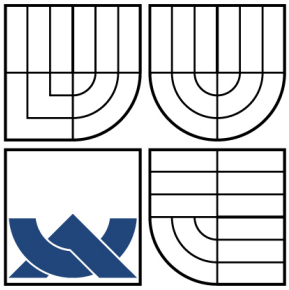
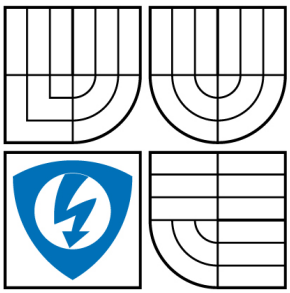


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE PRO ZOBRAZENÍ MAPOVÝCH PODKLADŮ VE WEBOVÉM PROSTŘEDÍ

APPLICATION FOR DISPLAYING CARTOGRAPHIC DATA IN WEB ENVIRONMENT

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

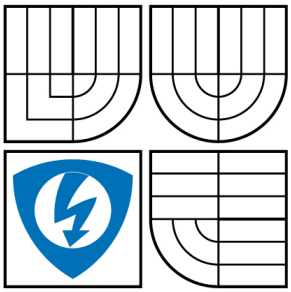
AUTOR PRÁCE — ADAM DUCHOVIČ

AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN KOUTNÝ

SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Duchovič Adam	***ID:***	78499
Ročník:	3	***Akademický rok:***	2007/2008

NÁZEV TÉMATU:

Aplikace pro zobrazení mapových podkladů ve webovém prostředí

POKYNY PRO VYPRACOVÁNÍ:

Podrobně se seznamte s možnostmi multiplatformního programování pro Internet. Ve Vámi vybraném vývojovém nástroji realizujte aplikaci pro zobrazování mapových podkladů. Aplikaci implementujte do webového prostředí a zabezpečte, aby bylo možné mapové podklady snáze implementovat. Při realizaci se snažte pracovat modulárně tak, aby bylo možné aplikaci v budoucnu jednoduše rozšiřovat.

DOPORUČENÁ LITERATURA:

[1] ZAKHOUR, Sharon, et al. Java 6 : Výukový kurz. 2007. 536 s. ISBN 978-80-251-1575-6.
[2] HEROUT, Pavel. Učebnice jazyka Java. 2006. 352 s. ISBN 80-7232-115-3.

Termín zadání:	11.2.2008	***Termín odevzdání:***	4.6.2008

Vedoucí práce: Ing. Martin Koutný

prof. Ing. Kamil Vrba, CSc.
předseda oborové rady

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Adam Duchovič
Bytem: Starohorská 6151/6, 97411, Banská Bystrica
Narozen/a (datum a místo): 20.2.1986, Banská Bystrica

(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 244/53, 60200 Brno 2
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Kamil Vrba, CSc.

(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☐ diplomová práce
   - ☑ bakalářská práce

   jiná práce, jejíž druh je specifikován jako .....

(dále jen VŠKP nebo dílo)

Název VŠKP: Aplikace pro zobrazení mapových podkladů ve webovém prostředí
Vedoucí/školitel VŠKP: Ing. Martin Koutný
Ústav: Ústav telekomunikací
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

- ☑ tištěné formě - počet exemplářů 1
- ☑ elektronické formě - počet exemplářů 1

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.

4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☑ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Vysoké učení technické v Brně — Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

POPISNÝ SOUBOR ZÁVĚREČNÉ PRÁCE

Autor: Adam Duchovič

Název závěrečné práce: Aplikace pro zobrazení mapových podkladů ve webovém prostředí

Název závěrečné práce ENG: Application for displaying cartographic data in web environment

Anotace závěrečné práce: Predkladaná práca sa zaoberá návrhom aplikácie na zobrazovanie mapových podkladov, ako aj jej implementáciou do webového prostredia. Ciežom bolo vytvorenie aplikácie nezávislej na platforme. V úvode práca zoznamuje s problematikou multiplatformového programovania, v ďalších kapitolách predstavuje programovací jazyk Java a zoznamuje s použitým vývojovým prostredím. Následne sa práca venuje rozdeleniu riešenia danej problematiky na dve možnosti a návrhu zobrazovania mapových podkladov. Ďalej je vysvetlený postup programovania požadovanej aplikácie s jej následnou implementáciou do webového prostredia. Pri realizácii navrhovanej aplikácie bolo prihliadané na jej modularitu. V závere práce je zhrnutý postup riešenia a porovnanie vhodnosti použitých metód.

Anotace závěrečné práce ENG: This thesis deals with designing application for displaying cartographic data and its implementation to the web environment. The goal was to design an application which will be independent on platform. Introduction provides basic information about the Java programming language, next chapter introduces to the applied developing environment. Then the thesis deals with an option of dividing the solution on two possibilities and the concept of displaying cartographic data. Further is described design procedure of programming application with its implementation to the web environment. Modularity of this application was ensured during its realization. In the conclusion the process of developing application is resumed and both of the solutions are compared.

Klíčová slova: multiplatformové programovanie, aplikácia, Java, aplet, servlet,

Vysoké učení technické v Brně | Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

POPISNÝ SOUBOR ZÁVĚREČNÉ PRÁCE

	mapové podklady, webové prostredie, Internet, HTML
Klíčová slova ENG:	multi-platform programming, application, Java, applet, servlet, cartographic data, web environment, Internet, HTML
Typ závěrečné práce:	bakalářská práce
Datový formát elektronické verze:	pdf
Jazyk závěrečné práce:	slovenština
Přidělovaný titul:	Bc.
Vedoucí závěrečné práce:	Ing. Martin Koutný
Škola:	Vysoké učení technické v Brně
Fakulta:	Fakulta elektrotechniky a komunikačních technologií
Ústav / ateliér:	Ústav telekomunikací
Studijní program:	Elektrotechnika, elektronika, komunikační a řídicí technika
Studijní obor:	Teleinformatika

Prehlásenie

Prehlasujem, že svoju bakalársku prácu na tému Aplikace pro zobrazení mapových podkladů ve webovém prostředí som vypracoval samostatne pod vedením vedúceho bakalárskej práce a s použitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a som si plne vedomý následkov porušení ustanovení § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovení § 152 trestného zákona č. 140/1961 Sb.

V Brne dňa 4.6.2008 .....

podpis autora

Poďakovanie

Ďakujem vedúcemu bakalárskej práce Ing. Martinovi Koutnému za užitočnú odbornú, metodickú a organizačnú pomoc, usmernenia, ochotu a cenné rady pri spracovávaní mojej bakalárskej práce.

V Brne dňa 4.6.2008 .....

podpis autora

OBSAH

Zoznam obrázkov

Úvod

Úvod bakalárskej práce stručne oboznamuje s problematikou multiplatformového programovania, ktoré je obsahom tejto práce.

Multiplatformové programovanie znamená programovať pre viaceré počítačové platformy.

Počítačovou platformou sa rozumie hardvérové, ale tak isto aj softvérové prostredie, ktoré umožňujú spúšťanie a činnosť programov. Pod bežné platformy zaraďujeme operačný systém, počítačovú architektúru, programovacie jazyky vrátane ich knižníc.

Pojem multiplatformový softvér však neznamená, že daný program musí fungovať pod všetkými platformami. Na to, aby sme mohli nazvať program multiplatformovým, stačí ak ho bude možné spustiť aspoň na dvoch rôznych platformách. Známym multiplatformovým softvérom sú napríklad internetové prehliadače Opera, Mozilla, Firefox.

Veľmi známym multiplatformovým softvérom sú napríklad aplikácie napísané v programovacom jazyku Java alebo v jazyku .NET. Práve programovací jazyk Java sa ukázal ako vhodný na vytvorenie aplikácie na zobrazovanie mapových podkladov, z hľadiska použitia na viaceré platformy.

Po vytvorení aplikácie sa musíme zamýšľať nad jej umiestnením, funkčnosťou a prístupe užívateľov k požadovaným informáciám. V práci sú spomenuté dve možnosti tvorby, realizácie a implementácie požadovanej aplikácie.

V oblasti zobrazovania mapových podkladov na Internete je v popredí doména Googlemaps, v našich zemepisných šírkach sú to napríklad domény Mapy.cz, Mapy.sk

1. Možnosti multiplatformového programovania

Pri multiplatformovom programovaní pre Internet je viacero možností, ktoré sa dajú uplatniť. Existuje viacero jazykov ako napríklad .NET, Flash, Java, PHP a pod. Pri návrhu požadovanej aplikácie som uprednostnil možnosť programovania v jazyku Java.

1.1 Technológia JAVA

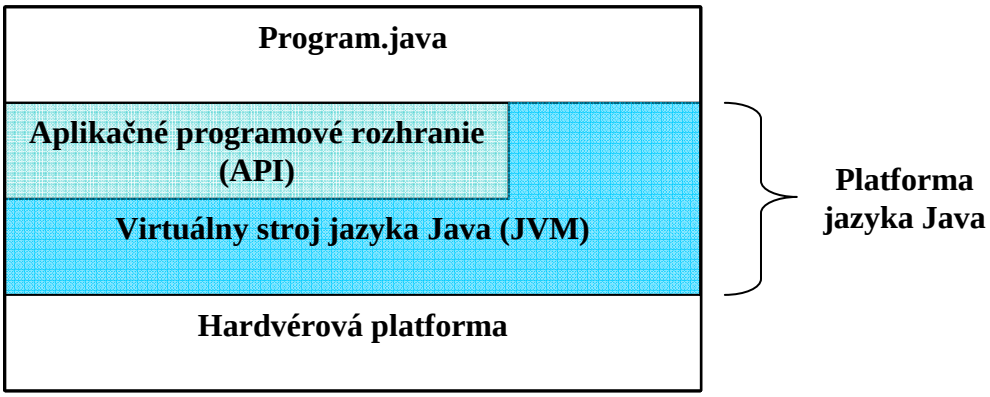
Tento jazyk bol od roku 1991 vyvíjaný na princípoch programovacích jazykov C a C++ firmou Sun Microsystems (ďalej len „Sun“) [1], [2]. Názov tohto jazyka je odvodený od amerického slangového výrazu pre kávu, čo sa odzrkadlilo aj na grafickej ikone tohto programu – šálke kávy. S narastajúcou dôležitosťou celosvetovej počítačovej siete, World Wide Web (ďalej len „WWW“) si firma Sun uvedomila možnosť využiť Javu na programovanie aplikácií pre WWW. V roku 1995 bola na konferencii firmou Sun oficiálne predstavená pôvodná vývojová sada Java Development Kit (ďalej len „JDK“). Od tejto doby sa Java zlepšuje a zdokonaľuje. Nedávno bola uvoľnená verzia 6 vydania Java Platform Standard Edition (Java SE).

V Jave sa zdrojový kód najskôr ukladá do textových súborov s príponou .java, za pomoci kompilátora javac sú tieto súbory skompilované do súborov .class. Súbor .class obsahuje bajtový kód (bytecode), čo je strojový jazyk Java Virtual Machine (ďalej len „JVM“). Práve táto vlastnosť Javy je podstatná pre multiplatformové programovanie, keďže bajtkód je zrozumiteľný pre všetky počítače, teda je nezávislý na operačnom systéme. Program napísaný v Jave je teda možné prenášať medzi rôznymi počítačovými platformami a operačnými systémami.

Je dôležité vedieť, že Java samotná nie je len programovací jazyk, ale zahrňuje v sebe aj platformu. Java ako platforma sa od ostatných platforiem (kombinácia operačného systému a hardvéru), líši tým, že je to výhradne softvérová platforma, ktorá funguje nad inými hardvérovo závislými platformami.

Platforma zahŕňa dva komponenty:

- Virtuálny stroj jazyka Java (JVM),
- Aplikačné programové rozhranie Java, Application Programming Interface (ďalej len „API“).



Obr. 1: Nezávislosť Javy na hardvérovej platforme

V API sú umiestnené hotové zbierky softvérových komponentov, triedené do knižníc so súvisiacimi triedami a rozhraniami. Knižnice sa označujú ako tzv. balíčky (packages) a tie poskytujú mnoho užitočných možností pri programovaní. Ak program využíva metódy z API, nie je súčasťou tohto programu kód použitých metód z API, čo znamená, že napísaný program má pomerne malú veľkosť.

Nezávislosť na platforme spôsobuje, že niektoré programy sú pomalšie ako natívny kód. Natívny kód je termín používaný pre strojový kód, ktorý používa samotný procesor Central processing unit (ďalej len „CPU“). Vďaka pokroku vo vývoji JVM sa však výkon programu zrovnáva s výkonom natívneho kódu, pri zachovanej prenositeľnosti.

V Jave sa dajú tvoriť viaceré druhy programov napr. aplikácie, applety, servlety.

2. Výber programovacieho jazyka

2.1 Objektovo orientované programovanie

Programovací jazyk Java je jazykom vyššej úrovne, na ktorý sa vzťahujú všetky nasledujúce termíny [1]:

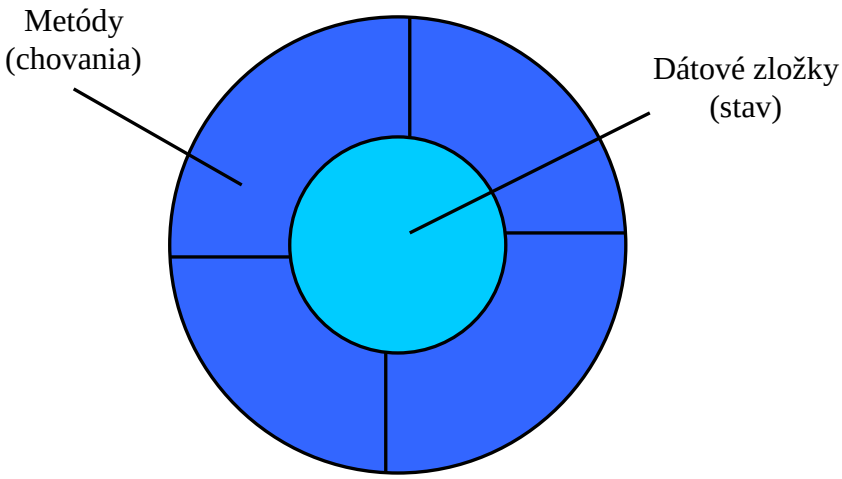
- jednoduchý,
- nezávislý na architektúre,
- objektovo orientovaný,
- prenášateľný,
- vysokovýkonný.

Javu považujeme za jednoduchý jazyk, keďže jednotlivé základné princípy sú ľahko pochopiteľné. To znamená, že aj so základnými vedomosťami a skúsenosťami môžeme vytvoriť efektívny program. Pojmy nezávislý na architektúre a prenášateľný boli vysvetlené vyššie. Vysoká výkonnosť súvisí s tým, že aplikácie vyžadujú väčšie množstvo výpočtového výkonu. Na zrýchlenie sa prepisujú do natívneho strojového kódu.

Java patrí medzi skupinu objektovo orientovaných jazykov. Aby sme pochopili princíp objektového programovania, môžeme softvérové objekty prirovnať objektom reálneho života.

Každý objekt má svoj stav a možnosti svojho chovania. Objekt ukladá stav svojho chovania do dátových zložiek (napr. v jazyku C sú to premenné) a poskytuje chovanie pomocou metód (v niektorých programovacích jazykoch funkcie). Metódy pracujú s vnútorným stavom objektu a slúžia ako primárny mechanizmus na komunikáciu medzi objektmi.

Pokiaľ objekt skrýva svoj vnútorný stav a požaduje, aby všetky interakcie prebiehali pomocou jeho metód, označuje sa to ako zapuzdrenie dát. Jedná sa o základný princíp objektovo orientovaného programovania [1].



Obr. 2: Softvérový objekt

Pri programovaní sa niekedy stretneme s viacerými objektmi rovnakého typu, majú rovnaké parametre akurát iných hodnôt. Môžeme ich zaradiť do jednej triedy, pričom trieda je plán, podľa ktorého vznikajú jednotlivé objekty [1].

V objektovo orientovanom jazyku je dôležitou vlastnosťou tiež dedičnosť. V prípade, ak majú triedy spoločný stav alebo chovanie, sa jedna najvšeobecnejšia trieda stane predkom (tzv. nadtriedou - superclass). V programovacom jazyku Java smie mať každá trieda jedného priameho predka a od každej nadtriedy sa teoreticky môže odvodzovať neobmedzený počet potomkov [1]. Syntax na vytvorenie potomka:

```
Class TriedaPotomka extends Predok  {

     // dátové zložky a metódy definujúce potomka
}
```

Objekty definujú svoju interakciu s okolím pomocou metód. Tieto metódy vytvárajú spojenie objektu s okolím, tvoria teda rozhranie. Rozhranie je tvorené skupinou súvisiacich metód. Môže vyzerať nasledovne :

```
interface  Názov {

     void funkcia1 ();
     void funkcia2 ();
}
```

Ak chceme implementovať rozhranie, musíme to spraviť pomocou kľúčového slova `implements`.

```
class Trieda implements Rozhranie{
//miesto pre naše funkcie
}
```

Rozhranie predstavuje zmluvu medzi triedou a okolím a dodržiavanie tejto zmluvy kontroluje kompilátor pri zhotovení programu. Pokiaľ trieda prehlasuje, že implementuje rozhranie, musí jej zdrojový kód obsahovať všetky metódy definované týmto rozhraním. V opačnom prípade nie je kód úspešne skompilovaný [1].

Pri písaní softvéru v Jave môžeme použiť veľké množstvo tried a rozhraní. Kvôli prehľadnosti sú jednotlivé súvisiace triedy a rozhrania umiestnené v tzv. balíčkoch (packages). Java obsahuje rozsiahlu knižnicu balíčkov. Ako sme sa dozvedeli vyššie, táto knižnica sa volá API.

2.2 Programovací jazyk JAVA

Pred samotným programovaním je dôležité vedieť niekoľko základných informácií o písaní kódu v Jave.

Patria medzi ne:
- spôsob zápisu identifikátorov,
- deklarácia premenných,
- použitie primitívnych dátových typov,
- základné operácie a operátory,
- vetvenie programu a cykly.

Pri písaní identifikátorov je dodržiavaná striktná konvencia [2]:

- ***Triedy a rozhrania*** – identifikátor začína vždy veľkým písmenom a ostatné písmena sú malé. V prípade použitia viacslovného pomenovania, sa každé nasledujúce slovo začína veľkým písmenom napr. String, StringBuffer.
- ***Metódy a premenné*** – identifikátor začína a pokračuje malým písmenom. Začiatok ďalšieho slova sa označuje veľkým písmenom. Napríklad metódy getSize(), start(), premenná pocetPrvkov.

- ***Balíky*** – identifikátor sa skladá iba z malých písmen. V zložených pomenovaniach sú slova oddelené bodkou napr. java.lang.
- ***Konštanty*** – používajú sa iba veľké písmená, vo viacslovných pomenovaniach používame podtržník, napr. PI, MAX_VALUE.

Premenné slúžia na ukladanie informácií a môžu byť viacerých typov (viď nižšie). Delíme ich podľa miesta deklarovania na:

globálne – deklarované na začiatku programu
lokálne – deklarované v metódach či funkciách.

Všeobecná deklarácia premennej vyzerá nasledovne:

dátový_typ názov_premennej [= počiatočná_hodnota].

Dátový typ určuje, aká hodnota môže byť priradená danej premennej. Zároveň určuje operácie, ktoré s premennou môžeme vykonať. Medzi primitívne dátové typy v Jave patria [7]:

Názov typu	**Veľkosť**	**Rozsah hodnôt**
byte	1B	-128 až 127
short	2B	-32 768 až 32 767
int	4B	-2 147 483 648 až 2 147 483 647
long	8B	-9 223 372 036 854 775 808 až 9 223 372 036 854 775 807
float	4B	+/- 1,4E-45 až +/-3,4E+38
double	8B	+/-4,9E-324 až +/-1,7E+308
char	2B	65 536 rôznych znakov
boolean	1b	hodnoty *true* alebo *false*

Tab. 1: Primitívne dátové typy, ich veľkosť a rozsah hodnôt

Medzi základné matematické operácie a operátory patria sčítanie (+), odčítanie (-), násobenie (*), delenie (/), zvyšok po delení (%) atď. Ak chceme inkrementovať alebo dekrementovať hodnoty premenných o jednotku používame operátory (++) a (--). Musíme však dávať pozor na poradie ich zápisu, keďže na tomto poradí závisí aj poradie vykonania operácií:

```
a=b++          //a=b; b=b+1;
a=++b          //a=b+1;b=b+1.
```

Okrem aritmetických operátorov existujú aj tzv. operátory priradenia (viď Tab. 2), ktoré dovoľujú kombinovať aritmetickú operáciu s priradením [7].

Operátor	**Príklad**	**Význam operátora**
+=	a+=b;	súčet, a=a+b;
*=	a*=4;	násobenie, a=a*4
/=	a/=3	delenie, a=a/3
<<=		bitový posun doľava, a=a<<2

Tab. 2: Operátory priradenia a ich význam

Ak chceme v Jave testovať rovnosť použijeme znamienko (==), ak nerovnosť použijeme znamienko (!=). Pri testoch ostrých či neostrých nerovností sa používajú operátory >, <, >=, <=.

K vetveniu programu v Jave používame slovo `if`, za ktorým je uvedená podmienka v zátvorke. Ak nie je splnená podmienka, vykonajú sa príkazy uvedené za slovom `else`, napr.:

```
if (podmienka){
                // naše príkazy1
}
          else{
                //naše príkazy2
}
```

Ak chceme využiť ľubovoľný počet vetvení, môžeme použiť príkaz `switch`. Ten vyhodnotí, ktorý prípad nastal a v závislosti na tom priradí výstup. Užitočným príkazom je `break`, ktorý sa používa na ukončenie alternatívy prepínača. Riadenie toku potom pokračuje prvým príkazom, nasledujúcim za prepínačom:

```
Switch (premenná) {
     case 1: //naše_príkazy1
     case 2: //naše_príkazy2
          break;
}
```

Na vykonanie rovnakých príkazov viackrát po sebe používame cyklus. Ak chceme použiť príkaz presne niekoľkokrát po sebe, použijeme cyklus `for`:

```
for(int i =1;1<100; i++){
     //naše príkazy
}
```

Ak chceme vykonávať príkaz do splnenia určitej podmienky, použijeme cyklus `while`:

```
while(podmienka) ++){
   //naše príkazy
}
```

Ďalším typom cyklu je `do-while`, ktorý povoľuje vykonať príkazy aspoň raz a potom, pokiaľ je nutné, tento príkaz opakovať:

```
do {
     //naše príkazy
} while(podmienka)
```

3. Možnosti riešenia aplikácie

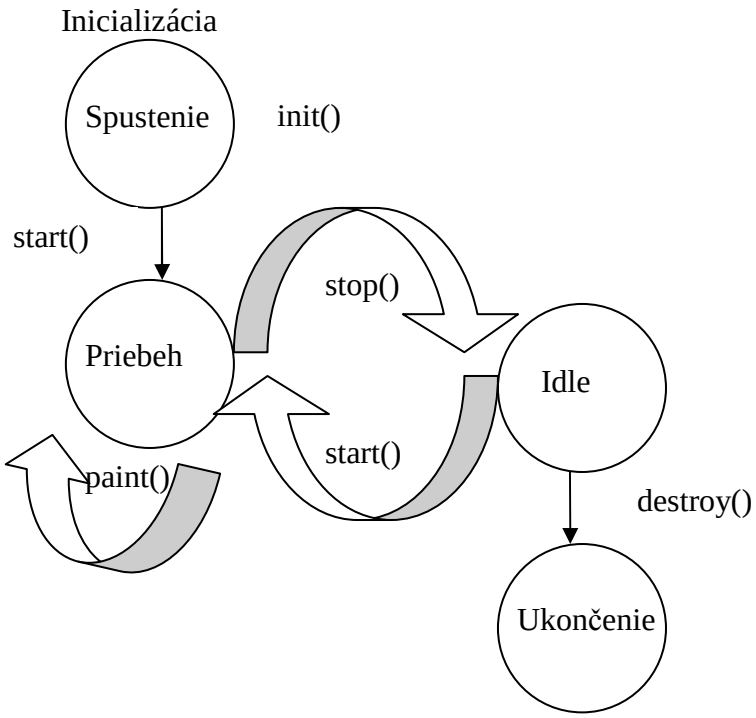
3.1 Technológia apletov

Aplet je špeciálny typ programu Java, ktorý môžeme stiahnuť z Internetu a spustiť v prehliadači s podporou technológie Java. Aplet je obvykle krátky program vložený do webovej stránky a spúšťa sa v kontexte prehliadača. Aplet musí byť podtriedou triedy java.applet.Applet, ktorá poskytuje štandardné rozhranie medzi apletom a prehliadačom [1]. Ak vytvárame Graphic User Interface (ďalej len „GUI“) appletu za pomoci komponentov sady Swing, musíme použiť triedu java.swing.JApplet.

Trieda Applet obsahuje štyri základné metódy, na ktorých je založený každý aplet:

init() – inicializácia apletu pri každom načítaní.
start() – spustí činnosť apletu. K jej volaniu dochádza zakaždým, keď sa užívateľ vráti na stránku s apletom po zobrazení iných stránok.
stop() – zastaví činnosť apletu, je automaticky volaná po každom opustení apletu. Umožňuje zastaviť animácie.
destroy() – urobí záverečné čistenie ako prípravu na vyradenie apletu. Táto metóda je volaná len pri normálnom ukončení prehliadača.

Java aplety predstavujú pravdepodobne najvšeobecnejšiu a najvýkonnejšiu technológiu na strane klienta. Avšak nevýhodou apletov je nutnosť predinštalovať doplnok aplikácie alebo samotný aplet stiahnuť. Čím viac závisia webové servery na týchto technológiách, tým rozsiahlejší musia byť weboví klienti, a tým viac sa Internet distribúciou týchto klientov zahlcuje [5].

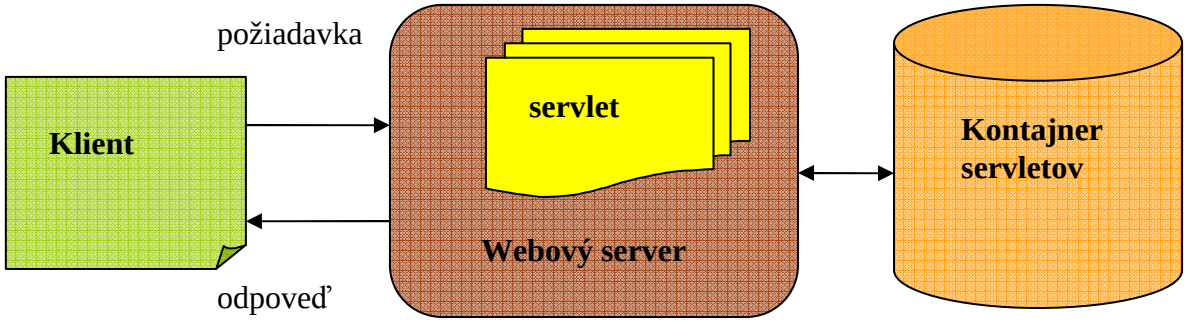


Obr. 3: Životný cyklus apletu

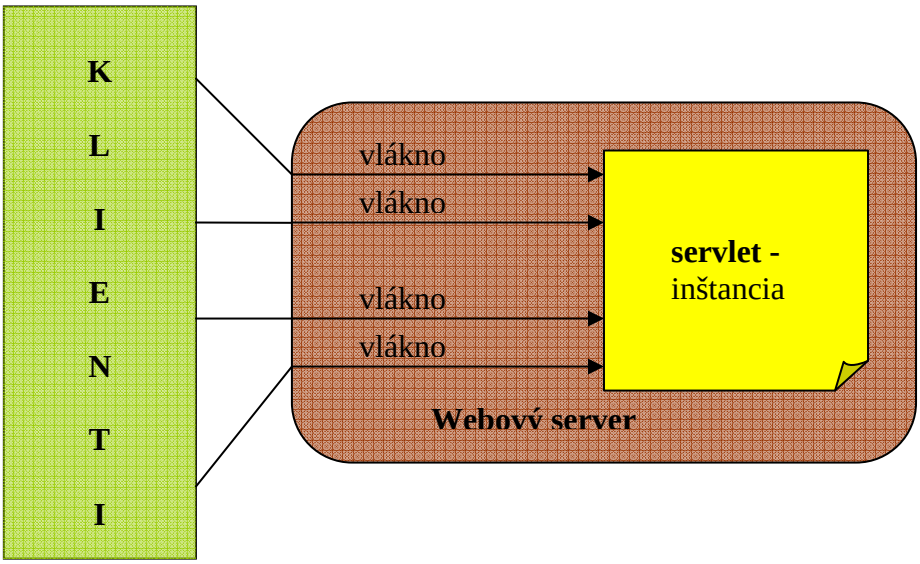
3.2 Technológia servletov

Servlety sú neoddeliteľnou súčasťou platformy J2EE od firmy Sun. Sú to programové komponenty bežiace na strane servera napísané v Jave. Načítavajú a spúšťajú sa pomocou JVM na príslušnom aplikačnom serveri. Java Servlety sa stali hlavným prostriedkom rozširovania a skvalitňovania webových aplikácií pomocou platformy Java. Podľa špecifikácie sú servlety nevizuálne komponenty, teda nezobrazujú sa pomocou žiadneho GUI. Avšak to neznamená, že nemôžu generovať používateľské rozhranie. V skutočnosti sa používajú servlety pracujúce nad internetovým protokolom Hypertext Transfer Protocol (ďalej len „HTTP“). Odpoveďou takýchto servletov je potom v prevažnej väčšine HTML kód. Servlety implementujú princíp požiadavka/odpoveď, typický pre architektúru klient – server [8].

Oproti Common Gateway Interface skriptom (ďalej len „CGI“), servlety nemajú výkonnostné obmedzenie a sú výkonnejšie, keďže vytvárajú len jediný proces, ktorý silno zaťažuje operačný systém. Tým umožňujú, aby bola každá užívateľská požiadavka spracovaná pomocou menej náročného vlákna, ktoré je spracované JVM. Požiadavky viacerých užívateľov je možné spracovať jednou inštanciou servletu. Servlety nespúšťa priamo webový server, potrebujú kontajner servletov, ktorému sa hovorí aj servletový stroj (servlet engine), ktorý spravuje servlet [6].



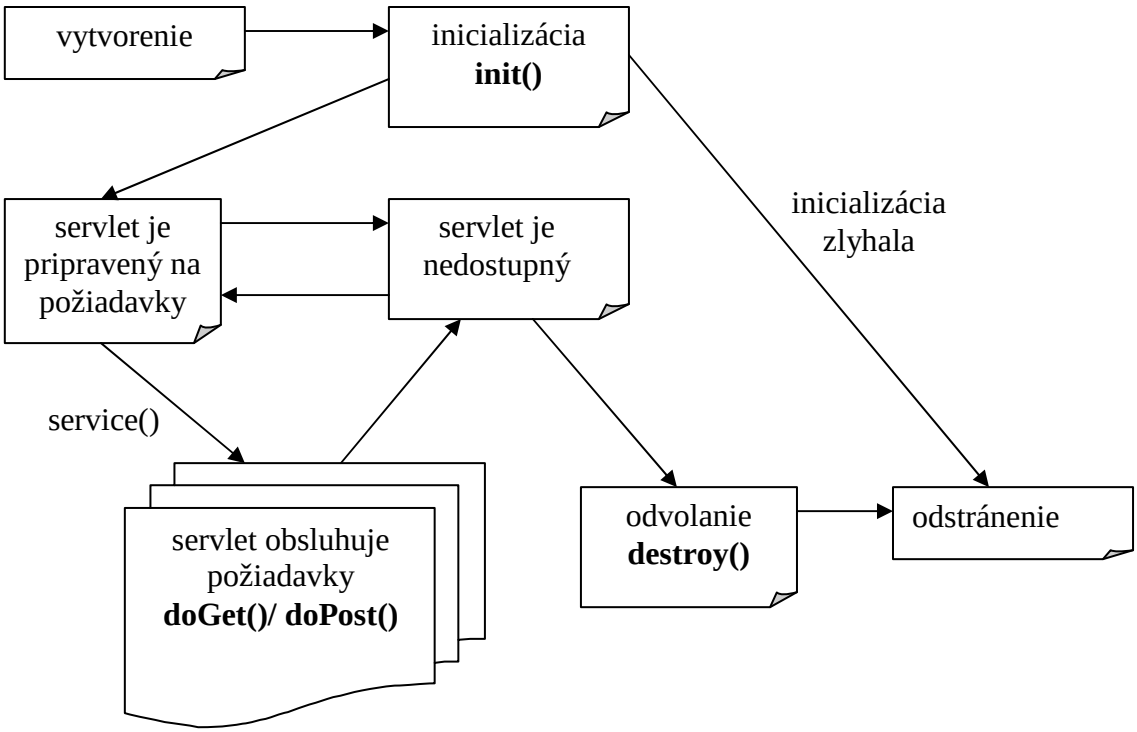
Obr. 4: Spracovanie požiadavky klienta



Obr. 5: Interakcia klient-server

Na tvorenie servletu je potrebné použiť balíček tried javax.servlet, ktorý obsahuje abstraktné triedy a rozhrania na tvorbu všeobecných servletov pre rôzne protokoly (HTTP, HTTPs, FTP). Balík javax.servlet.http rozširuje funkcionalitu základného balíka a pridáva podporu protokolu http. K najdôležitejšiemu rozhraniu balíka patrí rozhranie Servlet, definujúce potrebné metódy pre každý servlet. Sú nimi metódy init(), service() a destory(), ktoré zabezpečujú životný cyklus servletu.

Metóda init() slúži k počiatočnej inicializácii servera, môže byť zavolaná len jediný raz pre každú inštanciu objektu. Pomocou metódy service() servlet obsluhuje požiadavky SevletRequest a odpovede ServletResponse. Podľa typu požiadavky, zavolá príslušnú metódu. Najčastejšie používané metódy sú doGet() a doPost(). Opačný účinok ako init(), má metóda destroy(), ktorá uzatvára databázové spojenia, otvorené súbory a uvoľňuje servlet z pamäte.



Obr. 6: Životný cyklus servletu

Java Server Pages – JSP

Technológia Java Server Pages (ďalej len „JSP“) je prirodzeným rozšírením Java Servletov. Doplňuje ich architektúru, pretože poskytuje JSP kontajner, ktorý zaisťuje správu JSP stránok a ich prekladanie na servlety [5].

JSP sú textové dokumenty s príponou .jsp, obsahujúce kombináciu statických HTML tagov, tagov v štýle Extensible Markup Langauage (XML) a skriptletov. Po istých úpravách prekladača sa z JSP stávajú obyčajné servlety. Súbory .jsp sa vopred spracujú a prevedú na súbory s príponou .java a v tom okamžiku kompilátor Javy skompiluje zdrojový kód a vytvorí .class súbor, ktorý môže byť spustený kontajnerom servletov.

4. Vývojové prostredie NetBeans

Vychádzajúc z použitej literatúry, bolo pri vývoji aplikácie zvolené integrované vývojové prostredie NetBeans IDE. Tento program je voľne dostupný na Internete [10]. Pôvodne vznikol na Karlovej Univerzite v roku 1997 pod názvom Xelfi a neskôr bol kúpený spoločnosťou Sun, ktorá ho v roku 2000 uvoľnila ako open-source softvér, čo prispelo k jeho rozšíreniu. Jedná sa o pomerne jednoduchý program pre užívateľov na programovanie v Jave. Okrem samotného programu je na vytvorenie aplikácie potrebná vývojová sada JDK. Táto sada je dostupná buď spolu s programom NetBeans IDE alebo na WWW stránkach firmy Sun.

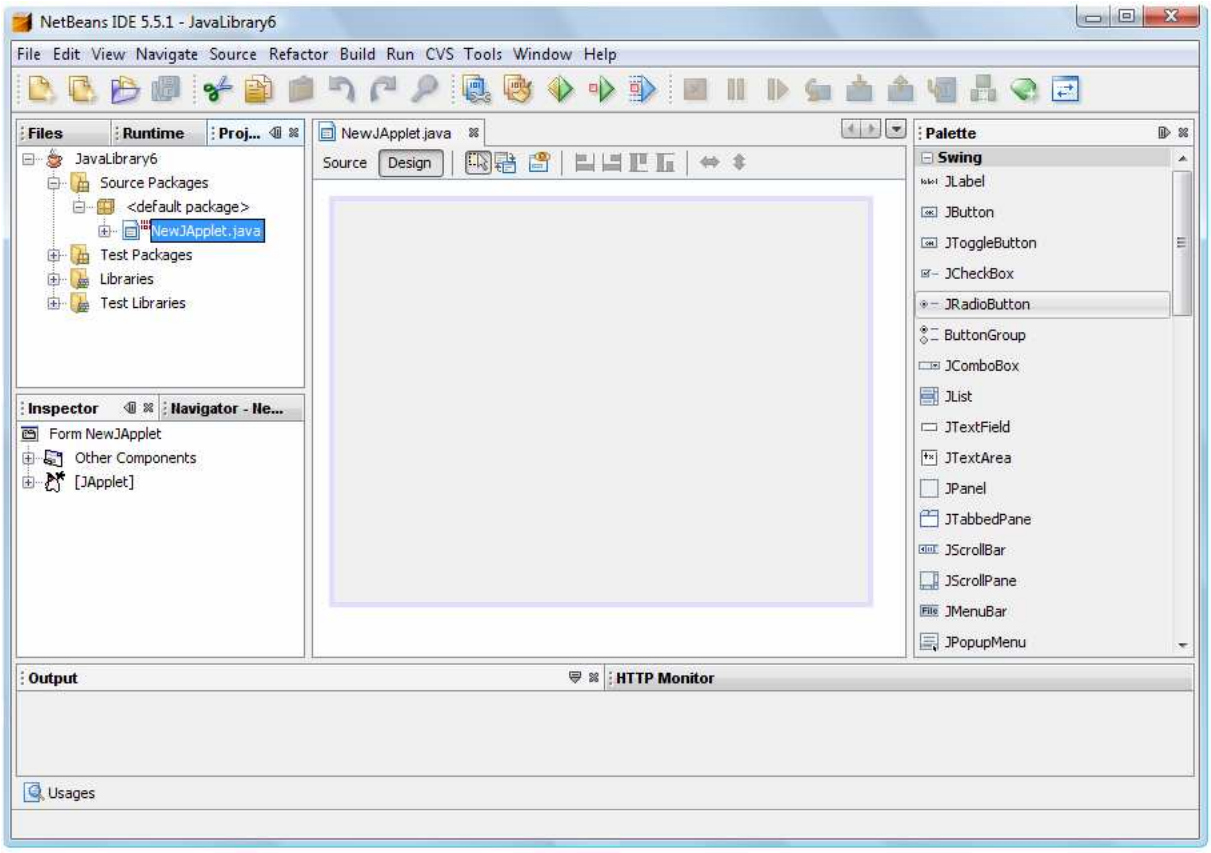
4.1 Vytvorenie apletu

Pri vytváraní appletu postupujeme nasledovne:

1. Spustíme program NetBeans.
2. Zvolíme príkaz File - New Project.
3. V novootvorenom okne New Project zvolíme General a z ponuky vyberieme Java Application.
4. Po potvrdení sa nám zobrazí okno Name and Location, kde zadáme cestu kam sa nami vytvorený projekt má ukladať.
5. Po stlačení tlačidla Finish sa nám vytvorí projekt a spustí vývojové prostredie.

Ako je na obrázku č. 7 vidieť, vývojové prostredie sa skladá z viacerých častí. V ľavej hornej časti Project sú informácie o otvorenom projekte a pripojených súčastiach ako balíčkov, knižníc atď. Pod týmto oknom je položka Inspector alebo Navigator, kde sa ukazuje, na ktorej úrovni programu sa nachádzame, resp. ktoré metódy sa používajú v zdrojovom kóde. V strednej časti sa nachádza okno Source/Design, kde môžeme písať zdrojový kód resp. pridávať rôzne položky z grafického užívateľského prostredia GUI.

V pravej časti je umiestnená paleta Palette, v ktorej môžeme vyberať z rôznych komponentov dvoch užívateľských prostredí, a to Abstract Windowing Toolkit (ďalej len „AWT“) a Java Foundation Class Swing (ďalej len „Swing“). AWT je staršie a jednoduchšie užívateľské prostredie – poskytuje (java.awt) 14 rozhraní, 77 tried a 11 podbalíkov so 141 triedami. Swing je novšie a (java.swing) má 20 rozhraní, 98 tried a 15 podbalíkov s 363 triedami. Z týchto čísiel vyplýva, aké veľké možnosti dané GUI majú.



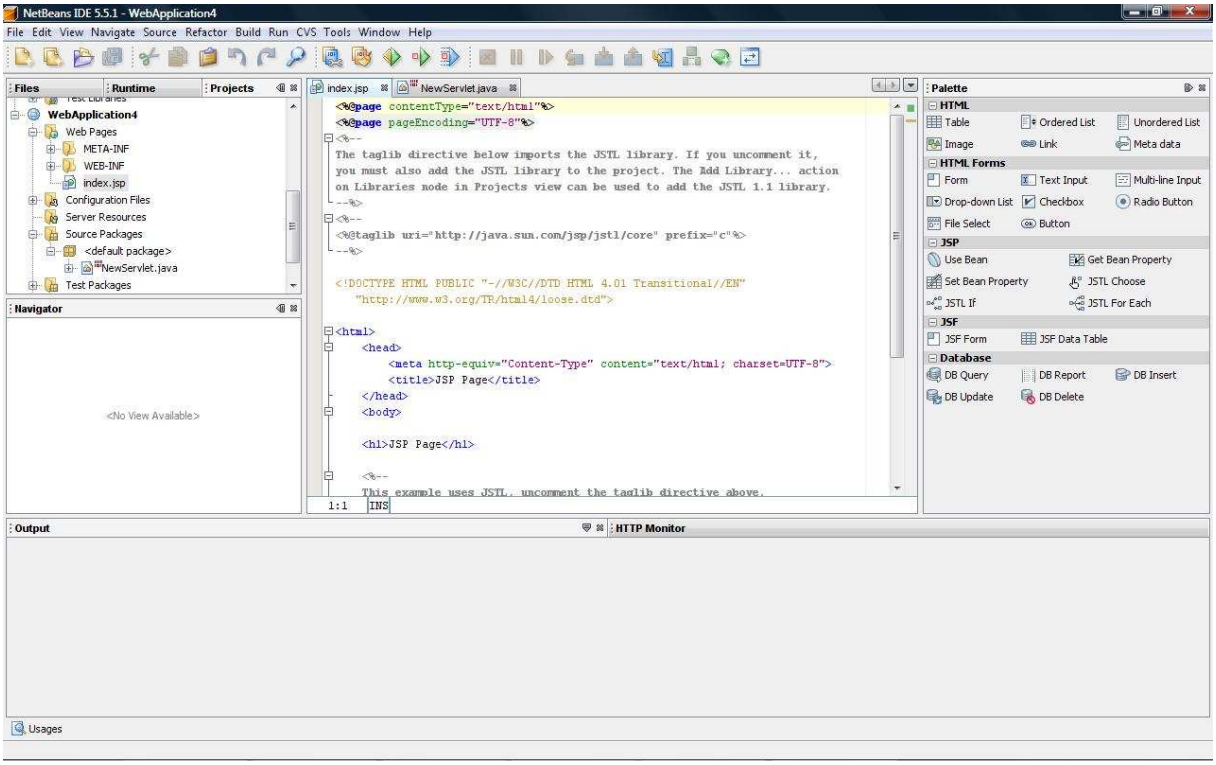
Obr. 7: Vývojové prostredie NetBeans pri tvorbe apletu

4.2 Vytvorenie servletu

Vytvorenie servletu je podobné ako vytvorenie apletu:

1. Vytvoríme nový projekt a z predvolenej ponuky Web vyberieme Web application.
2. Nazveme projekt a nastavíme cestu na jeho ukladanie.
3. Po potvrdení sa nám zobrazí okno s preddefinovanou stránkou index.jsp.
4. Na vloženie nového servletu klikneme pravým tlačidlom myši na náš projekt a zvolíme príkaz New – Servlet.
5. Na aktívne spojenie servletu s JSP stránkou nám poslúži HTML tag:
   `<a href="Náš_servlet"></a>`.

V ponuke Palette sú, podobne ako pri aplete, zobrazené preddefinované komponenty, ktoré môžeme vkladať do našej JSP stránky.



Obr. 8: Tvorba servletu

Vo vytvorenom servlete je automaticky (programom NetBeans) predpísaný základný kód, kde sú importované základné knižnice a triedy potrebné na funkčnosť servletu. Do samotného kódu servletu môžeme vkladať príkazy Javy a pomocou príkazu out.println() aj HTML kód.

5. Druhy mapových podkladov

Čo sa týka mapových podkladov, sú k dispozícii dva druhy máp: rastrové a digitálne vektorové mapy.

Rastrové mapy zobrazujú krajinu v grafickej podobe, zodpovedajúcej tlačeným mapám a je možné ich využiť ako orientačnú podkladovú vrstvu pri zobrazovaní vlastných dát. Sú zobrazené ako obrázky bez vnútornej inteligencie – to znamená, že k nim nie je možné pripojiť žiadne popisné dáta, nastaviť zobrazovanie individuálnych prvkov mapy podľa úrovne zväčšenia, zostavovať dopyty na atribúty mapových objektov a podobne. Napriek tomu predstavuje tento typ máp kvalitnú alternatívu vektorových máp.

Digitálne vektorové mapy predstavujú grafické zobrazenie krajiny, zodpovedajúce spracovaniu digitálnych dát pre využitie v grafických informačných systémoch. Tieto mapy umožňujú jednoduchú tvorbu rôznych analýz, poskytujú presný podklad pre riešenie čiastkových úloh, napr. v rámci mesta a je možné aj ich využitie na geokódovanie uličnej siete. Digitálne mapy obsahujú vrstvu uličnej siete s názvami ulíc, vrstvu hraníc zástavby, ďalej vrstvy vodných plôch, mestskej zelene a železníc.

6. Návrh aplikácie

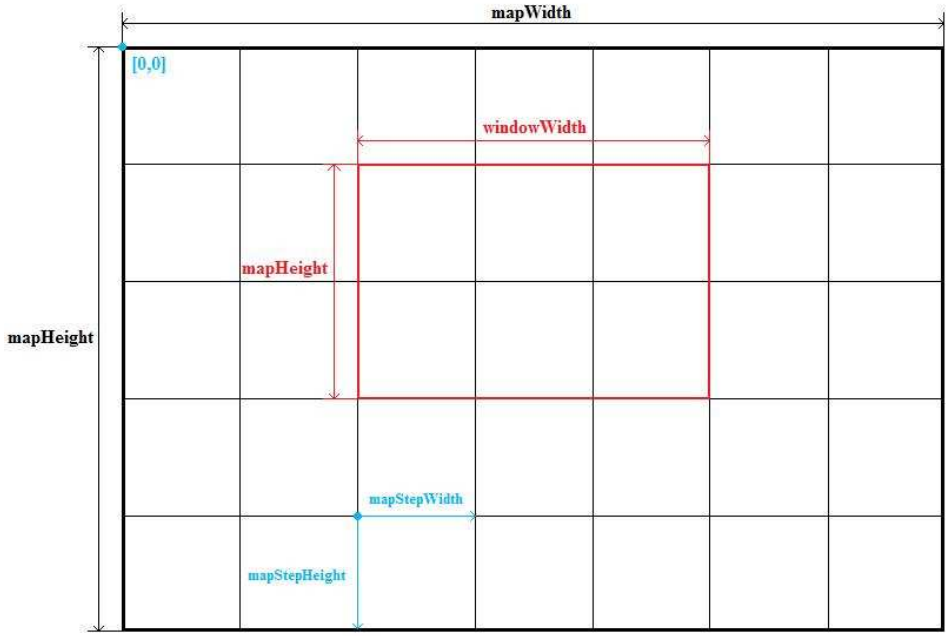
Pri návrhu aplikácie na zobrazovanie mapových podkladov som vychádzal z užívateľských poznatkov už spomenutých webových serverov.

Prioritou aplikácie bolo samotné zobrazenie mapy, ošetrenie pohybu po nej a čo najmenšie nároky na prenos dát medzi serverom a užívateľom, keďže niektoré mapy môžu zaberať stovky megabytov (MB).

Na základe týchto požiadaviek bola navrhnutá sieťová mriežka, ktorá obsahuje dôležité parametre:
- mapWidth, mapHeight,
- windowWidth, windowHeight,
- mapStepWidth, mapStepHeight.

Prvé dva parametre (mapWidth, mapHeight) určujú výšku a šírku načítanej mapy, ďalšie dva (windowWidth, windowHeight) výšku a šírku zobrazovacieho okna. Posledné parametre (mapStepWidth, mapStepHeight) určujú výšku a šírku kroku, ktorým sa pohybujeme po mape.

Z obrázka č. 9 vyplýva princíp, ktorý chceme dosiahnuť. Chceme, aby sa na samotnej web stránke zobrazovala určitá časť mapy, teda v našom prípade časť mapy ohraničená parametrami windowWidth a windowHeight. Vychádzajúca možnosť zobrazenia okna sa nastavuje pomocou parametrov windowPosX a windowPosY.



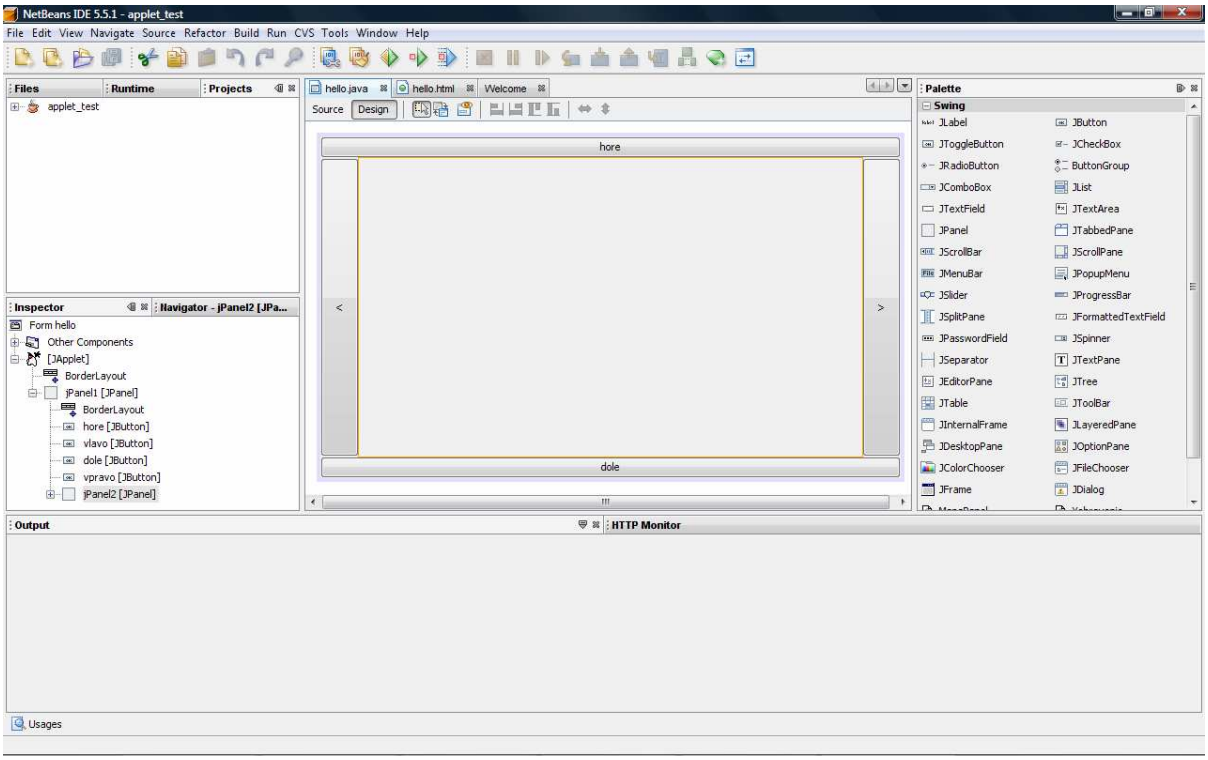
Obr. 9: Súradnicová mriežka s názvom dôležitých parametrov

6.1 Riešenie aplikácie apletom

Ako prvé nesmieme pri programovaní zabudnúť na deklaráciu parametrov a priradiť im zodpovedajúce hodnoty. Následne sa užívateľ bude z vychádzajúceho miesta pohybovať po mape. K pohybu slúži ovládanie pomocou tlačidiel (JButton), ktoré môžeme navrhnúť vďaka GUI vo vývojovom prostredí NetBeans.

Následne ich môžeme rozmiestniť podľa vlastnej potreby na zobrazovanú plochu. Kliknutím pravého tlačidla myši na jednotlivé buttony, zobrazia sa možnosti ich použitia a nastavenia vlastností. Po zvolení ponuky Events – Action – actionPerformed[] sa dostaneme automaticky do zdrojového kódu a môžeme písať príkazy, ktoré sa vykonávajú po použití tlačidla. V našej aplikácii sú umiestnené štyri tlačidlá na pohyb po mape hore, dole, vpravo a vľavo (Obr. 10). Medzi tieto tlačidlá je umiestnený panel (JPanel), do ktorého sa bude vykresľovať mapa.

Pre prácu s grafikou v Jave je potrebná trieda java.awt.Graphics, preto ju musíme na začiatok kódu importovať, resp. importujeme celú knižnicu java.awt.* pomocou príkazu import java.awt.*, keďže budeme používať aj iné súčasti tejto knižnice.



Obr. 10: Umiestnenie tlačidiel v časti Design

Vykreslenie, v našom prípade mapy, sa vykonáva len v metóde paint(Graphics g). V tejto metóde máme k dispozícii formálny parameter, podľa zvykového práva pomenovaný g [2], ktorý ukazuje na už vytvorený objekt triedy Graphics. Táto trieda umožňuje kreslenie geometrických útvarov, vypisovanie textu, kreslenie obrázkov. Na to, aby sme však mohli obrázky vykresľovať, potrebujeme ich najskôr načítať.

Na tento účel slúži vytvorená metóda getMap(). V nej sa na základe pozície v súradnicovej mriežke určuje názov súboru, ktorý máme získať (viď 4.2) a usporadúva sa do poľa typu Image. Metóda getMap() je volaná na začiatku metódy paint(). V nej sa vykresľuje obrázok príkazom g.drawImage(mapParts[p],i,j,this), kde mapParts sú časti mapy, ktoré kreslíme na základe indexu p a umiestňujeme ich na základe pozície i, j do súradnicovej mriežky.

Pri pohybe po mape potrebujeme, aby sa dopĺňali potrebné časti mapy a zároveň sa aj vykresľovali. To docielime umiestnením metódy repaint() do kódu ku každému buttonu. Repaint() volá metódu update(Graphics g). Metóda update() je v triede Component

naprogramovaná, aby vymazala celý komponent tým, že cezeň prekreslí obdĺžnik vo farbe pozadia. Potom nastaví farbu grafického kontextu na farbu popredia komponentu a nakoniec zavolá paint() [3]. Ten vykreslí požadovaný obrázok.

6.1.1 Implementácia apletu do webového prostredia

Aplikáciu riešenú formou appletu môžeme ľahko včleniť do Hypertext Markup Language (ďalej len „HTML“) kódu, jazyka vytvárajúceho WWW stránky. Znamená to, že sa nespustí priamo ako aplikácia, ale spustí sa s otvorením HTML dokumentu WWW prehliadačom.

Aby sme mohli applet použiť pri tvorbe WWW stránok, potrebujeme ho dostať do HTML súboru, z ktorého je vytváraná stránka. Požijeme na to párovú značku (tag) <APPLET></APPLET>.

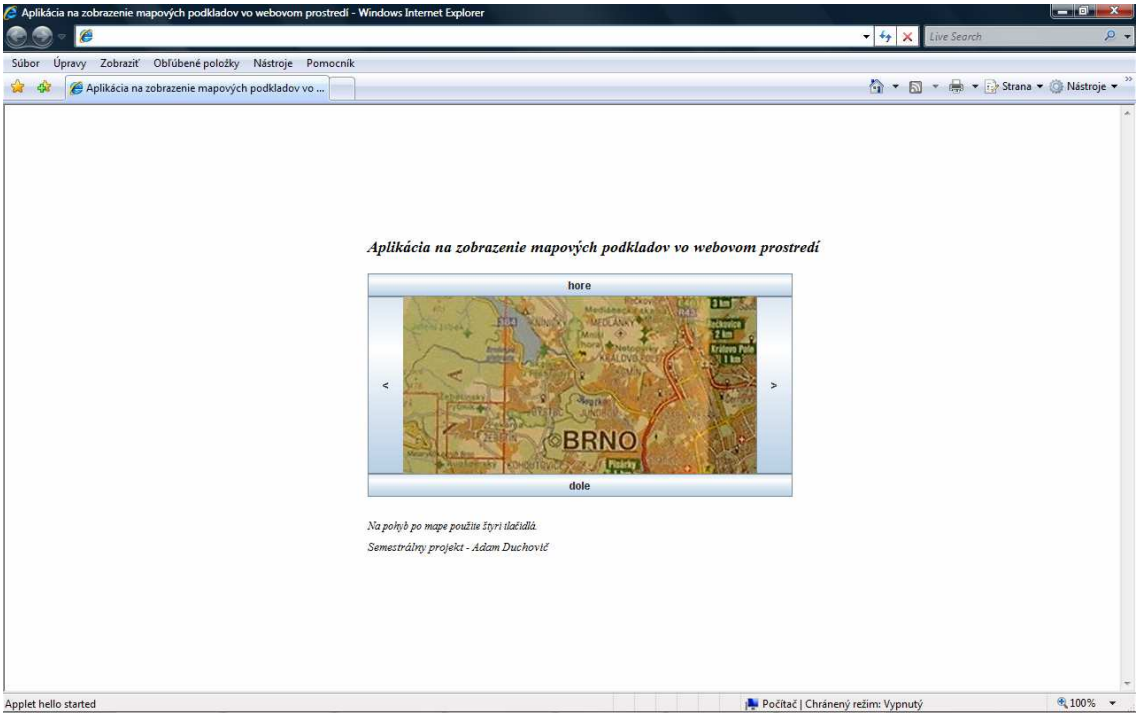
```
<APPLET codebase="classes" code="hello.class" width=500 height=250></APPLET>
```

Pri používaní v rôznych prehliadačoch sa jeden applet môže zobrazovať odlišne. Avšak vo väčšine prehliadačov sa objaví automatická možnosť doinštalovania pluginov, ak nie sú už vopred nainštalované. Tieto pluginy umožnia korektné zobrazenie appletu.

Základnými parametrami Appletu v HTML kóde sú code, width, height. V parametri code je uložená cesta k súboru .class, v codebase adresa, kde sa bude hľadať súbor tried pri spustení. Parameter width je šírka a height výška zobrazovaného appletu.

Medzi ďalšie parametre, ktoré môžeme dopísať patrí:

ALT – text, ktorý sa použije v prípade, že prehliadač nedokáže spúšťať applety.
ALIGN – zarovnanie appletu na WWW stránke vzhľadom k textu alebo okoliu.
- možnosti left, right – vzhľadom k stránke vodorovne,
- možnosti top, absmiddle, bottom- vzhľadom k textu zvisle.



Obr. 11: Zobrazenie appletu v prehliadači

6.2 Riešenie aplikácie servletom

Postup riešenia sa podobá na aplet. Najskôr preddefinujeme potrebné parametre pre našu pozíciu na mape a zobrazované okno mapy. Avšak servlety sú podľa špecifikácie nevizuálne komponenty, teda sa nezobrazujú pomocou žiadneho GUI [8], preto musíme ovládacie prvky (tlačidlá) zahrnúť do HTML kódu. Urobíme tak tagom `input`, kde nastavíme parametre

```
<input type=”submit” value=”smer_pohybu” name=”smer_pohybu”>.
```

Týmto sa nám po načítaní servletu zobrazí jedno tlačidlo s názvom smeru pohybu. Keďže sa budeme pohybovať po mape štyrmi smermi, vložíme ešte ďalšie tri tlačidlá a za pomoci podmienky if ošetríme, aby sme sa pri pohybe nedostali za hranicu mapy.

Na samotné vkladanie mapy slúžia dva cykly `for` a HTML tag pre vloženie obrázka `<img>` , kde parameter pozícia označuje miesto na vloženie a zároveň číslo obrázka. Pre lepšiu prehľadnosť a usporiadanie tlačidiel je mapa spolu s nimi vložená a usporiadaná do tabuľky za pomoci párových tagov `<table>`, `<tr>`, `<td>`. Párová značka `form` zabezpečuje prenesenie informácie o pohybe užívateľa a znovunačítanie stránky po posunutí sa na mape.

Príklad kódu servletu:

- Podmienka vytvorená pre pohyb po mape, za pomoci tlačidiel:

```
        if (request.getParameter("hore") != null)
        {
            ypos -= 1;
            smer = "hore";
                if (ypos<0)
                    ypos=0;
        }
```

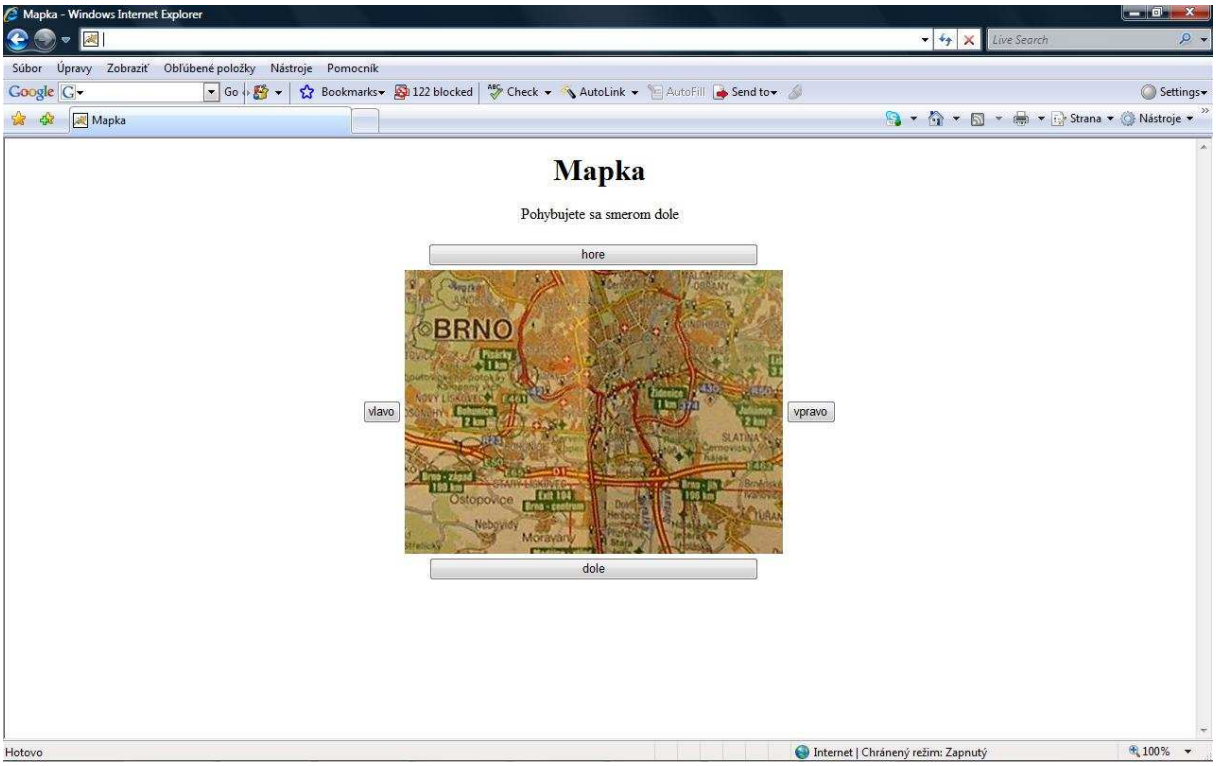
- Vloženie HTML kódu do stránky servletu:

```
     out.println("<html>");
     out.println("<head>");
     out.println("<title> Mapka</title>");
     out.println("</head>");
     out.println("<body>");
     out.println("<form action=\"GreetingServlet\"
method=\"POST\">");
     out.println("<input type=\"hidden\" value=\""+xpos+"\"
name=\"x\" />");
     out.println("<input type=\"hidden\" value=\""+ypos+"\"
name=\"y\" />");

     out.println("</form>");
     out.println("</body>");
     out.println("</html>");

out.close();
```

Aby sme mohli nami vytvorenú JSP stránku so servletom spojazdniť, potrebujeme vytvoriť serverové prostredie. Medzi známe servery patrí server Apache od neziskovej organizácie Apache Software Foundation [12]. Tá zakladá a podporuje niekoľko dôležitých softwarových projektov. Pre servlety a JSP stránky je to projekt Jakarta, ktorý poskytuje servlet a JSP kontajner Tomcat [5]. Z www stránky si stiahneme build Tomcatu. Pre správnosť ešte potrebujeme mať nainštalovanú verziu JDK.



Obr. 12: Výsledný servlet zobrazený v prehliadači

6.4 Implementácia mapových podkladov

Na to, aby sme do nami vytvorenej aplikácie mohli vložiť mapu, potrebujeme ju rozrezať na preddefinované rozmery (viď vyššie), v našom prípade parametre mapStepWidth x mapStepHeight (50x50 pixelov).

V našej práci nám pomôže freewarový program The Castle´s Split Image, voľne dostupný na stránkach firmy TheCastle. Tento program umožňuje načítať obrázok, v našom prípade mapu, upraviť ju na potrebnú veľkosť a následne rozdeliť na časti požadovaných rozmerov.

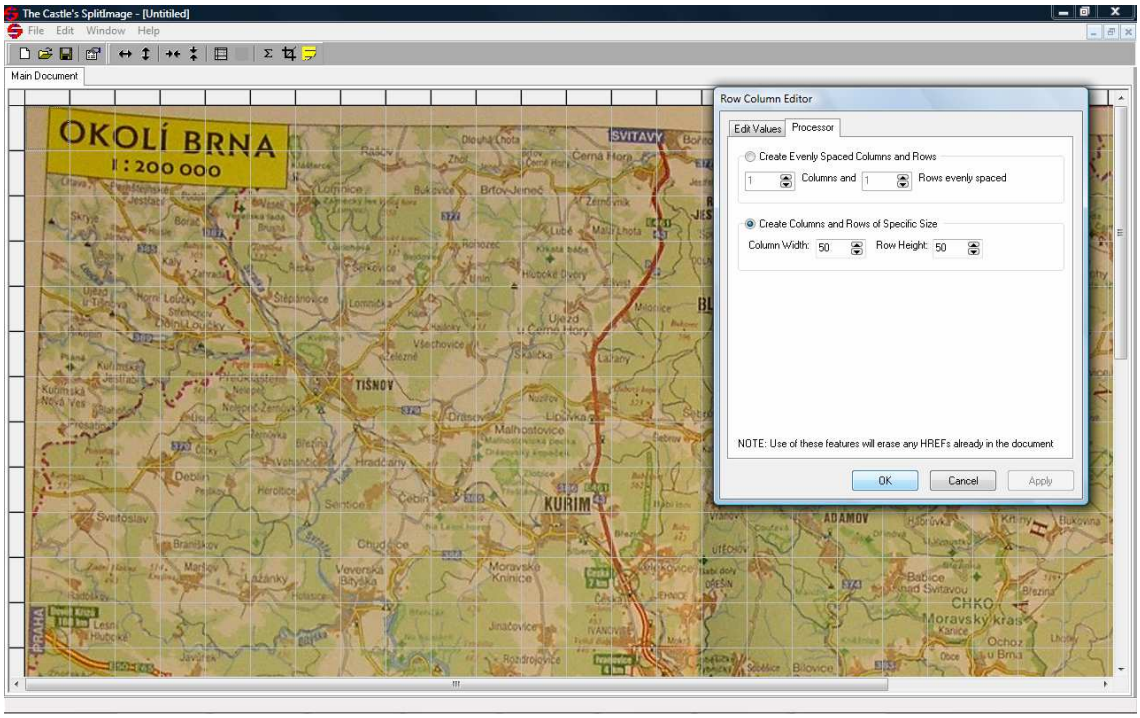
Po spustení programu zvolíme príkaz New a následne v okne zadáme obrázok, ktorý chceme rozdeliť. Po jeho otvorení máme možnosť upraviť ho za pomoci menu Edit. Nás zaujíma možnosť Crop/Expand image, kde zaokrúhlime rozmery obrázka podľa potreby, v našom prípade, pre lepšiu orientáciu na stovky.

Po tomto kroku nasleduje rozdelenie, opäť v menu Edit a možnosť Edit Using dialog. V tomto menu klikneme na záložku Processor, vyberieme Create Columns and Rows of

Specific Size, kde zadáme požadované rozmery podobrázkov. Po kliknutí na tlačítko Apply sa nám obrázok rozdelí. V menu File môžeme zmeniť v nastaveniach Preferences formát obrázkov, použijeme *.jpg .

Ak sme spokojní s výsledkom, môžeme vygenerovať podobrázky a to v menu File – Generate Images. Pri ukladaní obrázkov sa nás program spýta na umiestnenie a názov obrázkov. Názov bude rovnaký pre každý obrázok, líšiť sa budú len v čísle napr. nazov_1.jpg.

Program čísluje obrázky zľava doprava, po prejdení príslušného riadku sa posunie nadol. Toto číslovanie vyhovuje navrhnutej mriežke. Pre úplnosť, nesmieme zabudnúť zadať príslušnú cestu k jednotlivým obrázkom vo funkcii getMap() resp. pri vkladaní v tagu `<img>`.



Obr. 13: Rozdelenie obrázka v programe SplitImage

6.5 Zabezpečenie modularity

Modularitou rozumieme prístup k tvorbe hardwaru alebo softwaru v snahe utvoriť kompletný funkčný celok zo zameniteľných častí, modulov.

Pretože softwarové aplikácie patria, pre programátorov, vôbec k najzložitejším problémom dnešnej doby, musia všetky moderné technológie umožňovať jednoduché delenie softwaru na menšie časti, tzv. moduly. Softwarové moduly poskytujú niekoľko výhod [5]:

- riešenie jednoduchších problémov podporuje prehľadnosť riešenia,
- môžeme vytvárať a testovať každý modul zvlášť,
- pretože väčšie problémy sa spravidla rozdelia na podobné menšie problémy, zvyšuje sa schopnosť opätovného použitia softwaru.

Pri rozširovaní našej aplikácie, môžeme vložiť ďalšie servlety do projektu s podobným výstupom, avšak zameraných na iné dáta (napr. mapy iných oblastí). Tieto servlety by boli volané v prípade potreby, na základe požiadavky užívateľov.

Záver

Bakalárska práca sa zaoberá návrhom a implementáciou aplikácie na zobrazovanie mapových podkladov vo webovom prostredí. Z použitej literatúry a konzultácie s vedúcim vyplynulo riešenie programovacím jazykom Java.

Na zobrazovanie jednotlivých obrázkov ako celku bola použitá súradnicová mriežka, ktorá na základe polohy určila, aký obrázok mapy sa má načítať a kam má byť umiestnený. Riešenie danej aplikácie bolo rozdelené na dve časti.

Za pomoci vývojového nástroja NetBeans sa podarilo vytvoriť základ požadovanej aplikácie. Zobrazuje výsek mapy, po ktorom sa dá pohybovať na základe stláčania tlačidiel užívateľom. Po každom posunutí zobrazovaného úseku je volaná metóda repaint(), ktorá prekreslí zobrazované okno podľa pohybu užívateľa po súradnicovej mriežke. Do zobrazovaného výseku sa postupne načítavajú obrázky, ktoré vznikli rozdelením pôvodného obrázka mapy v programe The Castle´s Split Image pomocou vytvorenej metódy getMap(). Vytvorený applet je ľahko implementovateľný do HTML použitím párového tagu <APPLET></APPLET>. Po jeho vložení na WWW stránku, je aplikácia funkčná.

V tom istom vývojovom nástroji bola vytvorená aj webová aplikácia s použitím servletov. Na úpravu vzhľadu zobrazovanej stránky boli použité základné HTML príkazy a na zobrazovanie mapky dva cykly `for`. Na implementovanie vytvorenej JSP stránky s vytvoreným servletom bol použitý server Apache so servletovým kontajnerom Tomcat. Po jeho nainštalovaní a spustení sa v prehliadači zobrazila požadovaná stránka.

V prvej časti riešenia sa väčšina záťaže sústredí na klienta a v druhej časti je táto záťaž v prevažnej miere prenesená na server. Je pravda, že aplety sú výkonnou technológiou, poskytujú veľké možnosti v práci s GUI, avšak zaťažujú užívateľa a zahlcujú Internet. Jedná sa najmä o potrebu nainštalovať požadované doplnky alebo stiahnuť Java aplet.

Naproti tomu servlet sa načíta v servletovom kontajneri, ktorý servlet spravuje. Tento kontajner je spojený s inštanciou webového serveru a na základe požiadavky sa s ňou pracuje. Servlety majú výborné vlastnosti, avšak tiež sa nevyhli kritike. Predovšetkým je náročné vykonávať zmeny v HTML kóde, pretože daný servlet musíme zakaždým skompilovať.

Aj napriek tejto nevýhode, by bolo vhodnejšie uprednostniť realizáciu danej aplikácie za pomoci technológie servletov.

Zoznam použitých skratiek a symbolov

API – Application Programming Interface – Aplikačné programové rozhranie Java

AWT – Abstract Windowing Toolkit – staršie GUI v Jave

CGI – Common Gateway Interface - je protokol pre pripojenie externých aplikácií s webovým serverem.

CPU – Central Processing Unit - procesor

FTP – File Transfer Protocol – protokol prenosu súborov medzi počítačmi

GUI – Graphic User Interface – Grafické užívateľské prostredie

HTML – Hypertext Markup Language – jeden z jazykov na vytváranie WWW stránok

HTTP – Hypertext Transfer Protocol – základný internetový protokol

HTTPs – Hypertext Transfer Protocol Sacure – zabezpečená verzia HTTP

JDK – Java Development Kit – vývojová sada

JVM –Java Virtual Machine – strojový jazyk

JSP **–** Java Server Pages – špeciálna metóda ako písať servlety

MB – Megabyte – dátová jednotka

Sun – firma Sun Microsystems - firma, ktorá vyvinula Javu

Swing – Java Foundation Class Swing – novšie GUI v Jave

WWW – World wide web – celosvetová počítačová sieť

XML – eXtensible Markup Language – rozšírený značkovací jazyk

Zoznam literatúry a použitých zdrojov

[1] ZAKHOUR, Sharon, et al. Java 6: Výukový kurz. 2007. 536 s. ISBN 978-80-251-1575-6.
[2] HEROUT, Pavel. Učebnice jazyka Java. 2007. 384 s. ISBN 978-80-7232-323-4.
[3] HEROUT, Pavel. Java - grafické uživatelské prostředí a čeština. 2007. 320 s. ISBN 978-80-7232-328-9.
[4] HEROUT, Pavel. Java-bohatství knihoven.2006. 256 s. ISBN 80-7232-288-5.
[5] BOLLINGER, Gary, NATARAJAN, Bharanthi. *JSP - Java server pages* . 2003. 420 s. ISBN 80-247-0340-8.
[6] CAVANESS, Chuck. Programujeme Jakarta Struts. 2003. 468 s. ISBN 80-247-0667-9

Skriptá:
[7] BODEČEK, Kamil, ČÍKA, Petr , KRAJSA, Ondřej . *Multimediální služby (BMDS): Laboratorní cvičení*. [s.l.]: [s.n.], 2007. 69. Dostupný z WWW: <www.vutbr.cz/elearning>.

Internetové články:
[8] BRANICKÝ, Marek. *Java Servlets : predstavenie technológie* [online]. 2003-2004 , 2004 [cit. 2008-05-30]. Dostupný z WWW: <www.interval.cz>.

WWW odkazy:
[9] *Googlemaps* [online]. 2007. 2007 , 2007 [cit. 2008-05-30]. Dostupný z WWW: <www.googlemaps.com>.
[10] *NetBeans* [online]. 2007. 2007 , 18.12.2007 [cit. 2008-05-30]. Dostupný z WWW: <www.netbeans.org>.
[11] *TheCastle: SplitImage* [online]. 2007. 1998, 13.12.2007 [cit. 2008-05-30]. Dostupný z WWW: <www.thecastle.com>.
[12] *Apache server* [online]. 2008. 2008 , 2008 [cit. 2008-05-30]. Dostupný z WWW: <www.jakarta.apache.org>.