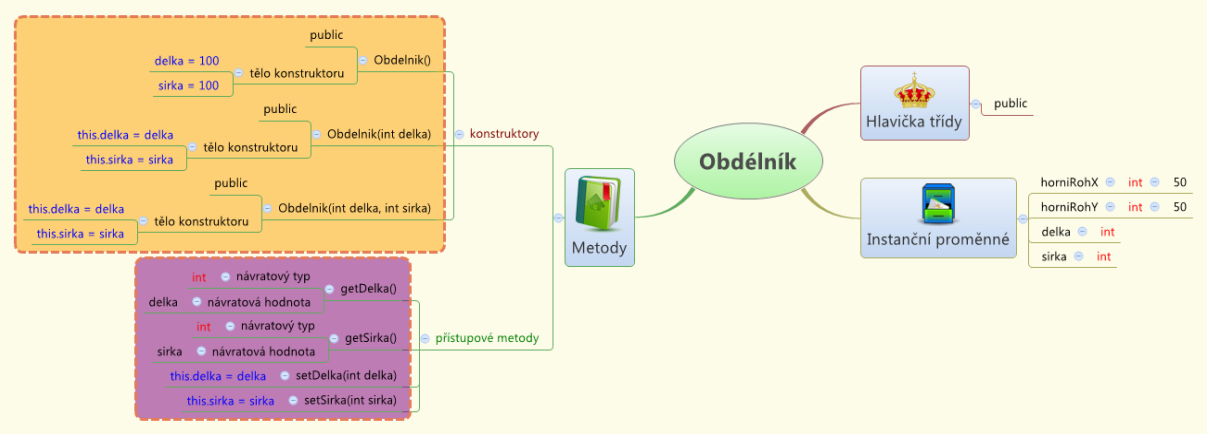
obr.: 22 Je prvočíslem?

15. Metoda vrátí maximální a minimální hodnotu čísla ze zadaného pole.



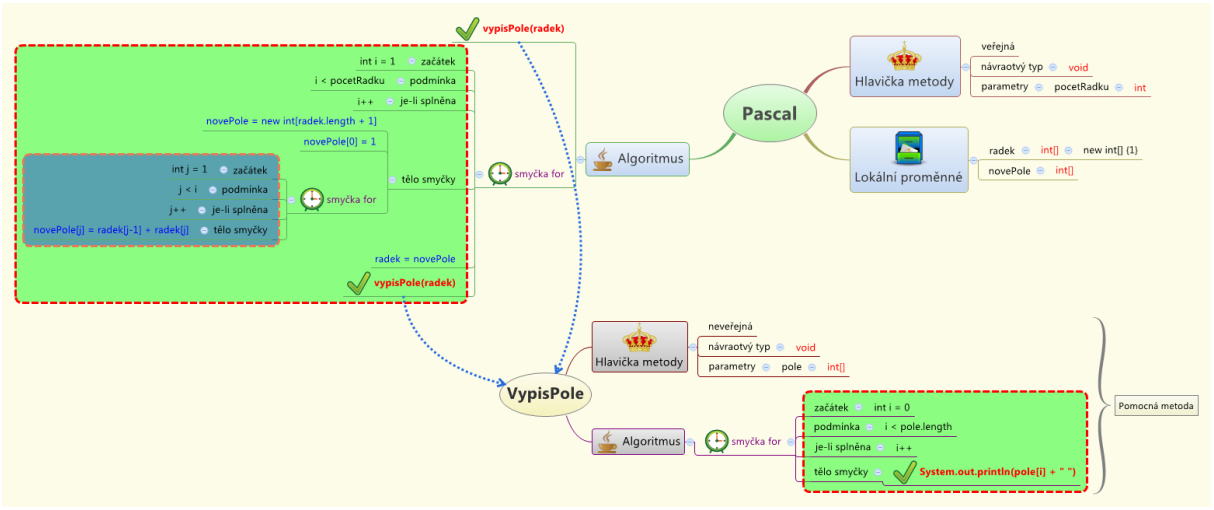
obr.: 23 Maximum / Minimum

16. Třída Obdélník obsahuje tři konstruktory (bez parametru, s jedním parametrem a se dvěma parametry) a přístupové metody k lokálním proměnným, které určují počáteční bod (X a Y) obdélníku, dále jeho šířku a výšku.



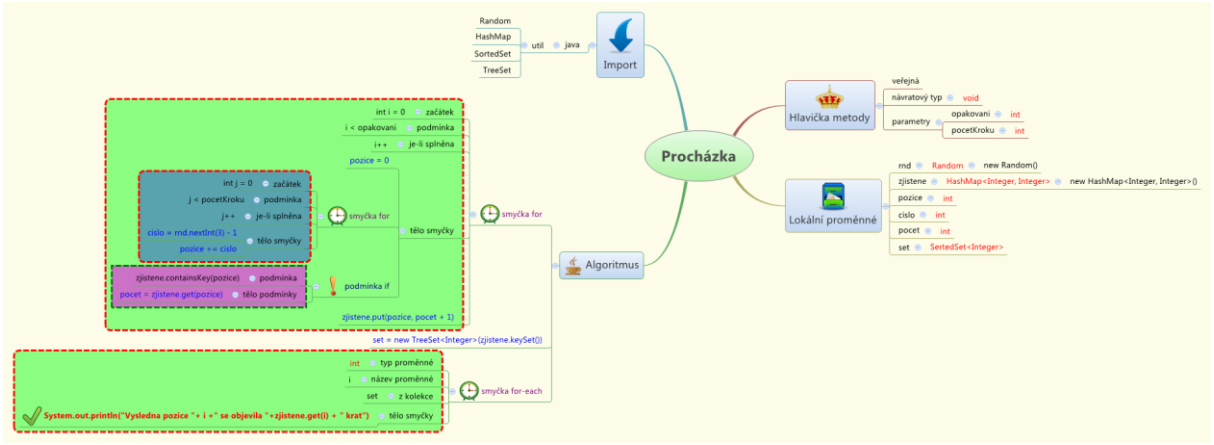
obr.: 24 Obdélník

17. Metoda reprezentující Pascalův trojúhelník. Geometrické uspořádání binomických koeficientů do tvaru trojúhelníku.



obr.: 25 Pascal

18. Metoda náhodně generuje tři stavy. Simulují krok vzad, zůstat na místě a krok vpřed, až do zvoleného počtu opakování. Po ukončení generování stavů je výsledek vypsán do terminálového okna.



obr.: 26 Procházka

6.7.2 Video příklady

Videa zpracovaných příkladů v myšlenkových mapách jsou obsaženy v příloze.

6.8 Testování metodiky

Navrhnutá metodika napomáhá uživateli zvolit vhodný typ mapy (myšlenkové, pojmové, topic) pro zadaný úkol. Poté popisuje jak do mapy zapisovat, případně z ní číst informace. Vysvětluje zakreslování datových typů, kolekcí, podmínek, smyček, výjimek, konstruktorů, přístupových metod…, aby bylo možné s nimi dále pracovat. Pro předávání

informací mezi učitelem a žákem slouží jako komunikační kanál. Pokud metodika není dodržena, vede to ke vzniku především závažných chyb, které způsobí nepoužitelnost celé mapy.

Navrhnutá metodika byla zatím testována na osmi vybraných osobách. Jejich jména nejsou dále uvedena a jsou označovány jako O1, O2… Základním předpokladem pro testování, byla základní znalost programování a algoritmů. Znalost myšlenkových map nebyla povinná. Testování probíhalo ve třech fázích:

1. Výklad, kde byla testovaná osoba seznámena s metodikou. Tato část trvala 15 minut.
2. Převod myšlenkové mapy na zdrojový kód.
3. Překreslení zdrojového kódu do myšlenkové mapy.

Po každém z těchto bodů mohla testovaná osoba přejít zpátky k bodu jedna, to i opakovaně nebo fázi jedna předčasně ukončit. Zaznamenával se výsledný čas pro pochopení navrhnuté metodiky a výstupy z bodů dva a tři.

6.8.1 Test O1

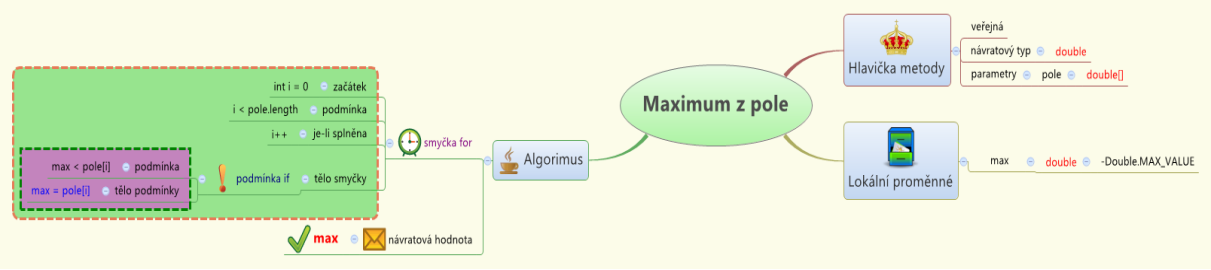
Základní informace:

Pohlaví: Muž

Věk: 22 let

Testovaná osoba má předchozí zkušenosti s tvorbou myšlenkových map.

1) Čas potřebný pro osvojení metodiky: 15 minut
2) Pro převod myšlenkové mapy na zdrojový kód byla použita následující myšlenková mapa.



obr.: 27 Maximum z pole

Výsledný kód:

```
public double maximumzPole(double[] pole){
      double act = -Double.MAX_VALUE;
      for(int i = 0; i < pole.length;i++){
            if(act < pole[i]){
```

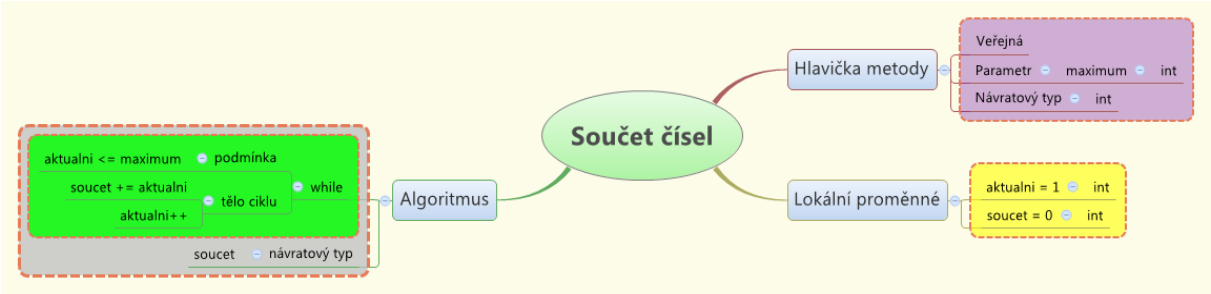
```
                act = pole[i];
            }
        }
        return act;}
```

Při převodu na zdrojový kód nevznikla žádná chyba.

3) Při překreslení zdrojového kódu do myšlenkové mapy byl použit následující kód:

```
public int soucetCisel(int maximum){
int aktualni = 1;
int soucet = 0;
        while(aktualni <= maximum){
            soucet += aktualni;
            aktualni++;
        }
return soucet;
}
```

Výsledná myšlenková mapa:



obr.: 28 Výsledná mapa O1 - součet N čísel

V myšlenkové mapě nebyly zaznamenány žádné závažné nedostatky. U lokálních proměnných byla nalezena drobná chyba, kdy k názvu této proměnné byla rovnou přiřazena její hodnota. Ta by měla být podle navrhnutých pravidel uvedena jako samostatná položka.

6.8.2 Test O2

Základní informace:

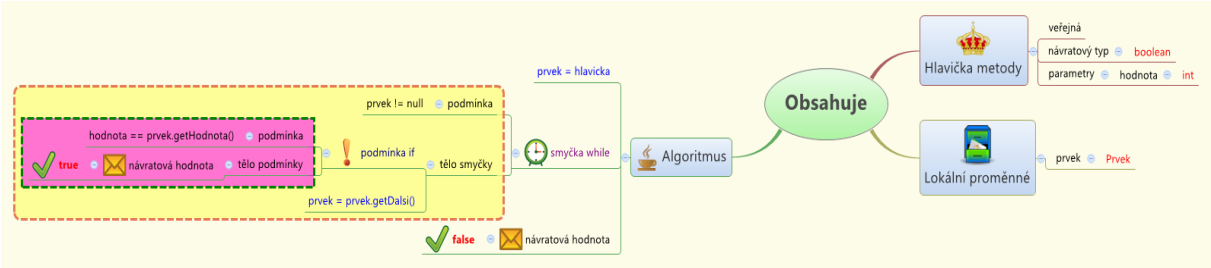
Pohlaví: Žena

Věk: 22 let

Testovaná osoba nemá předchozí zkušenosti s tvorbou myšlenkových map.

1) Čas potřebný pro osvojení metodiky: 23 minut

2) Pro převod myšlenkové mapy na zdrojový kód byla použita následující myšlenková mapa.



obr.: 29 Obsahuje

Výsledný kód:

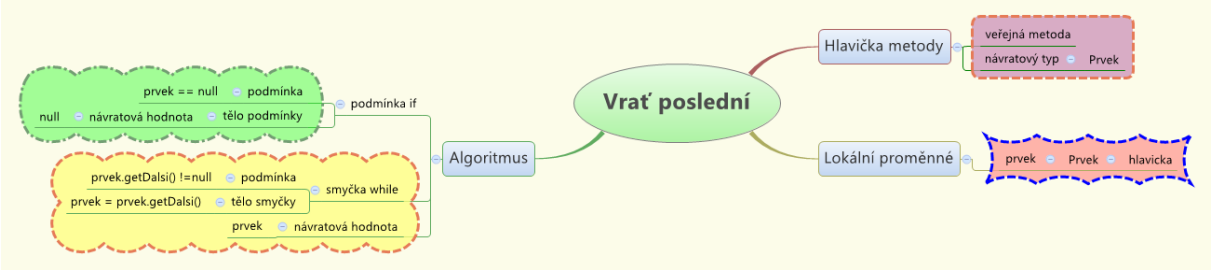
```
public boolean obsahuje(int hodnota){
      Prvek prvek = hlavicka;
      while(prvek != null){
            prvek = prvek.getDalsi();     Chyba
            if(hodnota == prvek.getHodnota()){
                  return true;
            }
            prvek = prvek.getDalsi();     Oprava
      }
      return false;}
```

Vyznačená chyba vznikla nedodržením pravidla, při kterém jednotlivé prvky ve větvi zapisujeme po řádkách.

3) Při překreslení zdrojového kódu do myšlenkové mapy byl použit následující kód:

```
public Prvek vratPosledni(){
Prvek prvek = hlavicka;
      if(prvek == null){
            return null;
      }
      while(prvek.getDalsi() != null){
            prvek = prvek.getDalsi();
      }
return prvek;}
```

Výsledná myšlenková mapa:



obr.: 30 Výsledná mapa O2 - toArray

Při zakreslování do myšlenkové mapy nevznikla žádná chyba.

6.8.3 Test O3

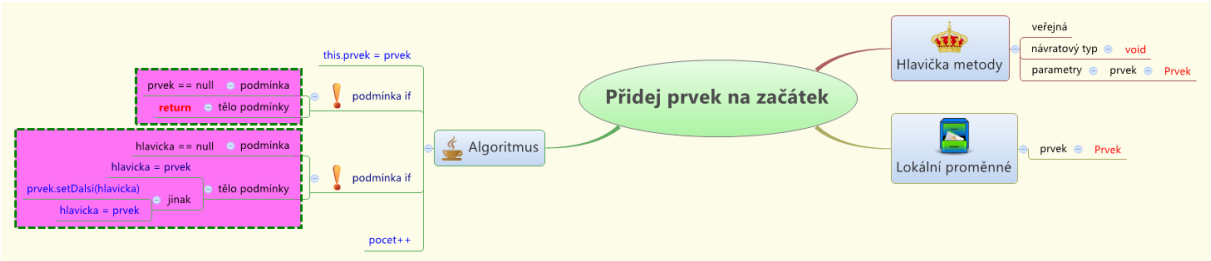
Základní informace:

Pohlaví: Žena

Věk: 21 let

Testovaná osoba nemá předchozí zkušenosti s tvorbou myšlenkových map.

1) Čas potřebný pro osvojení metodiky: 20 minut
2) Pro převod myšlenkové mapy na zdrojový kód byla použita následující myšlenková mapa.



obr.: 31 Přidej prvek na začátek seznamu

Výsledný kód:

```
public void PridejNaZacatek(Prvek prvek){
      Prvek prvek;    Chyba
      this.prvek = prvek;
      if(prvek == null){
            return;
      }
      if(hlavicka == null){
            hlavicka = prvek;
      }else{
      prvek.setDalsi(hlavicka);
```

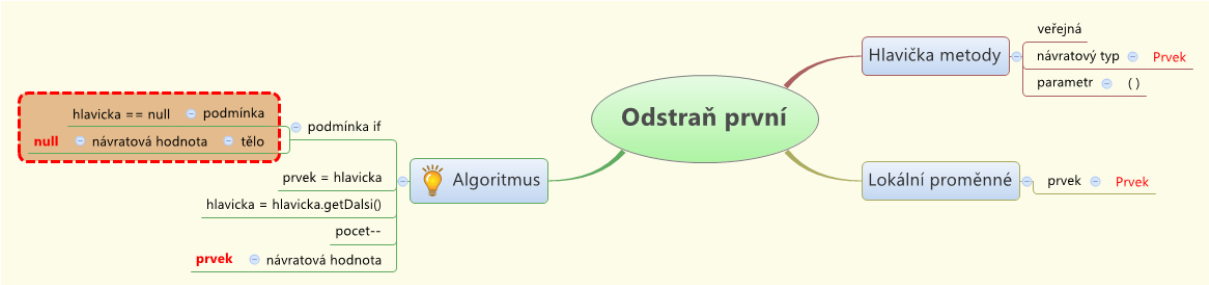
```
        hlavicka = prvek;
        }
        pocet++;
}
```

Při převodu byla vynechána deklarace lokální proměnné.

3) Při překreslení zdrojového kódu do myšlenkové mapy byl použit následující kód:

```
public Prvek OdstranPrvni(){
        if(hlavicka == null){
                return null;
        }
        prvek = hlavicka;
        hlavicka = hlavicka.getDalsi();
        pocet--;
        return prvek;
}
```

Výsledná myšlenková mapa:



obr.: 32 Výsledná mapa O3 - Odstraň první ze seznamu

Při zakreslování do myšlenkové mapy nevznikla žádná chyba.

6.8.4 Test O4

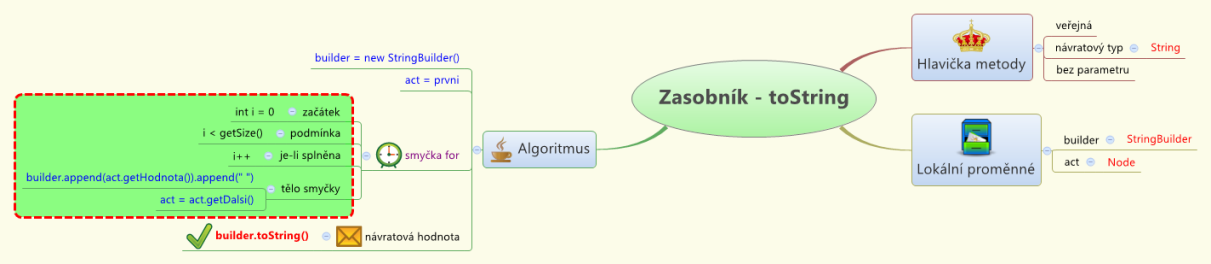
Základní informace:

Pohlaví: Muž

Věk: 22 let

Testovaná osoba má předchozí zkušenosti s tvorbou myšlenkových map.

1) Čas potřebný pro osvojení metodiky: 17 minut
2) Pro převod myšlenkové mapy na zdrojový kód byla použita následující myšlenková mapa.



obr.: 33 Zásobník - toString

Výsledný kód:

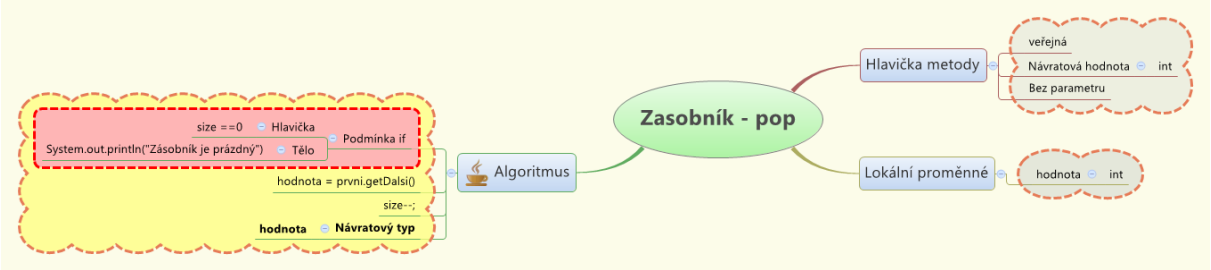
```
public String toString(){
      StringBuilder builder;
      Node act;
      builder = new StringBuilder();
      act = prvni;
for(int i =0; i < getSize(); i++){
      builder.append(act.getHodnota()).append(" ");
      act = act.getDalsi();    // drobná chyba
}
      return builder.toString();}
```

Při převodu na zdrojový kód vznikla pouze drobná chyba způsobená nepozorností.

3) Při překreslení zdrojového kódu do myšlenkové mapy byl použit následující kód:

```
public int pop(){
      if(size == 0){
            System.out.println("Zásobník je prázdný");
      }
      int hodnota = prvni.getHodnota();
      prvni = prvni.getDalsi();
      size--;
      return hodnota;
}
```

Výsledná myšlenková mapa:



obr.: 34 Výsledná mapa O4 - Zásobník pop

Při zakreslování do myšlenkové mapy nevznikla žádná závažná chyba. Pouze u dekrementace lokální proměnné *size* nemusí být na konci řádku středník, protože jednotlivé příkazy jsou vždy psány na jednu řádku.

6.8.5 Test O5

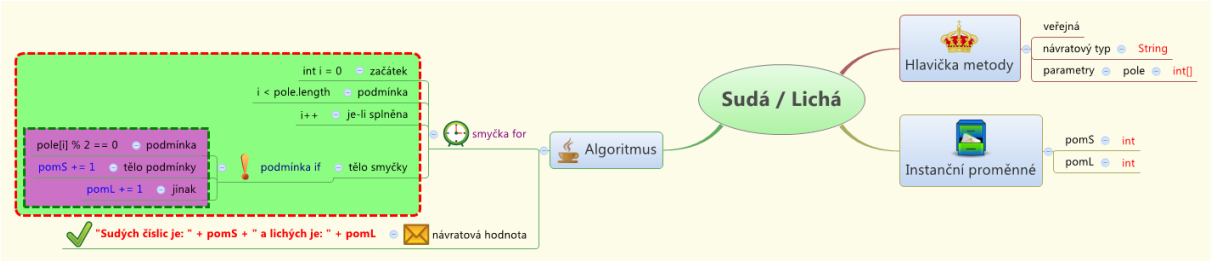
Základní informace:

Pohlaví: Muž

Věk: 23 let

Testovaná osoba nemá předchozí zkušenosti s tvorbou myšlenkových map.

1) Čas potřebný pro osvojení metodiky: 22 minut
2) Pro převod myšlenkové mapy na zdrojový kód byla použita následující myšlenková mapa.



obr.: 35 Počet sudých a lichých čísel v poli

Výsledný kód:

```
public String sudaLicha(int[] pole){ // Chyba
      int pomS = 0;
      int pomL = 0;
for(int i =0; i < pole.length; i++){
      if(pole[i] % 2 == 0){
            pomS +=1;
      }else{
            pomL +=1;
      }
```

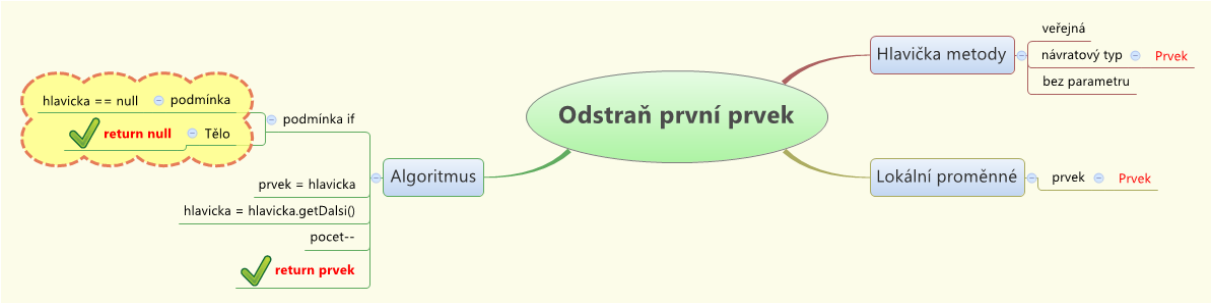
```
}
return "Sudých číslic je: " + pomS + " a lichých je: " +
        pomL;}
```

Při převodu na zdrojový kód se vyskytla chyba, která vznikla patrně nepozorností.

3) Při překreslení zdrojového kódu do myšlenkové mapy byl použit následující kód:

```
public Prvek odstranPrvni(){
      Prvek prvek;
      if(hlavicka == null){
            return null;
      }
      prvek = hlavicka;
      hlavicka = hlavicka.getDalsi();
      pocet--;
      return prvek;
}
```

Výsledná myšlenková mapa:



obr.: 36 Výsledná mapa O5 - Odstraň první prvek

Při zakreslování do myšlenkové mapy nevznikla žádná chyba.

6.8.6 Test O6

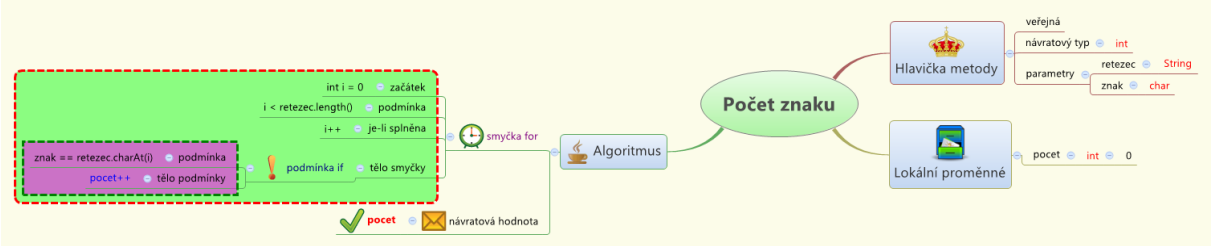
Základní informace:

Pohlaví: Žena

Věk: 22 let

Testovaná osoba nemá předchozí zkušenosti s tvorbou myšlenkových map.

1) Čas potřebný pro osvojení metodiky: 19 minut
2) Pro převod myšlenkové mapy na zdrojový kód byla použita následující myšlenková mapa.



obr.: 37 Počet znaku v řetězci

Výsledný kód:

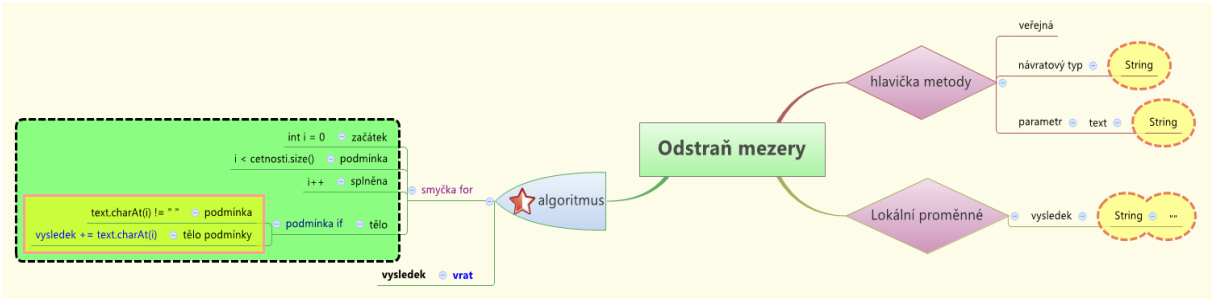
```
public int pocetZnaku(String retezec, char znak){
      int pocet = 0;
      for(int i= 0; i<retezec.length(); i++){
      if(znak == retezec.charAt(i)){
            pocet++;
      }
      }
      return pocet;
}
```

Při převodu na zdrojový kód nevznikla žádná chyba.

3) Při překreslení zdrojového kódu do myšlenkové mapy byl použit následující kód:

```
public String odstranMezery(String text){
      String vysledek = "";
      for(int i = 0; i <text.length();i++){
      if(text.charAt(i) != " "){
            vysledek += text.charAt(i);
      }
      }
return vysledek;
}
```

Výsledná myšlenková mapa:



obr.: 38 Výsledná mapa O6 - Odstraň mezery z řetězce

Při zakreslování do myšlenkové mapy nevznikla žádná chyba.

6.8.7 Test O7

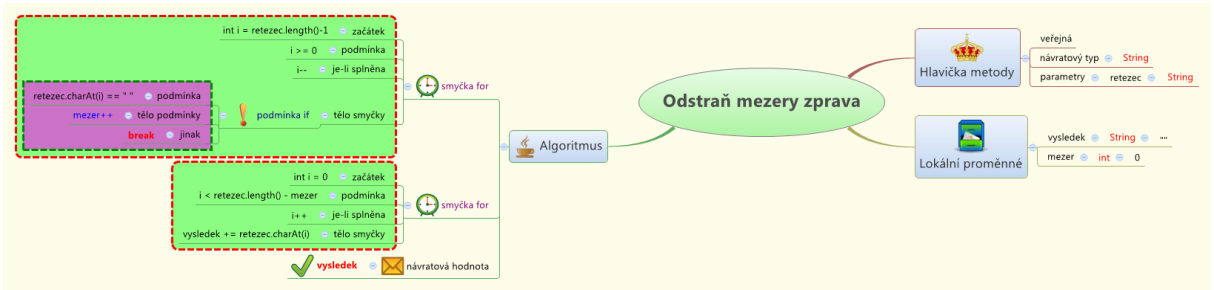
Základní informace:

Pohlaví: Muž

Věk: 22 let

Testovaná osoba má předchozí zkušenosti s tvorbou myšlenkových map.

1) Čas potřebný pro osvojení metodiky: 20 minut
2) Pro převod myšlenkové mapy na zdrojový kód byla použita následující myšlenková mapa.



obr.: 39 Odstraň mezery zprava

Výsledný kód:

```
public String odstranzPrava(String retezec){
      int mezer = 0;
      for(int i = retezec.length()-1; i >= 0; i--){
            if(retezec.charAt(i) == " "){
                  mezer++;
            }else{
                  break;
            }
      }
```

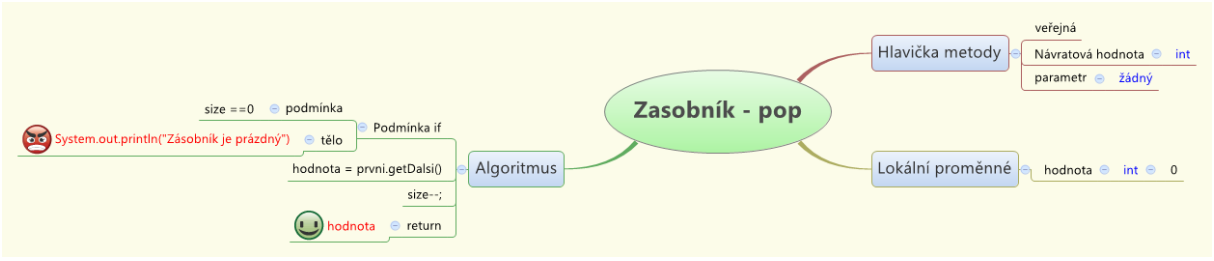
```
        String vysledek = "";
        for(int i = 0; i < retezec.length() - mezer; i++){
            vysledek += retezec.charAt(i);
        }
        return vysledek;
}
```

Při převodu na zdrojový kód nevznikla žádná chyba.

3) Při překreslení zdrojového kódu do myšlenkové mapy byl použit následující kód:

```
public int pop(){
        if(size == 0){
            System.out.println("Zásobník je prázdný");
        }
        int hodnota = prvni.getHodnota();
        prvni = prvni.getDalsi();
        size--;
        return hodnota;}
```

Výsledná myšlenková mapa:



obr.: 40 Výsledná mapa O7 - Zásobník - pop

Při zakreslování do myšlenkové mapy nevznikla žádná chyba.

6.8.8 Test O8

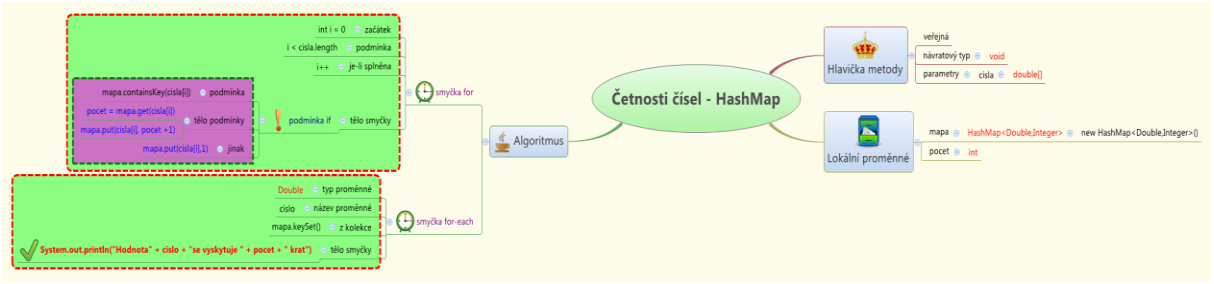
Základní informace:

Pohlaví: Muž

Věk: 23 let

Testovaná osoba nemá předchozí zkušenosti s tvorbou myšlenkových map.

1) Čas potřebný pro osvojení metodiky: 17 minut
2) Pro převod myšlenkové mapy na zdrojový kód byla použita následující myšlenková mapa.



obr.: 41 Četnosti čísel - HashMap

Výsledný kód:

```
public void CetnostiCisel(double[] cisla){
HashMap<Double,Integer> mapa = new HashMap<Double,Integer>();
int pocet = 0;
for(int i = 0; i < cisla.length;i++){
     if(mapa.containsKey(cisla[i])){
          pocet = mapa.get(cisla[i]);
     mapa.put(cisla[i], pocet + 1);
     }else{
          mapa.put(cisla[i],1);
     }
}
for(Double cislo : mapa.keySet()){
     System.out.println("Hodnota " + cislo + " se vyskytuje " +
     pocet + " krát.");
}
}
```

Při převodu na zdrojový kód se vyskytla chyba. Tato chyba byla způsobená patrně nepozorností.

3) Při překreslení zdrojového kódu do myšlenkové mapy byl použit následující kód:

```
public void PridejNaZacatek(Prvek prvek){
     Prvek prvek;
     this.prvek = prvek;
     if(prvek == null){
          return;
```

```
        }
        if(hlavicka == null){
              hlavicka = prvek;
        }else{
        prvek.setDalsi(hlavicka);
        hlavicka = prvek;
        }
        pocet++;
}
```

Výsledná myšlenková mapa:

obr.: 42 Výsledná mapa O8 - Přidej prvek na začátek

Při zakreslování do myšlenkové mapy nevznikla žádná chyba.

7 Porovnání software

V této kapitole jsou popsány programy pro tvorbu myšlenkových map XMind, FreeMind, IMindMap a Mind42. Programy jsou nabízeny v několika verzích a pro testování je vždy použita free verze. V každém z těchto programů je vytvořena myšlenková mapa na stejné téma, pro názornou ukázku.

Pro vyhodnocení jsou stanovena kritéria, na která bude kladen důraz při testování:

1. Hlavní téma musí být vždy uprostřed
2. Umístění větve na libovolnou pozici
3. Stylování textu
4. Používání piktogramů
5. Obarvení jednotlivých větví
6. Ohraničení části, nebo celé větve
7. Spolupráce více lidí po síti
8. Sdílení dat

7.1 XMind

XMind je kvalitní a spolehlivý nástroj pro vytváření myšlenkových map. Svojí funkčností je vhodný nejen pro jednotlivé uživatele, ale také pro firmy. Nabízí funkce pro sdílení map, vzdálenou spolupráci nebo organizaci pracovních týmů.

XMind je nabízen ve třech základních formách - Business, Pro a Free. Jeho verze jsou korektně spustitelné pod operačními systémy Windows, Mac a Linux. První nabízí řadu nástrojů pro práci v týmu velké společnosti. Přístup k vytvořeným mapám může mít každý, kdo obdrží potřebná práva od vedoucího týmu. Za verzi Business s těmito možnosti se pohybuje cena od 324 Euro za rok na deset licencí. Při větším množství zakoupených licencí jsou k dispozici slevy. Za verzi pro je cena 40 Euro ročně za licenci. Školy a neziskové organizace mají k dispozici slevy. Oproti Free verzi umožňuje především rozsáhlejší možnosti v oblasti exportu map a jejich sdílení online.

7.1.1 XMind Free

XMind free je ke stažení zdarma na adrese: http://www.xmind.net/download/win/. Pro většinu uživatelů je dostačující tato verze, která bude i dále testována. Aplikace nabízí tvorbu myšlenkových map a dalších diagramů. Je především zaměřena na vzhled mapy, takže

umožňuje například nastavení vzhledu a barvy uzlů, obrázky a tvary čar. Výsledek působí dobrým dojmem, proto se XMind využívá pro obchodní či prezentační úkoly.

Výhodou programu je bez potíží možnost pracovat s více hlavními uzli. Značná část úkonů lze vykonat metodou *táhni a pusť*, proto i když program poskytuje řadu funkcí, je jeho ovládání velmi jednoduché. Pro rychlé vyhledávání v rozlehlých mapách je užitečná funkce na zobrazení stromové struktury mapy.

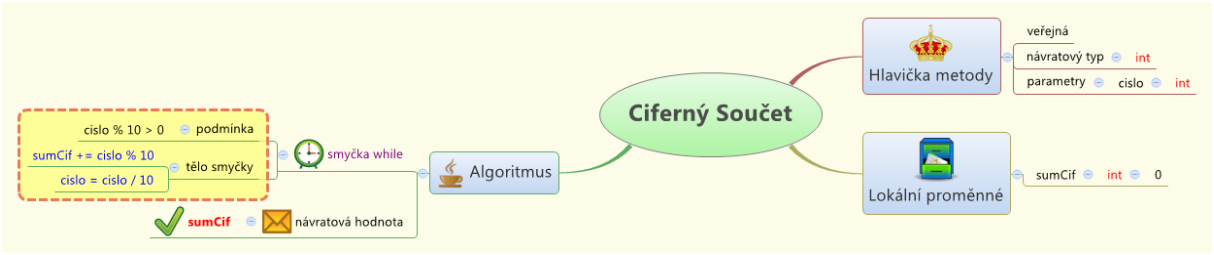
XMind umožňuje vkládání vztahů mezi objekty a jejich popis. Tematicky spojené uzly je možné umístit do speciálního boxu, který je graficky oddělí od zbytku mapy. Na dobré úrovni je funkce klávesových zkratek a práce na více myšlenkových mapách současně.

7.1.2 XMind Share

Nepostradatelnou funkcí, které XMind nabízí, je služba Share. Prostřednictvím této služby je možné nahrávat mapy na síť a sdílet je s ostatními. Free verze umožňuje nahrát mapu pouze jako volně dostupnou. Pro lepší zabezpečení je nutné zakoupit některou z rozšířených verzí. Nahrávání je velmi jednoduché, provádí se stiskem jednoho tlačítko.[26]

7.1.3 Příklad

Napiš veřejnou metodu, která vrátí ciferný součet zadaného čísla. Metoda má jeden parametr typu int.



obr.: 43 Ciferný součet - XMind

7.2 iMindMap

IMindMap je vyvíjen Tony Buzanem autorem konceptu myšlenkových map. Prostředí programu je velmi podobné MS Office, což je pro řadu uživatelů značná výhoda.

Po spuštění umožňuje nastavení řady šablon. Důraz při vytváření je kladen na organickou strukturu (žádné rovné čáry, obdélníky). Text je umísťován přímo na vytvářené křivky, tím je zlepšena přehlednost a dynamika výsledné mapy. Je vhodná pro tvorbu a ilustraci výukových materiálů.

[26] Xmind: myšlenkové mapy také pro business. ČERNÝ, Michal. *Myšlenkové mapy* [online]. Vyd. 1. 2011, 26.7.2011 [cit. 2013-02-13]. Dostupné z: http://www.myslenkove-mapy.cz/myslenkove-mapy/tvorba-myslenkovych-map/xmind-myslenkove-mapy-take-pro-business/

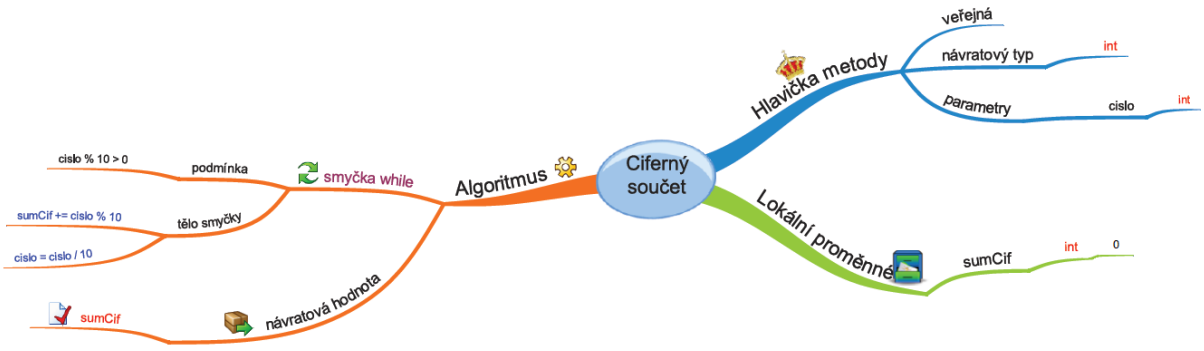
Vytvořené mapy lze exportovat do řady standardních formátů. Jednou z možností je export do 3D map. Tyto mapy jsou ovšem velmi nepřehledná a po grafické stránce také strádají. Na druhou stanu program obsahuje *prezentační mód*. Umožňuje vytvořit mapu a použít jí jako prezentaci podobnou PowerPointu.

IMindMap podporuje i kontrolu pravopisu. Obsahuje řadu jazykových mutací, vyjímaje českého jazyka. Umožňuje snadné upravování vzhledu a tvaru jednotlivých větví. Snaha podpořit kreativitu uživatele.

IMindMap je možné spustit pod operačními systémy Mac a Windows. Je nabízen ve verzích Basic, Home & Student a Ultimate. Byla testována free verze, která obsahuje sedmidenní zkušební verzi Ultimate. Cena licence u verze Home & Student se pohybuje od $42.50 do $85.00 na uživatele a je pouze pro nekomerční použití. Obsahuje rozšířené používání stylů, vkládání zvukových poznámek, větší možnosti exportu a správu vytvořených map. Od této verze je již možné spravovat myšlenkové mapy i na chytrých mobilních telefonech. Verze Ultimate je tvořena pro komerční využití a její cena se pohybuje od $128.20 do $256.00 na uživatele. Cena je určena v závislosti na množství zakoupených licencí. Tato verze obsahuje kompletní sadu funkcí, plnou integraci s Microsoft office a OpenOffice. Dále nástroje pro řízení projektů a vývojových diagramů.[27]

7.2.1 Příklad

Napiš veřejnou metodu, která vrátí ciferný součet zadaného čísla. Metoda má jeden parametr typu int.



obr.: 44 Ciferný součet - iMindMap

7.3 FreeMind

FreeMind je jeden z nejpoužívanějších programů pro tvorbu myšlenkových map. Je napsaný v Javě, proto pro jeho spuštění je nutné mít nainstalovaný Java Runtime. Navíc

[27] iMindMap: první zkušenosti s iMindMap 6. ČERNÝ, Michal. *Myšlenkové mapy* [online]. Vyd. 1. 2012, 18.9.2012 [cit. 2013-02-13]. Dostupné z: http://www.myslenkove-mapy.cz/myslenkove-mapy/tvorba-myslenkovych-map/prvni-zkusenosti-s-imindmap-6/

obsahuje i českou lokalizaci. Jak již název napovídá je tento software dostupný zdarma. Využívá i výhod Javy, tudíž je nezávislý na operačním systému a lze ho spustit i na chytrých mobilních telefonech.

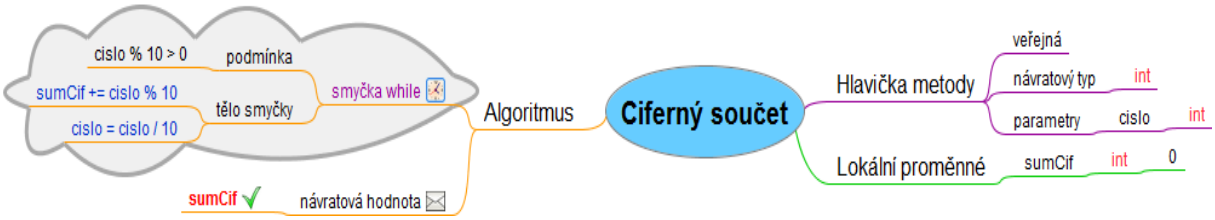
FreeMind je navržen jako velmi intuitivní a jednoduchý. Přidanou funkčnost tu má pravé tlačítko myši. Umožňuje lehce přidávat uzly, editovat atributy nebo popisky. Lze takto i vytvářet vztahy mezi uzly. Jeho základní výhodou je, že organizuje uzly na stránce, aby nedocházelo k jejich překrývání a výsledná mapa byla co nejpřehlednější. Pro rozsáhlé mapy je zde funkce skrýt libovolné uzly. Pro rychlejší ovládání je možnost využití předdefinovaných klávesových zkratek. Nové klávesové zkratky lze definovat při běhu programu.

Jednotlivé uzly a atributy lze graficky upravovat. Možnost přidávat grafické piktogramy k jednotlivým uzlům. Tematicky spojené uzly je možné umístit do speciálního boxu, který je graficky oddělí od zbytku mapy nebo vkládání vztahů mezi objekty.

Grafická podoba je zaměřena na jednoduchost. Nelze ji proto srovnávat s testovanými programy iMindMap a XMind. Kvůli této jednoduchosti je možné bez problémů operovat s velkým množstvím jednotlivých atributů. Vytvořené mapy lze exportovat do řady standardních formátů. Mezi nestandardní výstupy patří možnost převedení do Java appletu, Flashe nebo XHTML. Značnou nevýhodou je, že FreeMind nemá implementovanou funkci pro sdílení vytvořených map.[28]

7.3.1 Příklad

Napiš veřejnou metodu, která vrátí ciferný součet zadaného čísla. Metoda má jeden parametr typu int.



obr.: 45 Ciferný součet - FreeMind

[28] FreeMind: myšlenkové mapy na počítači s FreeMind. ČERNÝ, Michal. *Myšlenkové mapy* [online]. Vyd. 1. 2011, 27.6.2012 [cit. 2013-02-13]. Dostupné z: http://www.myslenkove-mapy.cz/myslenkove-mapy/tvorba-myslenkovych-map/myslenkove-mapy-s-freemind/

7.4 Mind42

Mind42 je online aplikace pro tvorbu myšlenkových map. Je možné ji použít i pro uložení záložek nebo seznamu úkolů. Aplikace je dostupná přes všechny webové prohlížeče a není závislá na platformě.

Mind42 vyžaduje jednoduchou registraci a je nabízen zcela zdarma a bez omezení. Nevýhodou je, že neobsahuje českou lokalizaci. Aplikace je navržena jako velmi jednoduchá a intuitivní. Pro rychlejší ovládání je možnost využití předdefinovaných klávesových zkratek.

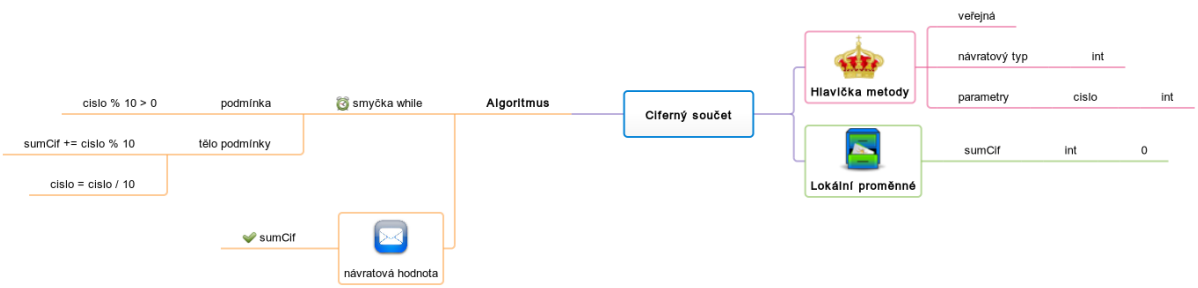
Mind42 organizuje uzly na stránce, aby nedocházelo k jejich překrývání a výsledná mapa byla co nejpřehlednější. Pro rozsáhlé mapy je zde možnost skrýt libovolné uzly. Grafická podoba programu je velmi prostá, pro obarvení jednotlivých uzlů je zde pouze několik základních barev. Možnost přidávat grafické piktogramy k jednotlivým uzlům je zde zahrnuta, ale nedosahuje estetické úrovně ostatních testovacích programů. Užitečnou funkcí, kterou Mind42 obsahuje, jsou *Todo listy*. K jednotlivým úkolům přidělují informace o datu, prioritě a procentech splnění.

Největší výhodou této aplikace je možnost týmové spolupráce několika členů v reálném čase. Pro připojení dalšího člena je využívána funkce Collaboration, která odešle email na zadanou adresu s vygenerovanou URL hodnotou. Při vytváření mapy více členy týmu, nedochází k technickým problémům aplikace. Změny jsou ostatním členům ukázány vždy po dokončení akce (editace atributu, vytvoření nového uzlu, přidání grafického prvku).

Vytvořenou myšlenkovou mapu lze volně sdílet nebo přiřadit práva pro správu a zobrazení jednotlivým členům. Možnost exportu obsahuje většinu standardních formátů. Rozšířením je možnost importu i exportu do programu FreeMind.[29]

7.4.1 Příklad

Napiš veřejnou metodu, která vrátí ciferný součet zadaného čísla. Metoda má jeden parametr typu int.



obr.: 46 Ciferný součet - Mind42

[29] JanKoWeb: Počítače a IT. *JanKoWeb* [online]. 24.08.2012 [cit. 2013-03-04]. Dostupné z: http://jankoweb.moxo.cz/blog/pocitace-a-it/mind42-com-myslenkove-mapy-a-todo-listy-online/

7.5 Vyhodnocení

X obsahuje

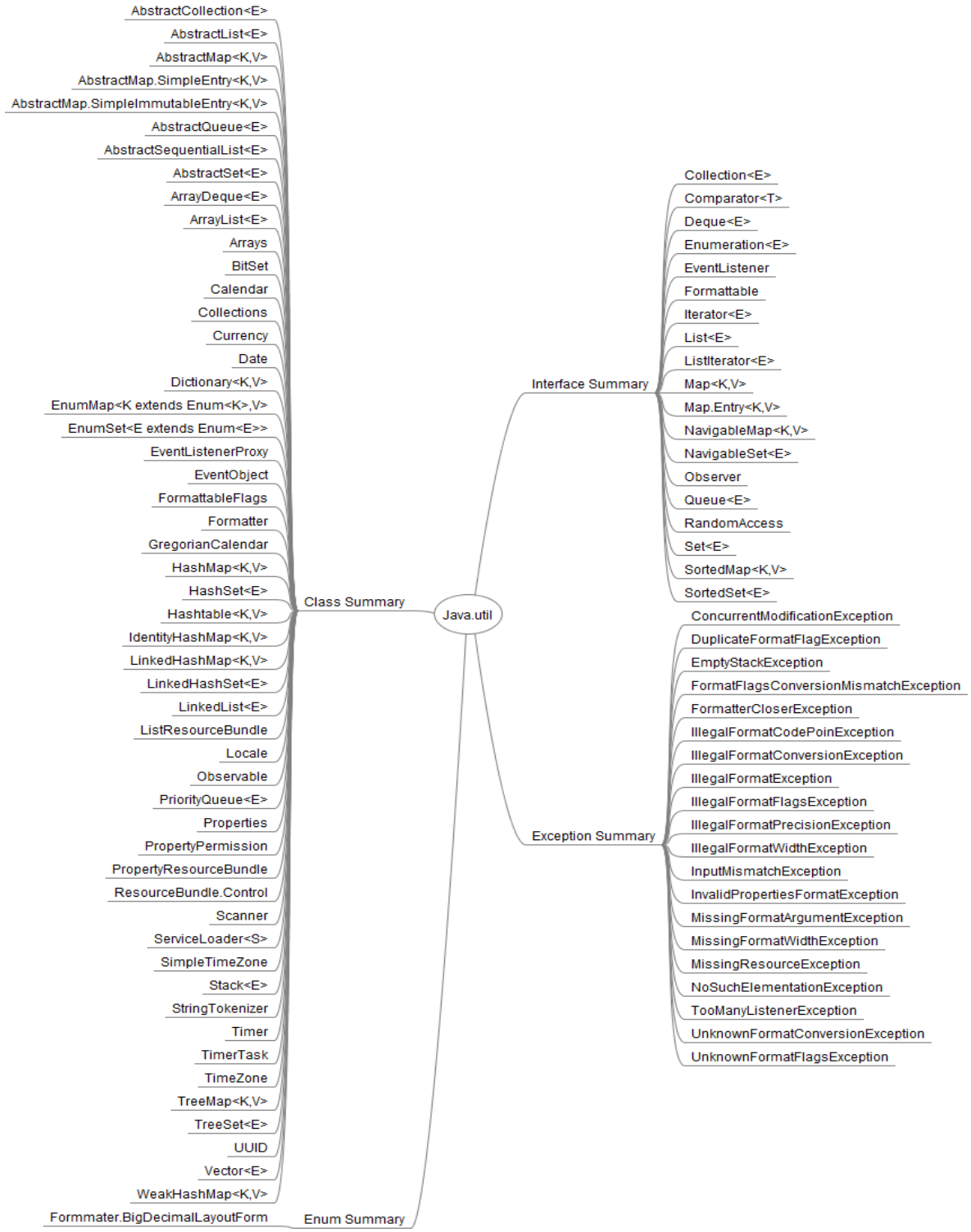
\- neobsahuje

	XMind	**IMindMap**	**FreeMind**	**Mind42**
Hlavní téma uprostřed	X	X	X	X
Větev na libovolné pozici	-	X	-	-
Stylování textu	X	X	X	-
Piktogramy	X	X	X	X
Obarvení větví	X	X	X	X
Ohraničení větve	X	X	X	-
Spolupráce více lidí	-	-	-	X
Sdílení dat	X	X	-	X

obr.: 47 Vyhodnocení programů

8 Knihovny

Java využívá velké množství knihoven (balíčků), které uživatelům napomáhají při práci. Jednou z často využívaných knihoven je Java.util. Proto byla vybrána pro překreslení do pojmové mapy. Výsledná mapa je vhodná pro představu, jaké možnosti tato knihovna nabízí.



obr.: 48 Java.util

8.1 ArrayList<E>

Třída ArrayList<E> je obsažena v balíčku Java.util. V hierarchii balíčků a tříd je zařazena na následující pozici:

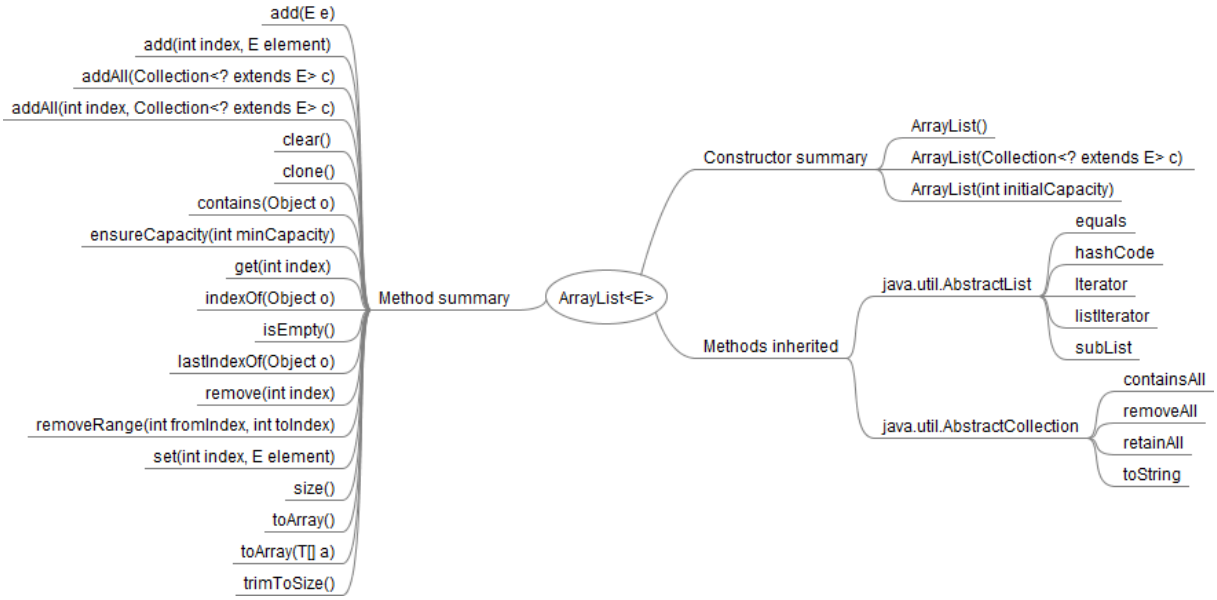
Java.lang.Object → java.util.AbstractCollection<E> → java.util.Abstract<E> → java.util.ArrayList<E>

Tato hierarchie značí, od jakých balíčků nebo tříd byla daná třída odvozena a koho je potomkem.

Třída ArrayList<E> implementuje následující rozhraní:

- Serializable
- Cloneable
- Iterable<E>
- Collection<E>
- List<E>
- RandomAccess

Znázornění třídy ArrayList<E> v pojmové mapě:



obr.: 49 ArrayList<E>

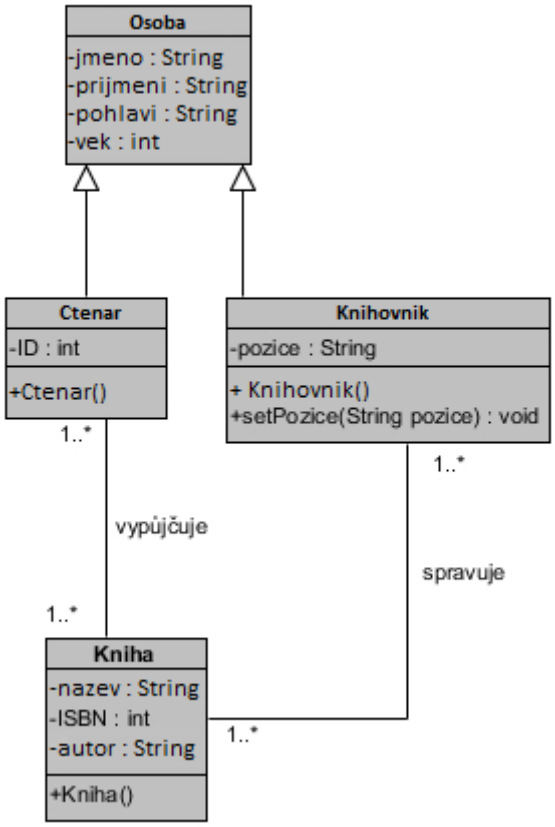
9 UML diagramy v myšlenkových mapách

Tato kapitola zjišťuje, zda vybrané UML diagramy, lze zakreslit do myšlenkových map a zda to přinese nějaké výhody.

9.1 Diagram tříd (Class diagram)

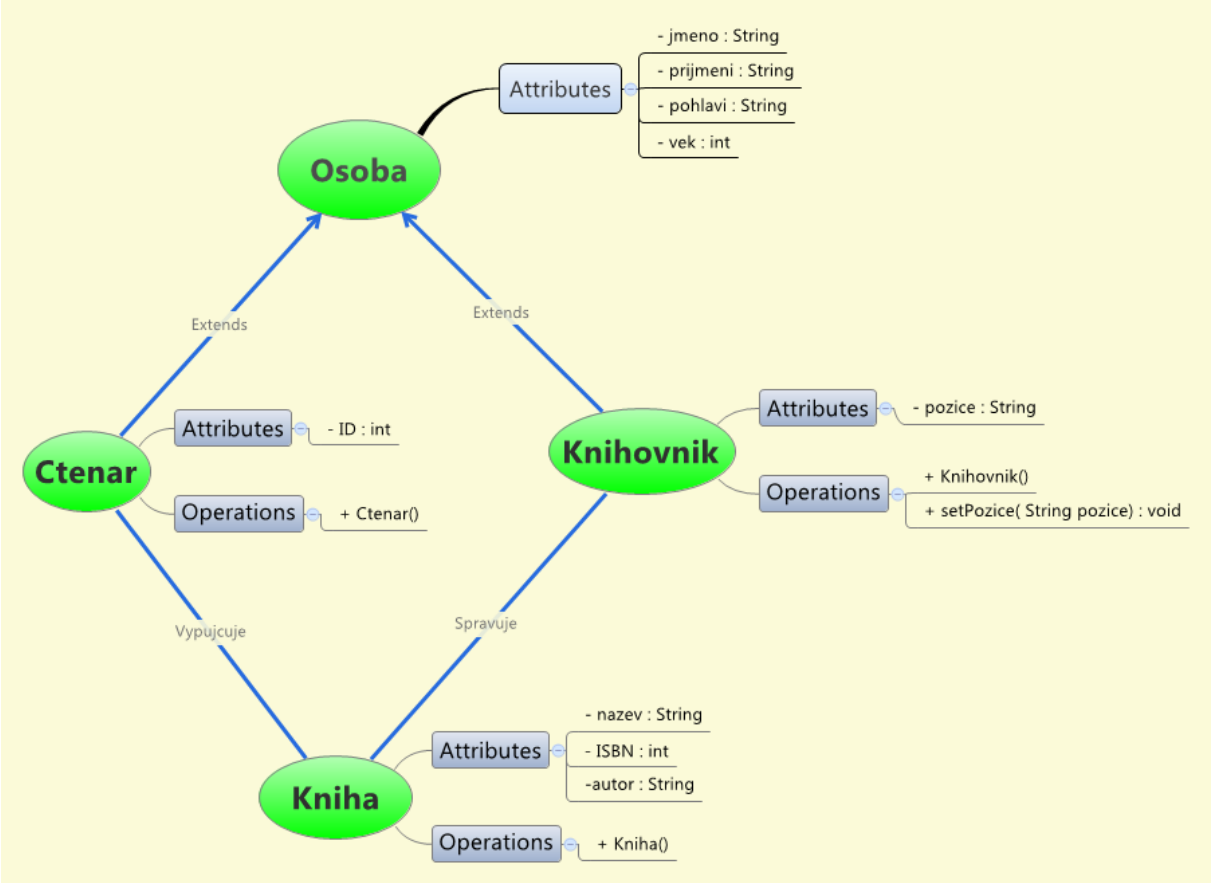
Prvním diagramem pro převod do myšlenkové mapy je diagram tříd (Class diagram). Ten znázorňuje existenci tříd a jejich vztahů v logickém pohledu na systém.

Byl vybrán jednoduchý návrh, který by mohl být dále rozvíjet, však pro toto testování je dostačující.



obr.: 50 Diagram tříd

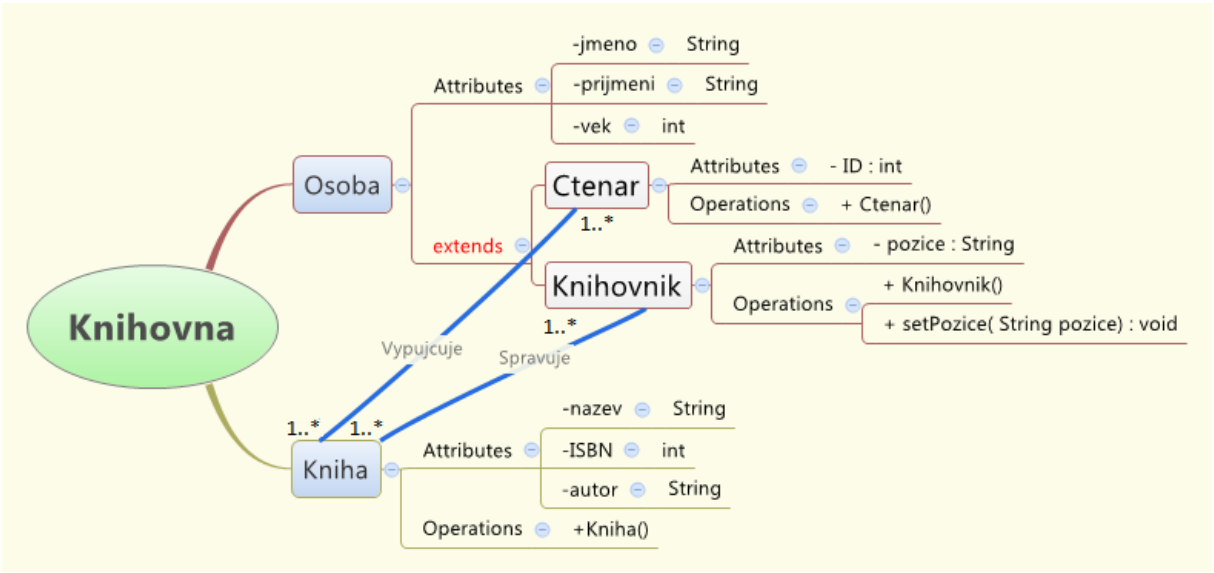
Pro zakreslení do myšlenkové mapy byl vybrán volně dostupný software XMind.



obr.: 51 Diagram tříd v myšlenkové mapě

Po převedení je patrné, že se již nejedná o myšlenkovou mapu, protože nerespektuje základní principy. Myšlenková mapa je vizuálním nástrojem pro paprskovité myšlení, tudíž musí vycházet z centrálního prvku. Musel by být zvolen centrální prvek, který by toto téma spojoval. Jednotlivé asociace by měly vycházet z prvků. Rozrůstající návrh by byl dále stále více monotónní. Jedná se o nebezpečné oblasti při navrhování myšlenkových map (kapitola 7.5).

Po provedení oprav, lze již tento návrh považovat za myšlenkovou mapu diagramu tříd. Byl přidán centrální prvek a jednotlivé třídy byly uspořádány do hierarchické struktury.

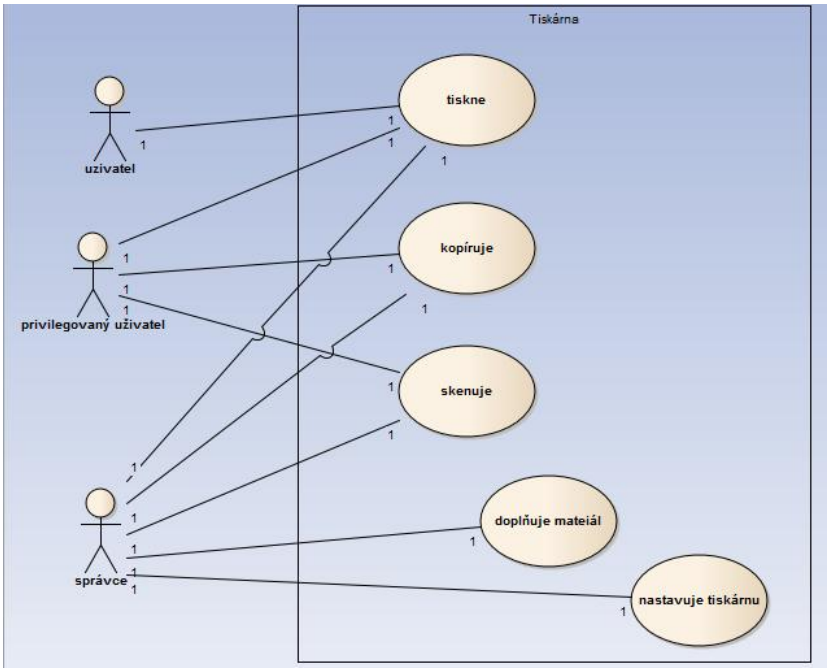


obr.: 52 myšlenková mapa - knihovna

Tento návrh přináší výhody pokud, je otevřen v programu, ve kterém byl vytvořen. Umožňuje skrytí jednotlivých větví diagramu. To je značná výhoda při rozsáhlejším návrhu. Dobrou pomůckou je také přidávání vlastních popisků k jednotlivým prvkům.

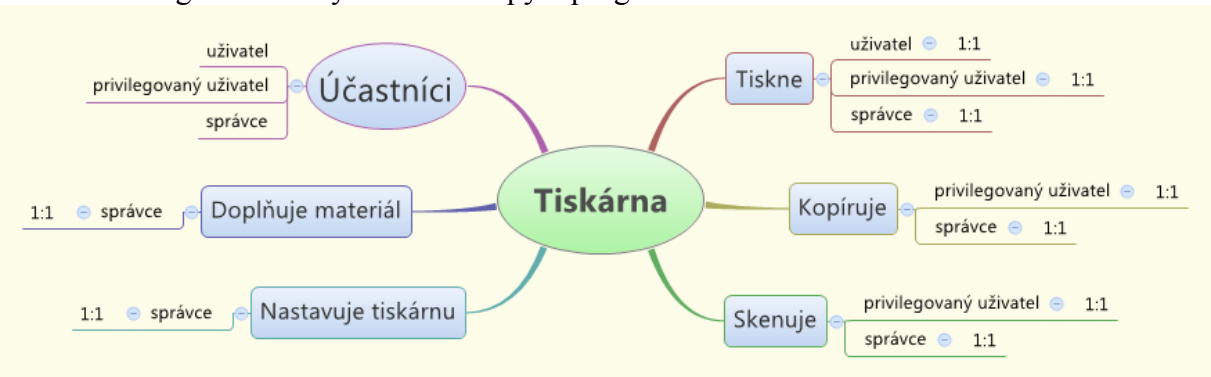
9.2 *Diagram případů užití (Use case diagram)*

Diagram případů užití umožňuje, vnější pohled na systém. Dále zobrazuje vztahy mezi účastníky a jednotlivými případy užití. Pro převedení do myšlenkové mapy byl vybrán jednoduchý návrh, který by mohl být dále rozvíjet, však pro toto testování je dostačující.



obr.: 53 Use case - knihovna

Převedení diagramu do myšlenkové mapy v programu XMind.



obr.: 54 Use case v myšlenkové mapě - tiskárna

Diagram případů užití lze lehce převést do myšlenkové mapy. Nebyly nalezeny žádné závažné problémy při vytváření myšlenkové mapy. Výsledný návrh zůstává přehledný a v některých případech může být i více vhodný. Zvláště pokud nechceme používat více programů nebo chceme zachovat stejný styl vzhledu.

10 Závěr

Byla navrhnuta metodika jak zpracovávat příklady z programování i algoritmizace. Následně byla vytvořena sbírka příkladů na toto téma, která byla doplněna o vypracovaná ukázková videa, která jsou umístěna v příloze bakalářské práce. Navrhnutá metodika byla testována na 8 osobách. Dalším z cílů byly knihovny, kde byl rozebrán a překreslen do pojmové mapy balíček Java.util. a třída z tohoto balíčku ArrayList<E>. Poté byly porovnány a vyhodnoceny bezplatné programy, pro tvorbu myšlenkových map. Jako poslední byly UML diagramy (Use Case a Class diagram) zakresleny do myšlenkových map a byl vyhodnocen jejich možný přínos v této podobě.

11 Použitá literatura

BUZAN, Tony a Barry BUZAN. *Myšlenkové mapy: probuďte svou kreativitu, zlepšete svou paměť, změňte svůj život*. Vyd. 1. Brno: Computer Press, 2011, 213 s. ISBN 978-80-251-2910-4.

PECINOVSKÝ, Rudolf. *Myslíme objektově v jazyku Java 5.0*. Vyd. 1. Praha: Grada, 2004, 601 s. ISBN 80-247-0941-4.

PERRY, Paul J. *JAVA - tvorba dokonalých WWW stránek*. 1. vyd. Praha: Grada Publishing, 1996, 327 s. ISBN 80-716-9415-0.

SEDGEWICK, Robert. *Algoritmy v C*. Překlad Jiří Gree. Praha: SoftPress, 2003, 688 s. ISBN 80-864-9756-9.

ŠEŠERA, L'ubor. *Datové modelování v příkladech*. 1. vyd. Praha: Grada, 2001, 151 s. ISBN 80-247-0049-2.