UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Kalkulátor pro mobilní zařízení

Michal Malec

Bakalářská práce
2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Michal MALEC**
Osobní číslo:	**I07922**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Kalkulátor pro mobilní zařízení**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce je pomocí Java technologie vytvořit aplikaci kalkulátor pro mobilní zařízení. Všechny dnešní telefony v sobě kalkulačku již obsahují. Ta ale bohužel neumí víc než základní operace, a proto není vhodná pro složitější výpočty. To by měla vyřešit tato práce.
Teoretická část:
Popis a vlastnosti JME (Java Micro Edition) a JavaFX technologií pro programování mobilních zařízení. Možnosti programování grafických aplikací pro mobilní zařízení. Práce s malými nebo velkými čísly v javě.
Implementační část:
Pro konkrétní mobilní zařízení vytvořit aplikaci, která bude kromě základních funkcí realizovat i složitější matematické operace a funkce. Pro programování využít vhodné vývojové prostředí.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

***Qusay H. Mahmoud: Naučte se Java 2 Micro Edition, GRADA Publishing, 2002.**

***Sharon Zakhour a kol.: Java 6 - Výukový kurz, Computer Press, 2007.**

Vedoucí bakalářské práce: **Ing. Zdeněk Šilar**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. ledna 2010**

Termín odevzdání bakalářské práce: **14. května 2010**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2010

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 09. 05. 2010 Michal Malec

Poděkování

Tímto bych chtěl poděkovat panu ing. Zdeňku Šilarovi za to, že mi umožnil zpracovat bakalářskou práci na mnou vybrané téma a byl mi nápomocen při závěrečných konzultacích.

Anotace

Práce shrnuje možnosti programování aplikací pro mobilní zařízení pomocí technologie Java ME a JavaFX. Práce řeší problém s absencí kalkulátoru na mobilních zařízeních schopného složitějších matematických výpočtů a operací.

Klíčová slova

kalkulátor, mobilní zařízení, Java ME, JavaFX

Title

Calculator for mobile device

Annotation

The work summarizes possibilities of programming applications for mobile devices through the use of Java ME and JavaFX. The work solves the problem with an absence of calculator in mobile devices, which is capable of solving complicated mathematical calculations and operations.

Keywords

calculator, mobile device, Java ME, JavaFX

Obsah

Seznam zkratek

API	Aplication Programming Interface
CDC	Connected Device Configuration
CLDC	Conected Limited Device Configuration
EE	Enterprise Edition
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HSDPA	High-Speed Downlink Packet Access
JDK	Java Development Kit
JRE	Java Runtime Environment
JVM	Java Virtual Machine
LCD	Liquid Crystal Display
ME	Micro Edition
PC	Personal Computer
PDA	Pocket Digital Asistant
RAM	Random Access Memory
ROM	Read Only Memory
SE	Second Edition
TFT	Thin-film transistor
UML	Unified Modeling Language
VM	Virtual Machine
WVGA	Wide Video Graphics Array

Seznam obrázků

Seznam tabulek

1 Úvod

Tato práce má za úkol shrnout edice Javy pro programování mobilních aplikací a vyřešit problém se zatím dostupnými kalkulátory pro mobilní zařízení. Každý telefon obsahuje v základu předinstalovaný kalkulátor, který ovšem umí povětšinou pouze základní matematické operace. V práci přiblížím možnosti programování aplikací v Java ME a JavaFX, zejména pak jejich využití ve vývoji grafických uživatelských prostředí. V rámci vývoje aplikace kalkulátoru uvedu možnosti práce s velkými a malými čísly, nejprve v Java SE a pak pro již zmiňované edice Java ME a JavaFX. Nejprve krátký přehled edic Javy.

2 Technologie Javy

2.1 Java SE

Java SE (Standard Edition) je dnes základní edicí Javy pro programování aplikací pro stolní počítače. Programovat v Java SE je možné dnes na všech rozšířených operačních systémech. Těmi jsou Windows, MacOS X, Linuxové distribuce a Solaris. Java SE také umožňuje programovat aplikace pro servery a aplety pro webové stránky, aplikace pro databáze a komunikaci s nimi. Následující údaje jsou vytaženy ze stránek společnosti Oracle. (Oracle, 2010)

Javu SE je možné stahovat ve dvou verzích. První z nich je Java SE Runtime Enviroment (JRE). JRE nám poskytne veškeré nástroje a prostředky pro spuštění Java aplikací. JRE obsahuje všechny knihovny potřebné pro spouštění stejně jako Java Virtual Machine (JVM). JRE také obsahuje Java Plug-in pro spuštění Java apletů ve všech současných webových prohlížečích a Java Web Start. Java Web Start umožňuje spuštění aplikací přes síť. V současné době je k dispozici JRE 6 update 19. Druhá verze je Java SE Development Kit (JDK). JDK je určeno pro vývoj aplikací v Java SE. Obsahuje jak JRE, tak nástroje potřebné pro kompilaci a debugování aplikací při vývoji.

Java Virtual Machine, jak se píše v oficiální dokumentaci, je abstraktní výpočetní stroj, který má vlastní instrukční sadu a alokovanou paměť pro svůj běh a běh aplikací. JVM je k dispozici pro různé platformy, aby zajistil hardwarovou nezávislost a nezávislost na operačním systému. Java SE poskytuje dvě implementace virtual machine. První z nich je Java HotSpot Client VM. Tato VM je určena především pro klientské aplikace. Je nastavena tak, aby pokud možno co nejvíce redukovala čas potřebný ke startu aplikace a omezila paměťovou náročnost. Druhá VM je Java HotSpot Server. Tahle VM je designována pro maximální rychlost provádění programu výměnou za větší paměťovou náročnost a spouštěcí čas.

Pro nasazení Javy SE v business sektoru je možnost získat Java SE for Business. Tato edice Javy v podstatě nabízí možnost dlouhodobé podpory v podobě záplat a updatů i pro starší vydání Javy (1.4.2, 5.0 a 6). V podnikovém nasazení je někdy potřeba zachovat

starší software, třeba z důvodu neexistence jeho novějšího vydání, a proto je nabízena tato možnost distribuce Javy SE. Zákazník tak bude získávat po delší dobu kritické záplaty a bezpečnostní updaty uvolňované i pro tyto starší verze. Jako příklad Oracle uvádí, že pro Javu 1.4.2 bude tyto updaty poskytovat až do roku 2013.

Poslední edicí z rodiny Java SE je Java SE pro vestavěné zařízení. Tato verze je určená zejména pro systémy běžící na linuxovém jádře. V dnešní době to je především operační systém pro mobilní zařízení Google Android. Je také dostupná verze pro operační systém Windows XP Embedded pro procesory s architekturou x86.

2.2 Java EE

Java EE (Enterprise Edition) je edice Javy určená pro vývoj komerčního softwaru. Java EE je založená na Java SE, ale přidává navíc další knihovny a systémové služby pro zlepšení škálovatelnosti, přístupnosti, bezpečnosti a integrity, vlastností tolik potřebných pro komerční software. Java EE nabízí dva balíčky pro vývoj aplikací. První z nich je Java EE SDK (software development kit), balíček obsahující nástroje pro vyvíjení a testování aplikací v Java EE. Druhý balíček je Java EE Web Profile. Tento balíček je určený pro programování webových aplikací. Oba dva nástroje obsahují Sun GlassFish Server, což je server pro běh a vývoj Java EE aplikací pro komerční použití.

2.3 Java ME (Java Micro Edition)

Java ME je určena pro programování na malých přenosných zařízeních. Tím jsou především v současné době mobilní telefony bez možnosti spustit robustnější aplikace psané v Java SE. Java ME je v základu rozdělena na dvě konfigurace. Konfigurací je nastaven typ, nebo spíše skupina zařízení, pro které je daná aplikace určena. V dnešní době jsou dvě konfigurace pro Java ME. Těmito konfiguracemi jsou CLDC a CDC.

2.3.1 CLDC (Conected Limited Device Configuration)

Je určena především pro mobilní telefony a starší typy PDA s omezenou pamětí a výkonem. V současné době snad všechny mobilní telefony umějí pracovat s Java ME Midlety, jak se nazývají aplikace napsané v konfiguraci CLDC. CLDC proto obsahuje opravdu jen nejnutnější prostředky pro vytváření Midletů.

Typické požadavky pro konfiguraci CLDC na zařízení, na kterém může být tato konfigurace provozována: (Sun Microsystems, 2006)

- CLDC vyžaduje 16 nebo 32bit procesor s frekvencí hodin minimálně 16MHz,
- alespoň 160 KB volného místa v hlavní paměti zařízení (většinou označována jako ROM) pro knihovny CLDC a samotný Java Virtual Machine,
- 192 KB volné paměti pro uložení souborů a dat samotné Java ME platformy,
- povětšinou je vyžadována i možnost připojení k internetu, buď přes bezdrátovou technologii nebo přes ostatní typy připojení (GPRS, HSDPA)

Pod CLDC najdeme ještě několik profilů. Konfigurace a profil dávají dohromady konečnou sadu nástrojů pro programování aplikací v Java ME pro cílovou skupinu zařízení.

2.3.2 Mobile Information Device Profile (MIDP)

Profil MIDP je určen pro programování aplikací pro mobilní telefony. Profil přidává k CLDC konfiguraci nástroje pro vytváření jednoduchých GUI (Graphical User Interface- Grafické uživatelské rozhraní), nástroje pro práci se sítí (komunikace je založená na protokolu http 1.1) a možnosti práce s lokálním úložištěm. V současné době jsou k dispozici dvě verze MIDP. MIDP 1.0 obsahuje jen ty nejnutnější balíčky pro vytváření jednoduchých aplikací. Profil MIDP 2.0 rozšiřuje možnosti o něco více. Především o možnost přehrávání médií a práci s desetinnými čísly. Dalším značným rozšířením oproti MIDP 1.0 je API zaměřené na jednoduché 2D hry. (Sun Microsystems, 2006)

2.3.3 CDC (Connected Device Configuration)

Tato konfigurace je určena pro zařízení s mnohem větším výkonem než u předchozí konfigurace CLDC. CDC většinou obsahuje kompletní implementaci Java virtual machine a také dovoluje využívat mnohem více prvků z prostředí Java. Tato konfigurace se nejvíce používá u novějších PDA, smartphonů a některých set top boxů. (Oracle, 2010)

Typické požadavky na zařízení pracující s konfigurací CDC:

- 32bit procesor s alespoň 2MB paměti RAM
- 2,5MB paměti ROM

Stejně jako CLDC i CDC konfigurace obsahuje několik profilů. Podíváme se na ně trochu zblízka.

2.3.4 Foundation Profile

Foundation Profile je základní z profilů pod konfigurací CDC. Tento profil je souborem Java API pro zařízení s limitovanými možnostmi a povětšinou pro zařízení nepotřebující grafické uživatelské prostředí. Profil poskytuje kompletní prostředí pro aplikace spotřební elektroniky a embedded zařízení. Tento profil ve verzi 1.1.2 zahrnuje také několik bezpečnostně orientovaných volitelných balíčků z prostředí Java SE. Těmito balíčky jsou například Java Authentication and Authorization service, dále pak Java Secure Socket Extension nebo také Java Cryptography Extension.

2.3.5 Personal Basis Profile

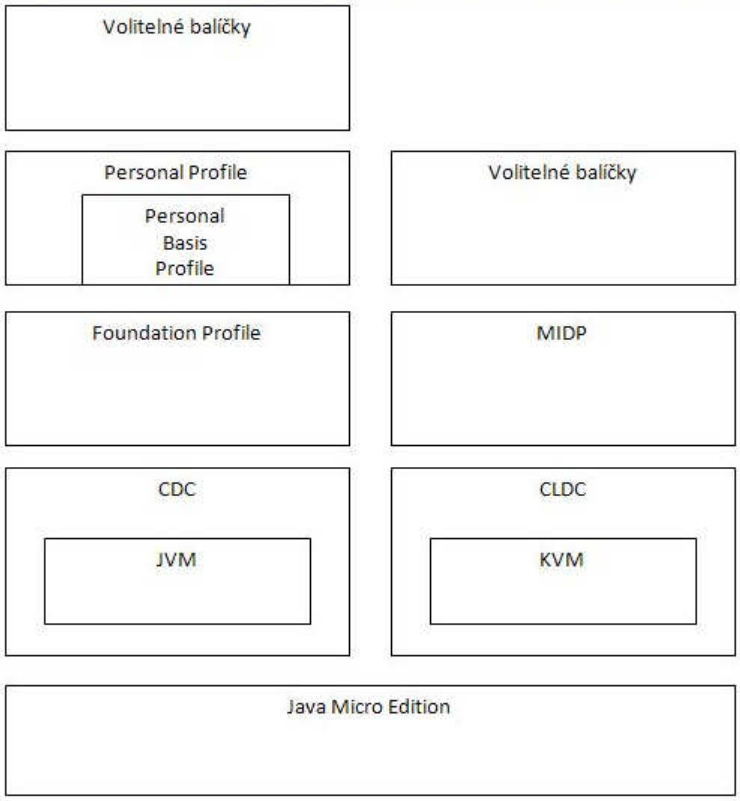
Personal Basis Profile je vlastně nadstavba Foundation profile. Tento profil je zaměřen na podobná zařízení jako Foundation profile s tím rozdílem, že je pomocí tohoto profilu možné vytvářet jednoduchá grafická rozhraní. Tento profil proto obsahuje některé třídy z balíčku AWT. Nejsou zde zastoupené žádné složitější grafické prvky, protože se stále nepředpokládá, že by tato zařízení měla nějaké polohovací zařízení typu myši apod.

Tento profil může být použit třeba pro programování složitějších ovládacích prvků například u Blu-ray disk přehrávačů.

2.3.6 Personal Profile

Toto je poslední z profilů pod konfigurací CDC. Tento profil opět rozšiřuje možnosti Personal Basis profilu, takže je opět jeho nadstavbou. Profil tentokráte disponuje úplnou podporou pro grafická rozhraní založených na AWT. Tento profil je určen především pro PDA, moderní komunikátory, herní konzole atd. Profil používá model Apletů pro vývoj aplikací.

Na obrázku 1 vidíme uspořádání jednotlivých konfigurací a profilů v rámci Java ME

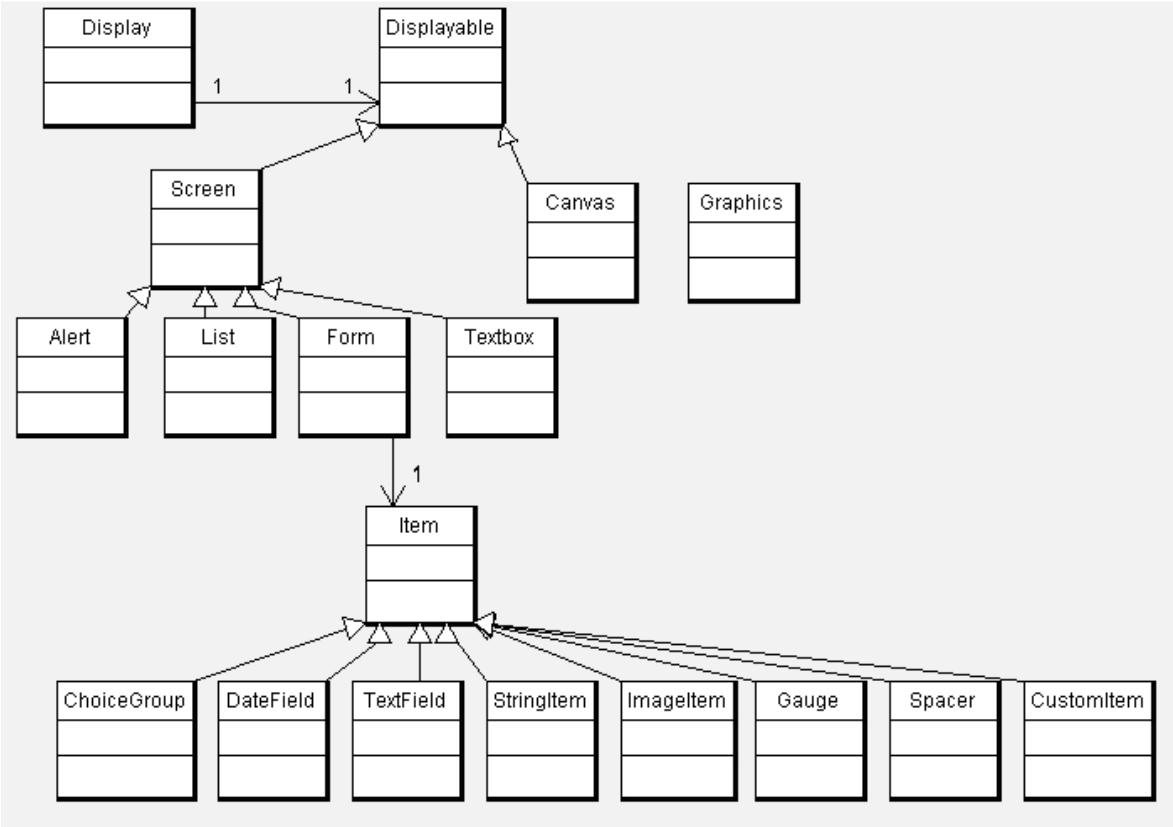


Obrázek 1 – Java ME

2.3.7 Grafické uživatelské prostředí v Java ME

JavaME je určena pro programování aplikací pro zařízení s omezenými hardwarovými prostředky. Tomu odpovídá i možnost programování grafických uživatelských rozhraní. Zaměřím se na konfiguraci CLDC s profilem MIDP, jelikož konfigurace CDC a profil Personal Profile umožňují programovat grafická uživatelská prostředí pomocí balíčku AWT, který se využívá i při programování aplikací na desktopy. Profil MIDP umožňuje programovat jednoduchá uživatelská prostředí pro mobilní telefony. Profil MIDP umožňuje dva přístupy k programování grafických rozhraní. Těmi jsou tzv. high-level API a low-level API. Pro lepší názornost se podíváme, jak jsou

v MIDP organizovány jednotlivé třídy pro práci s grafickými prvky. Na obrázku číslo 2 je UML diagram tříd MIDP obsahující jak třídy low-level, tak i třídy high-level. (YUAN, a další, 2005)



Obrázek 2 – UML diagram grafických prvků lcdui

JavaWorld k tomuto uvádí: Display poskytuje přístup k fyzickému displeji. Každý midlet proto může obsahovat jen jednu instanci třídy Display. Třída Display má důležitou metodu, která zobrazuje na displeji zobrazitelné prvky (potomky třídy Displayable). Displayable prvky reprezentují jednotlivé prvky grafického rozhraní. Na rozdíl od Windows může MIDP zobrazovat pouze jeden Displayable objekt zároveň. K nastavení změny viditelnosti jednotlivých prvků slouží metoda setCurrent(), která přepíná aktivní zobrazované prvky.

Displayable je abstraktní třída, která sama o sobě neposkytuje žádné algoritmy pro referování grafických prvků. Jediné, co z grafických prvků tato třída spravuje, je titulek aplikace a možnost získat rozměry okna aplikace, se kterými můžeme dále pracovat. Hlavní funkcí třídy Displayable je poskytovat základní infrastrukturu pro práci s příkazy. K tomu slouží CommandListener.

2.3.8 Low-level a high-level API

Na obrázku číslo 2 vidíme strukturu balíčku lcdui, jenž obsahuje jak low tak i high-level API. Low-level API je zastoupeno dvěma třídami – Canvas a Graphics. Low-level API představuje práci s grafikou na nejnižší úrovni. Umožňuje tak programátorovi

navrhnout si vlastní grafický vzhled aplikace na úrovni pixelů. Na rozdíl od hig-level je zde potřeba brát ohled na různé typy displejů a jejich rozlišení, jelikož je potřeba většinou specifikovat jednotlivé rozměry. Tím může vzniknout problém s multiplatformovostí aplikace, jelikož existuje mnoho druhů displejů s různým rozlišením.

Třída Canvas poskytuje možnosti pro programování především her. Tato třída umožňuje zpracovávat zejména herní události a obsluhu zmáčknutých kláves. Třída umí spolupracovat i s objekty ze skupiny Screen. Je tedy možné například ve hrách využívat jejích formulářových prvků.

Třída Graphics je určena především pro kreslení jednoduché 2D grafiky, jako jsou například přímky, obdélníky, kruhy, kružnice a oblouky. Třída také umí pracovat s textem a obrázky. Podle MIDP API třída Graphics poskytuje 24bitovu hloubku barev, přičemž každý kanál z RGB barev má k dispocizi 8 bitů. Hloubka barev závisí na zařízení a jeho displeji a také na tom, jaké barevné spektrum je schopný zobrazit.

High-level API je určeno především pro programování aplikací ne příliš závislých na vlastnostech displeje. U většiny prvků z high-level API není možné definovat jejich rozměry, a proto jsou na různých displejích zobrazovány jinak velké. Záleží na typu telefonu, na kterém aplikace běží, jak budou jednotlivé prvky velké. High-level API je zastoupeno třídami ze skupiny Screen. Pro lepší názornost doporučuji podívat se na obrázek číslo 2.

2.4 JavaFX

JavaFX je určena pro programování desktopových aplikací, apletů pro internetové prohlížeče a aplikací pro mobilní zařízení. JavaFX umožňuje programovat aplikace, které poběží zároveň na všech vyjmenovaných platformách. Vize je taková, že naprogramovaný aplet pro internetové stránky půjde spustit v prohlížeči a jednoduchým vytažením z okna prohlížeče pak tento aplet půjde spustit přímo z počítače, popřípadě půjde stáhnout do mobilního zařízení s nainstalovaným prostředím JavaFX a tam opět používat jako Java aplikaci. JavaFX je zatím poměrně nová technologie a proto její zamýšlené použití naráží na několik problémů.

JavaFX běží sice na všech vyjmenovaných platformách, ale zatím je poměrně těžké přenášet aplikace z prohlížeče na PC a mobilní zařízení jinak než přes zdrojové kódy programů. V praxi jsem si ověřil, že je opravdu možné spustit jednu aplikaci v prohlížeči tak v samostatném okně přímo v PC v prostředí Javy.

Pro spuštění JavyFX musíme mít nainstalováno Java Runtime Enviroment 6 v sestavení 10 a vyšší. Toto prostředí nám umožní běh JavaFX aplikací jak přímo v prohlížeči, tak na samotném PC. Toto prostředí přímo obsahuje plugin pro Mozilla Firefox, který umožňuje běh aplikací JavyFX, tak jak je deklarováno vývojáři JavyFX. Tedy spuštěním aplikace jako apletu v prohlížeči s možností přenést tuto aplikaci přímo do prostředí Javy nainstalované na PC. Tento plugin ovšem nespolupracuje s prohlížečem

Opera 10.50, avšak běh samotné aplikace jako apletu v prohlížeči Opera 10.50 možný je. Prohlížeč Google Chrome ve verzi 4.1 umožňuje běh JavaFX apletů bez problémů, ale není možné aplikaci „vzít“ a přetáhnout si ji do počítače, tak jak to umožňuje Mozilla Firefox.

V současné době je možné používat aplikace napsané v JavěFX na mobilních telefonech s operačním systémem Windows Mobile 6.0 a vyšší. (V současné době to je Windows Mobile 6.0, 6.1 a 6.5). Pro běh programu je potřeba stáhnout instalační balíček určený přímo pro Windows Mobile a nainstalovat ho v tomto zařízeni. Po instalaci tohoto prostředí jsme pak schopni na mobilním zařízení spustit aplikace napsané v JavaFX.

Instalaci prostředí JavaFX na mobilním zařízení s operačním systémem Windows Mobile 6.0 a vyšší můžeme provést dvěma způsoby. První způsob je přes ActiveSync 4.5 v případě, že máme na PC nainstalován operační systém Windows XP nebo přes Mobile Device Center na Windows Vista. Druhý způsob je přenesení instalačního balíčku přímo do mobilního zařízení (přes datový kabel, přes bluetooth nebo další rozhraní. Záleží na typu mobilního zařízení a jemu dostupné přenosové technologie.) a jeho spuštěním.

Nainstalování nových aplikací v mobilním zařízení je možné opět provést několika způsoby. První způsob je ten, že si do mobilního zařízení přeneseme spustitelný archiv dané aplikace. Jeho „spuštěním“ se otevře dialog instalace aplikace. Zde si můžeme vybrat, zda chceme danou aplikaci nainstalovat do hlavní paměti zařízení, nebo na paměťovou kartu (pokud ji v zařízení máme). Druhý způsob je spuštění samotného Java prostředí a přes nabídku v menu zadat URL odkazující na archiv požadované aplikace. K tomuto kroku samozřejmě potřebujeme, aby mobilní zařízení mělo přístup k internetu (v dnešních dobách se toto stává celkem běžným standardem). Archiv bude stažen a opět nainstalován do zvoleného umístění.

2.4.1 Grafické uživatelské prostředí v JavaFX

JavaFX je nástroj pro vývoj grafických aplikací pro stolní počítače, webové stránky a mobilní zařízení. Proto její možnosti pro tvorbu grafických uživatelských prostředí jsou značné. Proto mimo jiné JavaFX používá pro psaní programového kódu nový skriptovací jazyk tzv. JavaFX Script. Pomocí tohoto jazyku jsme schopni velice rychle a efektivně vytvářet grafická rozhraní pro naše aplikace. JavaFX Script je jazyk deklarativní, takže pomocí něj je velice jednoduché vytvořit grafická prostředí. Jako příklad zde uvedu jednoduché přidání kruhu do výsledného grafického rozhraní. JavaFX má hlavní část kódu označenou jako Stage. Stage nahrazuje u JavaFX klasickou funkci main. Stage obsahuje mimo jiné Scene. Scéna je určená pro vytváření grafického prostředí aplikace a tvoří základní prvek pro jeho vytváření. Pro podrobnější informace doporučuji shlédnout tutoriály na stránkách společnosti Oracle.

Následující ukázky zdrojových kódů jsou výtažek z tutoriálů na stránkách Oracle. (Oracle, 2010) Zacházení s JavaFX Scriptem je jednoduché. Pro přidání kruhu do scény napíšeme:

```
Circle{}
```

Tím jsme zadeklarovali přidání kruhu do výsledného grafického rozhraní. V tento moment se však ještě nebude zobrazovat, jelikož nemá nadeklarované žádné rozměry. To provedeme následovným krokem:

```
Circle {
centerX: 150
centerY: 150
radius: 100
}
```

Těmito jednoduchými úpravami jsme kruh umístili 150 pixelů od levého okraje a 150 pixelů od horního okraje scény (nebo také okna aplikace). Kruh bude mít poloměr 100 pixelů.

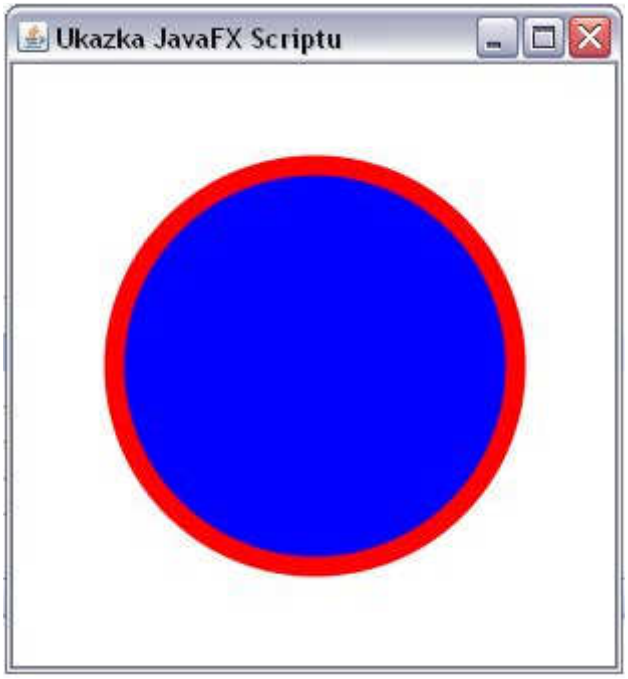
V následujícím kroku kruh ještě „vylepšíme“ přidáním okraje a vybarvením kruhu a jeho okraje předdefinovanými barvami z třídy javafx.scene.paint.Color, a to následovně:

```
Circle {
centerX: 150
centerY: 150
radius: 100
fill: Color.BLUE
stroke: Color.RED
strokeWidth: 10.0
}
```

Toto rozšíření obarví vnitřek kruhu modrou barvou, přidá 10 pixelů široký okraj a vyplní ho červenou barvou. Pro úplný přehled přidám celý kód programu, který nám tento kruh zobrazí.

```
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
Stage {
      title: "Ukazka JavaFX Scriptu"
scene: Scene{
height:300
      width:300
content: [
      Circle{
      centerX:150
      centerY:150
      radius:100
      fill:Color.BLUE
      strokeWidth: 10
      stroke:Color.RED
      }]}}
```

Tento kód nám zobrazí okno aplikace 300 pixelů široké a 300 pixelů vysoké. V něm bude jediný prvek, a to výše popsaný kruh. Jak celé okno této „aplikace“ bude vypadat po spuštění, to můžeme vidět na obrázku 3.



Obrázek 3 – Výsledné okno ukázky

Z výše uvedeného příkladu je zřejmé, že vytváření grafických prvků a jejich vkládání do scény je velice jednoduché a dá se říci, že i celkem intuitivní. Stejným způsobem se do scény vkládají aktivní prvky ovládání jako jsou tlačítka a posuvníky. Uvedu jednoduchý příklad skriptu pro vložení tlačítka.

```
Button {
text: "Tlacitko"
height:30
width:60
onMousePressed: function( e: MouseEvent ):Void {
}
}
```

Tímto jsme do scény přidali tlačítko vysoké 30 pixelů a široké 60 pixelů. Tlačítko bude popsané textem „Tlacitko“ a bude reagovat na stisknutí tlačítkem myši (v případě mobilního telefonu s dotykovým displejem bude reagovat na dotyk). Z tohoto skriptu je zřejmé, že i přidávání aktivních grafických prvků pro ovládání aplikace je jednoduché. Na obrázku 4 můžeme vidět ukázku ovládacích prvků v JavaFX. Jak je vidět, JavaFX nám poskytuje širokou paletu prvků pro vytváření grafického uživatelského prostředí s aktivními ovládacími prvky.

Obrázek 4 – Ukázka ovládacích prvků

Opět platí to, co jsem zmiňoval výše. Všechny zde prezentované prvky lze do scény velice jednoduše přidat a nastavit jim požadované vlastnosti, jako je šířka, výška, textový popisek atd. Zde se musím zmínit o tom, že všechny tyto prvky pro ovládání je možné použít i pro mobilní zařízení s dotykovým displejem. Na zařízení s dotykovým displejem se tyto prvky budou chovat obdobně. Jen je potřeba dát pozor, aby tlačítka byla dostatečně velká, aby se dala bez problémů stisknout prstem. (Dnešním trendem je právě u těchto zařízení prosazovat ovládání pouze prsty a aplikace ovládané stylusem se stávají spíše menšinové. Tento trend dnešních aplikací navíc podpořily kapacitní displeje, ke kterým sice jsou vyráběny stylusy, ale ty nejsou v ovládání telefonu o moc přesnější než ovládání prstem.)

Na obrázku 5 se můžeme podívat na ukázku několika základních jednoduchých geometrických obrazců, pomocí kterých opět můžeme tvořit grafická rozhraní. Pomocí těchto obrazců si tedy můžeme vytvářet vlastní interaktivní či jiné grafické prvky pro naše grafické uživatelské rozhraní.

Obrázek 5 – Ukázka základních tvarů

JavaFX nám umožňuje těmto námi vytvořeným prvkům velice jednoduše zajistit interaktivitu. Můžeme si tedy například nakreslit „barevná“ tlačítka s ikonkami pro naši aplikaci. Také nám kontejner Path z balíčku Javafx.scene.shape umožňuje vytvářet složitější geometrické tvary. Pomocí Path s nimi pak můžeme pracovat jako s jedinou komponentou, což samozřejmě velice zjednodušuje práci s těmito složitými obrazci.

JavaFX také umožňuje pro programování grafických uživatelských rozhraní používat takzvané rozložení scény (layout). To nám umožní Javafx.scene.layout, která obsahuje HBox a VBox. Kontejner HBox nám pomáhá při horizontálním rozložení grafických prvků, zejména pak jejich rozestupů mezi sebou v rámci jednoho řádku. Jestliže HBox umožňuje pozicování v horizontálním směru, pak VBox má na starosti totéž, ale ve vertikálním směru. Pomocí těchto dvou kontejnerů jsme tedy schopni rozložit grafické uživatelské rozhraní do pomyslné mřížky. Samozřejmě jde tyto dva kontejnery různě kombinovat a nejsou jedinými možnostmi, jak dosáhnout výsledného rozložení grafických prvků v okně aplikace.

Pro vytváření grafických prvků pro aplikace nám JavaFX také umožňuje pracovat s obrázky. K tomuto účelu slouží balíček Javafx.scene.image. Pomocí tohoto balíčku můžeme pracovat s obrázky ve formátech: jpg, gif a png. Načtení obrázku do aplikace nám zajišťuje třída Image, zatímco samotnou práci a zobrazení načteného obrázku pak třída ImageView. Načtení obrázku do aplikace se provádí přes atribut URL. Tento atribut nám tedy umožňuje obrázky umístit do specifické složky a odtamtud je pak následně načítat do aplikace.

Na obrázky je samozřejmě možné aplikovat sadu transformací. Transformace je možné aplikovat i na ostatní prvky v grafickém rozhraní, proto vznikla třída Javafx.scene.transform. Tyto transformace, vybrané z JavaFX API 1.2.1, jsou uvedeny v tabulce číslo 1.

Tabulka 1 – Transformace v JavaFX

Transformace	Popis
Affine	Afinní transformace složená z několika transformací jako posunutí, otáčení a změna měřítka. Tato transformace se obvykle nepoužívá a v aplikacích se používají níže uvedené transformace zvlášť.
Rotate	Umožňuje obrázek otáčet.
Scale	Změna měřítka obrázku.
Shear	Zkosení v osách x a y.
Translate	Pousnutí o určitý vektor.

Tady je dobré podotknout, že transformace translate se používá celkem běžně pro pozicování prvků grafického prostředí. Pomocí této transformace jsme tedy schopni rozvrhnout celé rozložení grafického uživatelského prostředí.

Další možností pro programování grafických rozhraní v JavaFX je použití animace. Práci s animací objektů nám zajišťuje balíček Javafx.animation. Pomocí tohoto balíčku jsme schopni pro naši aplikaci naprogramovat jednoduché animace grafického prostředí. Animace se skládá z časově rozvrhnutých transformací objektu. O časový plán se stará právě balíček Javafx.animation. Transformace objektu pak má na starosti balíček transform, o kterém se zmiňuji výše. Můžeme proto třeba naprogramovat posun obrázku po okně aplikace. Jeho plynulost, velikost posunu a trvání specifikujeme pomocí kontejneru Timeline. Tento kontejner zajišťuje celkovou správu proměnných použitých pro animaci. Timeline se proto stará například o hodnoty souřadnic objektu, který se má po okně aplikace posouvat. Timeline je složena z jednoho nebo více KeyFrame. KeyFrame se chová stejně jako jedno políčko filmu (k tomuto se ještě později vrátím). Pomocí KeyFrame jsou zaznamenány jednotlivé změny pro provedení požadované animace. Jako příklad si uvedeme krátký kód, na kterém bude lépe vidět, jak funguje princip Timeline:

```
Timeline {
      keyFrames: [
      KeyFrame{
      time: 0s
      values: x => 0.0
      },
      KeyFrame{
      time: 4s
      values: x => 158.0 tween Interpolator.LINEAR
      }
      ]
}.play();
```

Zde můžeme vidět, že Timeline se skládá ze dvou KeyFrame. První z nich nám definuje výchozí stav. Timeline tedy začne s hodnotou x=0. Druhý KeyFrame má čas nastavený na 4s. V tomto případě je tento KeyFrame koncovým stavem. Tím tedy dostáváme změnu hodnoty x z hodnoty 0 na hodnotu 158 za 4 sekundy s lineárním přírůstkem hodnoty x. Tuto časovou osu můžeme například použít pro posouvání objektu po obrazovce ve směru jedné z os nebo po úhlopříčce pomyslného čtverce o straně 158 pixelů a to tak, že hodnotu x budeme předávat transformaci translate. Timeline se postará o to, aby se hodnota x měnila, a translate pak o její posun. Změna hodnoty x probíhá jednosměrně a v tomto případě proběhne také jenom jednou. Timeline však samozřejmě umožňuje tuto časovou osu přehrávat i opakovaně, popřípadě v nekonečné smyčce. Ale tím pořád dostáváme jednosměrnou inkrementaci hodnoty x. Objekt by se tedy přesunul na konec dráhy a pak by se skokově vrátil na její začátek. Aby se nemusel připisovat další KeyFrame pro návrat hodnoty x na 0, je možnost zapnout přehrávání v módu autoreverse a tím docílit i zpětné dekrementace hodnoty x. Zároveň tím zajistíme, že se objekt bude neustále plynule posouvat zleva doprava a zpět. K tomuto slouží výše zmíněný atribut autoReverse. AutoReverse je booleovský atribut a má hodnotu true pro zpětné provádění Timeline.

Tím se dostáváme opět k tomu, že KeyFrame se chová podobně jako filmové políčko. Není tomu úplně tak. Jelikož KeyFrame obsahuje změny, které se mají provést, a způsob, jakým se mají provést (zajišťuje Interpolator). Zatímco pro film je potřeba natočit políčko pro každou fázi pohybu, KeyFrame určuje začátek a konec a o ostatní se už stará Timeline.

3 Prostředky pro programování mobilních zařízení.

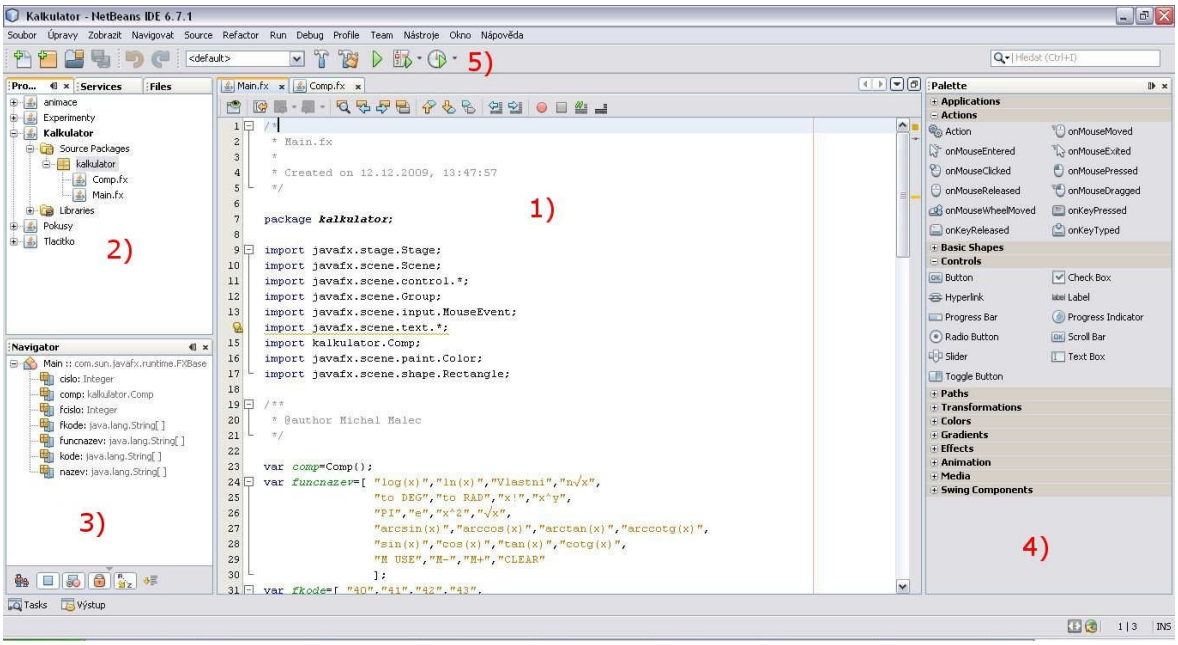
Pro programování mobilních zařízení potřebujeme nejdříve nainstalovat vývojové prostředí, které umí pracovat se zvolenou verzí Javy. Pro vývoj aplikací v Java ME nám bude stačit stáhnout a nainstalovat Sun Java Wireless Toolkit. Toto prostředí existuje ve dvou verzích. První verze je Sun Java Wireless Toolkit 2.5.2 for CLDC a jak název vypovídá, je určeno pro vývoj aplikací pod konfigurací CLDC a profilem MIDP. Druhá verze je Sun Java Toolkit 1.0 for CDC, jenž je určená pro vývoj aplikací na konfiguraci CDC. Obě tyto verze je možné stáhnout a nainstalovat pomocí Java Platform Micro Edition Software Development Kit 3.0, což je balíček nástrojů zastřešující oba předcházející nástroje. Tento nástroj je základem pro vývoj aplikací v Java ME. Mnohem lepší je pracovat v některém z pokročilejších vývojových prostředí. Pro vývoj aplikací v JavaFX je potřeba mít nainstalován balíček Java Development Kit 6 v sestavení 7 nebo vyšší. Pro tuto edici není dostupné žádné „základní“ vývojové prostředí a proto je nutné použít jiné, jež s touto edicí Javy umí pracovat. Proto je třeba vhodné použít vývojové prostředí Eclipse, které po nainstalování základních balíčků Javy a doinstalování pluginů pro samotné vývojové prostředí umožňuje vyvíjet aplikace v obou edicích Javy. Druhým takovým prostředím je NetBeans, které má již obě dvě edice Javy integrované do sebe a po nainstalování samotného prostředí je možné začít hned vyvíjet aplikace. Proto jsem si ho vybral jako nástroj pro vývoj kalkulátoru na mobilní zařízení.

3.1 Vývojové prostředí NetBeans

Pro vývoj aplikace kalkulátoru pro mobilní zařízení jsem si vybral vývojové prostředí NetBeans 6.7.1 licencované pod duální licencí Common Development and Distribution License (CDDL) a GNU General Public License version 2 s Classpath exception. NetBeans v této verzi umožňují vyvíjet aplikace v široké paletě jazyků, počínaje Javou ve všech verzích, přes PHP a C/C++ konče. Prostředí jsem volil hlavně proto, že umí pracovat jak s Java ME v obou konfiguracích, tak s JavaFX. Toto prostředí je možné stáhnout v několika verzích. Jednotlivé verze jsou zaměřené na určité programovací jazyky. Je tedy možné třeba stáhnout NetBeans pouze pro vývoj v jazyce Java. Mnohem výhodnější ale je instalovat prostředí kompletní, umožňující programovat a vyvíjet aplikace ve výše zmiňovaných jazycích.

Výhodou tohoto prostředí je, že po nainstalování je hned možné začít vyvíjet. Instalací prostředí nainstalujeme i potřebné nástroje pro vývoj v podporovaných jazycích. Tyto nástroje jsou integrované do samotného prostředí, takže prostředí se chová stejně pro různé programovací jazyky.

Na obrázku číslo 6 můžeme vidět okno vývojového prostředí NetBeans.



Obrázek 6 – Vývojové prostředí NetBeans

Okno prostředí je rozděleno na několik částí. Jednotlivé části si podrobněji popíšeme. Nahoře kromě obvyklé nabídky příkazů (Soubor, Úpravy atd.) je několik tlačítek určených zejména pro rychlé spuštění a kompilování projektu. Na obrázku jsou označena číslem 5. Největší plochu okna prostředí zabírá samotný editor kódu programu. Na obrázku je označen číslem 1. Editor umožňuje velice efektivní psaní kódu, jelikož v průběhu psaní nabízí možnosti automatického doplnění kódu. Sám nabízí dostupné třídy, případně možné knihovny na importování do zdrojového kódu. Samozřejmostí je zvýraznění a barevné odlišení syntaxe, zvýraznění souvisejících závorek. Mezi další funkce patří například možnost zazáložkovat si určité místo v kódu a poté se k němu rychle vrátit pomocí navigátoru záložkami. Mezi další vlastnosti patří práce se samotným zdrojovým kódem, což je vyhledávání, nahrazování nebo případně formátování samotného textu kódu. Druhou částí vývojového prostředí NetBeans jsou záložky pro procházení jednotlivých souborů v projektu. Na obrázku číslo 6 jsou označené číslem 2. První záložka je určená pro přehledné procházení otevřených projektů NetBeans a důležitých souborů v nich. Soubory jsou přehledně členěny do stromové struktury, včetně jejich zařazení do příslušných package. Další oblastí je přehled deklarovaných tříd, proměnných a funkcí. Tento navigátor umožňuje rychlý pohled na deklarované třídy, atributy třídy, její metody anebo obecně deklarované proměnné a funkce. U každé položky v seznamu je obsažen její typ a ikonka znázorňující, zda jde o veřejnou nebo neveřejnou funkci/proměnnou. Samozřejmostí je, že při dvojitém poklepání myší se kurzor v editoru přesune na deklaraci dané položky.

Celou pravou část okna zabírá paleta. Na obrázku číslo 6 je označená číslem 4. Na tomto panelu máme přehledně do jednotlivých kategorií zařazené grafické komponenty,

jež můžeme „přetáhnout“ do editoru kódu a zde dojde k vložení kódu potřebného pro základní zobrazení dané grafické komponenty.

Jak jsem předeslal o kousek výše, v okně vývojového prostředí NetBeans je možné procházet jednotlivé soubory v projektu. Tento panel je označen na obrázku číslo 6 číslem 2. Na skutečnou adresářovou strukturu projektu se podíváme na obrázku číslo 7.

Obrázek 7 – Adresářová struktura projketů

Na obrázku číslo 7 je zachycen kompletní adresářový strom projektu. V nejnadřazenější složce s názvem NetBeansProjects jsou složky jednotlivých projektů, které jsme pomocí NetBeans vytvořili a uložili do výchozího umístění. V každé složce projektu je pak velké množství složek, kam si NetBeans ukládají další soubory spojené s projektem. Nejdůležitějšími složkami v tomto stromě jsou složky dist a src. Ve složce src jsou uložené podle balíčků zdrojové kódy aplikace. Ve složce dist jsou pak výsledné zkompilované soubory, v našem případě pro vývoj aplikace v jazyce Java to jsou archivy s příponami jar a jad. Ve složce build a v jejích podadresářích compiled, optimized a verified jsou dočasné soubory pro případné zpětné změny v kódu a jiné pomocné soubory, které si NetBeans ukládá v průběhu práce na projektu.

4 Aplikace Kalkulátor v mobilním zařízení

Každý mobilní telefon má v sobě předinstalovanou kalkulačku. Tato kalkulačka většinou stačí na základní matematické operace. Mobilní telefon máme skoro neustále u

sebe a tato kalkulačka obvykle stačí na většinu výpočtů, se kterými se můžeme během pracovního dne setkat. Stejná situace je i na zařízení HTC Touch HD, i zde je předinstalovaná systémová kalkulačka zvládající základní matematické operace. Proto je nutné pro složitější výpočty použít jinou kalkulačku. Tato práce si klade za cíl problém vyřešit tak, aby na tomto zařízení šla používat kalkulačka se širšími možnostmi matematických výpočtů. V tabulce číslo 2 se můžeme podívat na technické parametry tohoto zařízení.

Tabulka 2 – Technické parametry HTC Touch HD

HTC Touch HD	
Procesor	Qualcomm® MSM 7201A™ 528 MHz
Operační systém	Windows Mobile® 6.1 Professional
Paměť	ROM: 512 MB RAM: 288 MB
Displej	3,8palcový dotykový TFT-LCD displej s rozlišením WVGA (480 X 800)
Rozměry (D x Š x T)	115 mm x 62,8 mm x 12 mm

Z této tabulky je zřejmé, že se jedná o poměrně výkonné zařízení. Proto jsem se rozhodl naprogramovat kalkulačku v JavaFX tak, aby využívala velkého dotykového displeje tohoto zařízení. Nejprve se budu zabývat problematikou všech kalkulátorů, a to je práce s velkými a malými čísly v Javě.

4.1 Práce s velkými a malými čísly v Javě

4.1.1 Velká čísla v Java SE

Java SE (Second Edition) pro práci s velkými čísly obsahuje několik tříd pro práci s nimi. První z nich je Java.math.BigDecimal. Třída BigDecimal implementuje neexponenciální integer základ a 32bitový integer exponent. Číslo je tedy ve formátu: $zaklad \times 10^{32bit\ exponent}$. Další třídou pro velká čísla je Java.math.BigInteger. Tato třída dovolí konstruovat BigInteger z pole obsahujícího dva prvky, které bitově reprezentují BigInteger. Umožňuje také konstruovat BigInteger z textového řetězce String.

Pro práci s malými čísly můžeme použít třídu Short, která obsahuje datový typ short. Pokud chceme pracovat s opravdu malými čísly, jde použít BigDecimal, kdy exponent bude záporné číslo.

4.1.2 Velká čísla v Java ME

Práce s velkými čísly v Java ME je složitější. Záleží totiž na tom, na jaké platformě budeme programovat. Uvedu zde jednotlivé konfigurace a profily, vždy se všemi třídami, které umožňují práci s velkými čísly.

CLDC ve verzi 1.0 má pro interpretaci velkých čísel pouze třídu Java.lang.Long. Tato třída dovoluje tedy pracovat s čísly do maximální hodnoty 9 223 372 036 854 775 807 a minimální hodnoty -9 223 372 036 854 775 808.

CLDC verze 1.1 pro práci s velkými čísly přidává třídu Java.lang.Double. Pomocí této třídy jsme schopni zapsat čísla do maximální hodnoty 1.7976931348623157 E 308.

CDC nabízí pro práci s velkými čísly tyto třídy: Java.math.BigInteger – kromě všech operátorů třídy Integer obsahuje i operace pro aritmetické výpočty, testování, generování a manipulaci s jednotlivými bity. Třída Java.lang.Double je shodná s třídami konfigurací CLDC. Stejně tak obsahuje třídu Java.lang.Long, opět shodnou se třídou z CLDC.

Doplňující profily MIDP 1.0 a MIDP 2.0 používají shodné třídy pro interpretaci velkých čísel. MIDP 1.0 tedy třídu Java.lang.Long a MIDP 2.0 kromě této třídy ještě třídu Java.lang.Double.

4.1.3 Malá čísla v JavaME

Pro malá čísla v Java ME se používá třída Java.lang.Short bez rozdílu konfigurace (CLDC nebo CDC) nebo profilu MIDP 1.0 a MIDP 2.0. Short obsahuje základní datový typ short, který je 16bitové číslo se znaménkem.

4.1.4 Velká čísla v JavaFX

Problém velkých čísel v JavaFX je poněkud složitější. JavaFX nemá žádnou speciální třídu, nebo datový typ pro tento účel. JavaFX má následující datové typy pro práci s číselnými hodnotami:

Tabulka 3 – Datové typy JavaFX

JavaFX	Odpovídající typ proměnné v JavaSE
Number	double / java.lang.Double
Integer	int / java.lang.Integer

Datový typ Number je tedy 64bit číslo s plovoucí desetinnou čárkou. Integer je klasicky 32bit číslo se znaménkem. Takže je celkem zřejmé, že pro uchovávání velkých čísel v programu psaném v JavaFX je určen právě datový typ Number.

Pro aplikace určené na mobilní platformu (telefony, komunikátory, PDA) je vhodné používat typ Number jedině tehdy, pokud ho opravdu potřebujeme. Number by mohl

v případě neúplného využití zbytečně zabírat místo v paměti a běh aplikace by se tím mohl zpomalovat.

Jak bylo předesláno ve vlastnostech JavaFX, je možné do programu psaného v JavaFX skriptu vložit jakoukoliv javovskou třídu. Toto se týká i obalových tříd a datových typů pro třídy Double a Long a datové typy double a long. JavaFX s nimi umí pracovat, ale mohou nastat problémy při práci s nimi. Problémové jsou zejména převody mezi datovými typy JavaFX (Number a Integer) a datovými typy z Java SE, v našem případě datový typ long. Mobilní aplikace s typem long umí pracovat, ale problém nastává při převodu do JavaFX proměnné typu String.

4.1.5 Malá čísla v JavaFX

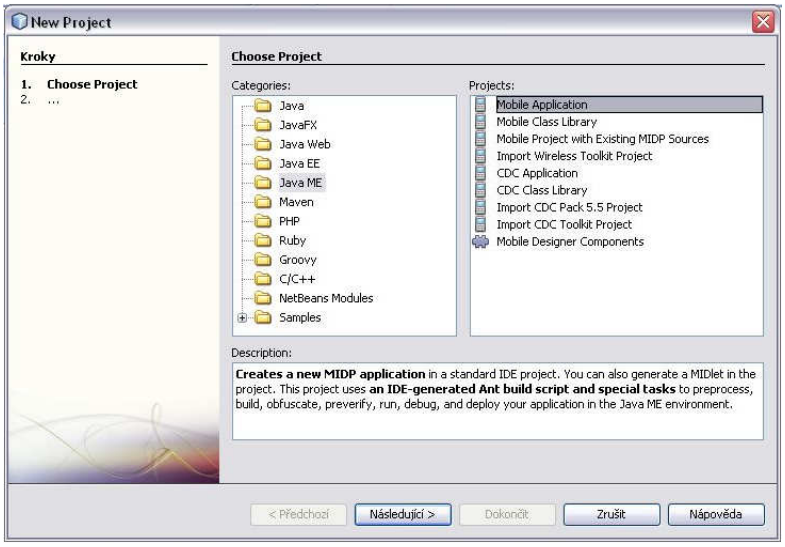
Jak je vidět v tabulce 3, JavaFX má pouze dva číselné datové typy. Takže pokud chceme pracovat s malými čísly, musíme použít datový typ Integer. Jinou možnost nám JavaFX nenabízí.

Stejně jako u velkých čísel platí, že JavaFX umí pracovat s třídou Short a datovým typem short. I zde ovšem platí omezení, podobné tomu u tříd Long a Double. V některých případech se můžeme setkat s nemožností tyto typy použít.

4.2 Ukázka tvorby jednoduchého kalkulátoru v Java ME

Jelikož tato práce má vyřešit problém s kalkulátory na mobilních zařízeních, budu se nejprve krátce věnovat kalkulátoru v Java ME pro konfiguraci CLDC a profil MIDP. Zařízení HTC Touch HD obsahuje předinstalované prostředí Javy – Jbed od firmy Esmertec. Toto prostředí je emulátor pro výše zmíněnou konfiguraci a profil.

Nejdříve musíme založit nový projekt ve vývojovém prostředí NetBeans. Toho dosáhneme následujícími kroky. Stiskneme tlačítko New Project a zobrazí se nám dialog, který můžeme vidět na obrázku číslo 8.



Obrázek 8 – Založení nového projektu

V tomto okně dialogu vybereme, jaký projekt chceme vytvořit. V našem případě je to Java ME. V pravé části dialogu se nám zobrazí seznam typů projektů, které můžeme vytvářet. Vybereme Mobile Application a potvrdíme stisknutím tlačítka Následující.

V další části tvorby nového projektu budeme dotázáni na jméno projektu. Dále pak máme možnost zvolit si cestu, kam bude projekt uložen. Výchozí nastavení této cesty směřuje do složky NetBeansProjects, jak bylo ukázáno výše v části o vývojovém prostředí NetBeans a jeho adresářové struktuře ukládání projektů. V tomto okně nesmíme zapomenout na zaškrtnutí volby Set as Main Project. Tím se nám tento projekt nastaví jako hlavní a budeme s ním moci pracovat. Pro tento typ aplikace je vhodné si nechat vygenerovat Hello MIDlet. Tím nám NetBeans vygeneruje základní kód a od začátku spustitelný MIDlet. Opět dialog potvrdíme stisknutím tlačítka Následující.

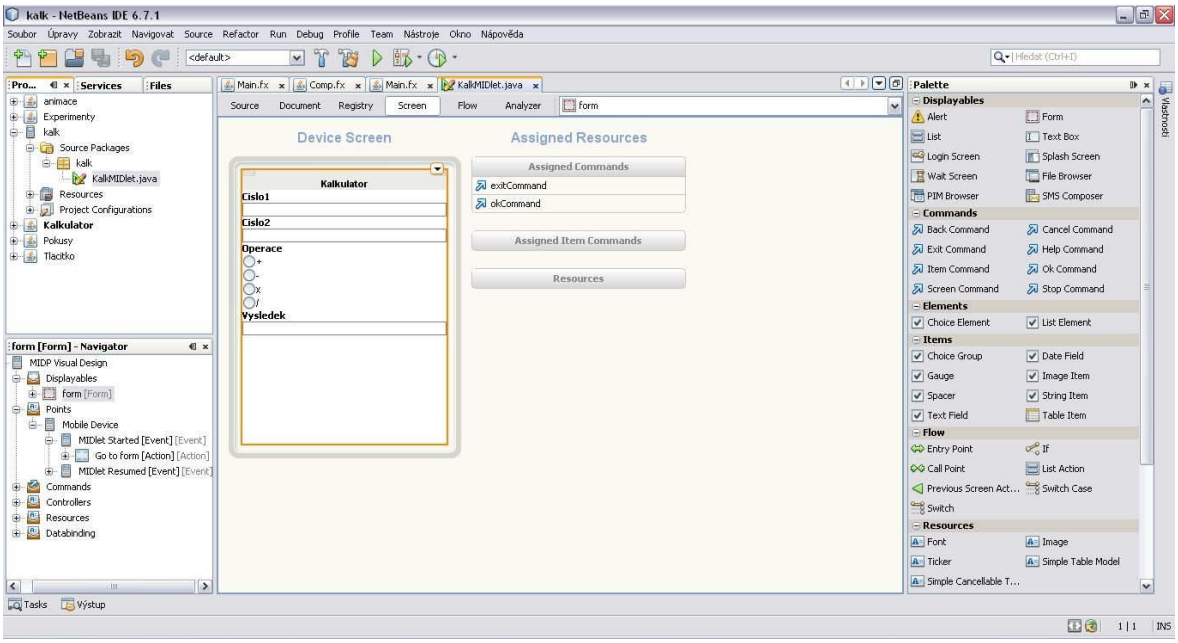
V následujícím kroku musíme nastavit verzi konfigurace a verzi profilu. Toto nastavení záleží na telefonu, pro který má být daná aplikace vyvíjena. Zařízení HTC Touch HD umí pracovat s konfigurací CLDC ve verzi 1.1 a profilem MIDP 2.1, proto jej zvolíme. V tomto okně ještě dále nastavíme emulátor, ve kterém se bude námi vyvíjený MIDlet spouštět pro testování chodu aplikace. NetBeans ve verzi 6.7.1 při instalaci také předinstalují balík Java Platform Micro Edition SDK 3.0, proto tuto volbu necháme. Poslední, co je potřeba udělat, je zvolit si typ emulovaného zařízení. Po nastavení všech těchto voleb můžeme stisknout tlačítko Dokončit. Tím se nám vytvoří projekt a vygeneruje jednoduchý MIDlet. Můžeme začít vytvářet vlastní MIDlet.

Druhou možností je nevytvářet Hello MIDlet, ale v tomto případě se nám nevygeneruje žádný počáteční kód a budeme si tedy muset vše udělat ručně. Doporučuji proto tento Hello MIDlet vytvořit, jelikož těch pár okamžiků, které strávíme přejmenováním souborů a MIDletu se nám vyplatí.

Zvolením možnosti vygenerování Hello MIDletu se automaticky vygeneruje základní struktura MIDletu. Jsou to všechny základní metody nutné pro spuštění MIDletu. Vygenerovanou základní strukturu MIDletu si můžeme prohlédnout níže:

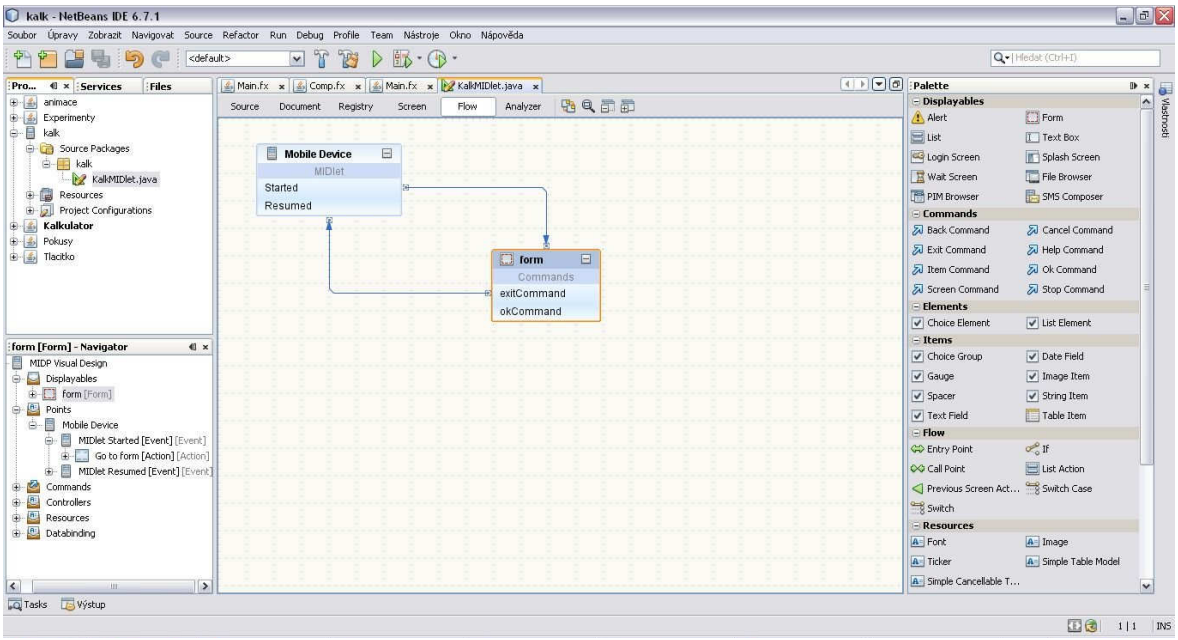
```
public class KalkMIDlet extends MIDlet implements CommandListener {
private boolean midletPaused = false;
public KalkMIDlet() {
}
public void startMIDlet() {
switchDisplayable(null, getForm());
}
public void resumeMIDlet() {
} public Display getDisplay () {
 return Display.getDisplay(this);
}
public void exitMIDlet() {
switchDisplayable (null, null);
destroyApp(true);
notifyDestroyed();
}
public void startApp() {
if (midletPaused) {
resumeMIDlet ();
} else {
initialize ();
startMIDlet ();
}
midletPaused = false;
}
public void pauseApp() {
midletPaused = true;
}
public void destroyApp(boolean unconditional) {
}}
```

Po vytvoření projektu musíme nejprve pomocí grafického editoru vytvořit obrazovku budoucího MIDletu. Máme zde k dispozici celou paletu nástrojů pro tvorbu grafického uživatelského prostředí. Obrazovka vytváření grafického prostředí je vidět na obrázku číslo 9.



Obrázek 9 – Vytvoření obrazovky kalkulátoru

Vytvoříme formulář. Do něho přidáme textové vstupy, skupinu radiových tlačítek pro volbu operace a jako poslední prvek pole výsledek, kam se bude zapisovat výsledek výpočtu. U vstupu prvních dvou operandů nastavíme v revizi objektu numerický vstup, tzn. pokud do nich začneme zadávat vstup z klávesnice telefonu, budou se zobrazovat čísla daným klávesám přiřazená. Jako poslední musíme přidat uživatelský příkaz, který bude formulář potvrzovat a provádět samotný výpočet. Vývojové prostředí nám zatím generuje veškerý kód potřebný pro zobrazení tohoto formuláře. Dalším krokem je nastavení životního cyklu MIDletu. Proto se přepneme do záložky Flow. Tuto obrazovku můžeme vidět na obrázku číslo 10.



Obrázek 10 – Obrazovka Flow

Zde je schematicky vidět, jak se budou přepínat jednotlivé displayable prvky v závislosti na stavu MIDletu. Jelikož jsme tento projekt vytvářeli z Hello MIDletu, je tato obrazovka už vygenerována. Po startu MIDletu se zobrazí formulář. Námi přidaný uživatelský příkaz nemění displayable prvky, takže zde je vše v pořádku. Na této obrazovce by se nastavovala změna zobrazovaných prvků v případě, že by se po potvrzení výpočtu měl výsledek zobrazit jako vyskakovací informační okno.

Nyní na řadu přichází už samotné naprogramování chování námi přidaného příkazu. Zatím za nás kód generoval NetBeans. Do metody public void commandAction(Command command, Displayable displayable), která obsluhuje příkazy, musíme odprogramovat akci, která se má stát v případě, že zvolíme možnost Vypočti (potvrzení formuláře). To provedeme dopsáním následujícího kódu:

```
cis1=Double.valueOf(Cislo1.getString()).doubleValue();
cis2=Double.valueOf(Cislo2.getString()).doubleValue();
if(choiceGroup.isSelected(0)){vysl=cis1+cis2;}
if(choiceGroup.isSelected(1)){vysl=cis1-cis2;}
if(choiceGroup.isSelected(2)){vysl=cis1*cis2;}
if(choiceGroup.isSelected(3)){vysl=cis1/cis2;}
Vysledek.setString(Double.toString(vysl));
```

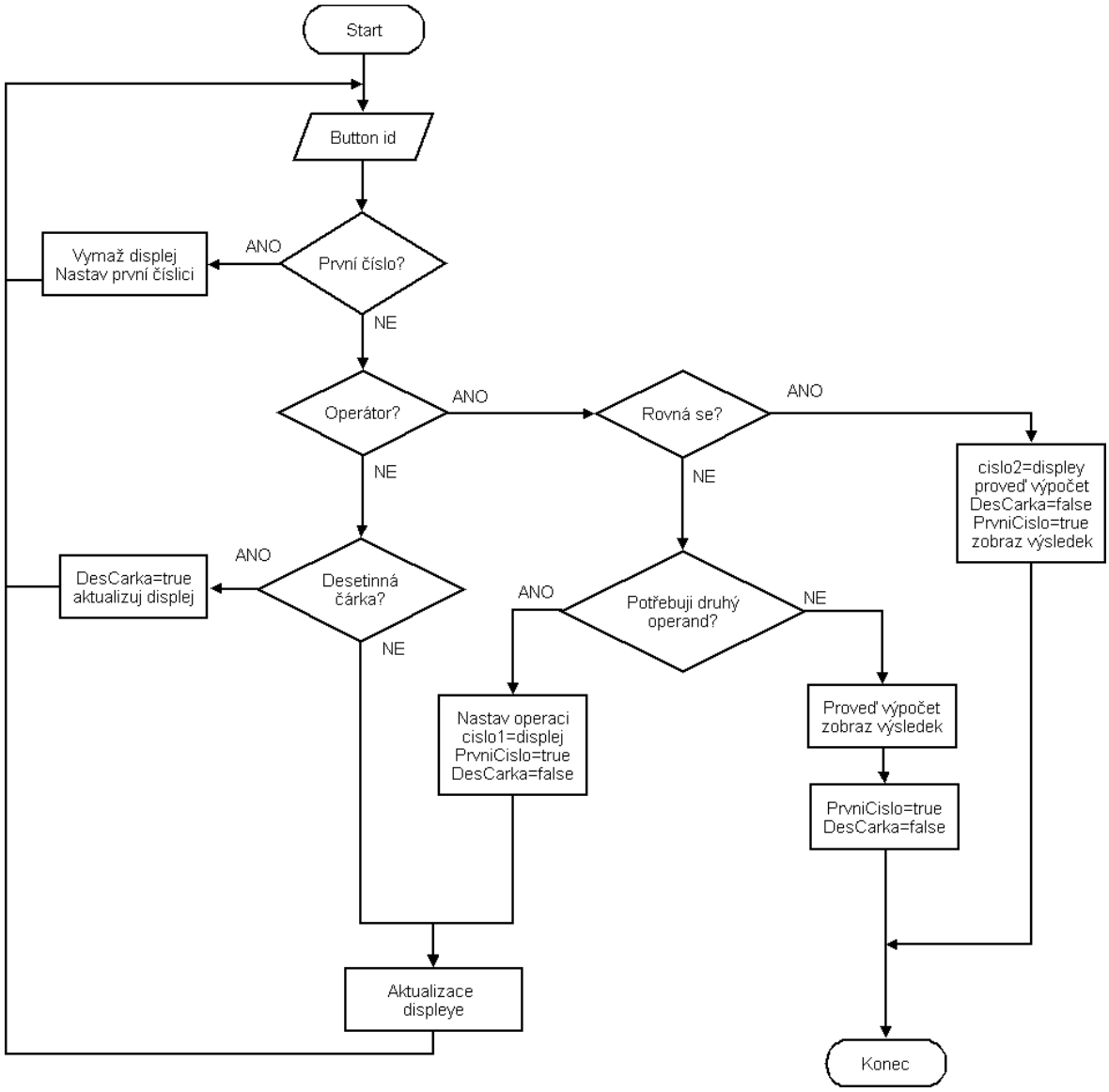
Proměnné cis1, cis2 a vysl jsou typu double. Tato část kódu obstarává veškerou vnitřní logiku této jednoduché kalkulačky. V momentě potvrzení formuláře se nejdříve do proměnných cis1 a cis2 uloží double hodnoty získané z textového vstupu Cislo1 a Cislo2. Poté se zjistí, které z radiových tlačítek bylo vybráno, a podle toho se provede příslušná matematická operace. Úplně nakonec se do pole Vysledek vloží na řetězec převedený obsah proměnné vysl. Obrázek číslo 11 ukazuje, jak tento kalkulátor vypadá spuštěný v emulátoru ve vývojovém prostředí NetBeans a na zařízení HTC Touch HD.

Obrázek 11 – Porovnání Java ME kalkulátoru

4.3 JavaFX Kalkulátor

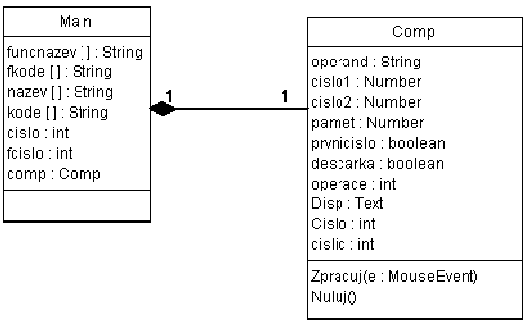
Jak jsem již předeslal výše, samotný kalkulátor jsem se rozhodl naprogramovat v JavaFX. Zařízení HTC Touch HD má velký dotykový displej s rozlišením 480x800 pixelů, což je velká plocha, takže pomocí grafických možností JavaFX jde udělat klasický tlačítkový kalkulátor. JavaFX obsahuje i třídu Math, konkrétně Javafx.util.Math. Tato třída obsahuje složitější matematické operace. Příkladem mohou být goniometrické funkce, mocnina, odmocnina. Pomocí této třídy se tedy dají vytvořit potřebné funkce pro kalkulátor.

Nejprve bylo potřeba vyřešit problém vnitřní logiky kalkulátoru. Konkrétně to, jak se bude kalkulátor chovat po stisku tlačítka, jak bude vyhodnocovat zmáčknutá tlačítka a jak se budou provádět jednotlivé operace. Řešením tohoto problému je vývojový diagram, který si můžeme prohlédnout na obrázku číslo 12. Diagram znázorňuje, jakým způsobem se vyhodnocuje příchozí kód zmáčknutého tlačítka.



Obrázek 12 – Diagram vnitřní logiky

Dále bylo potřeba vyřešit, jaké třídy budou v samotném programu kalkulátoru. Řešením je implementovat dvě třídy. První třída se bude starat o zobrazování veškeré grafiky. Druhá třída pak bude obstarávat samotnou práci kalkulátoru. Bude tedy obsluhovat zmáčknutá tlačítka a provádět výpočty podle diagramu, který je uvedený výše. UML Class diagram je na obrázku číslo 13.



Obrázek 13 – UML Diagram tříd

Na tomto diagramu je konečný stav tříd tak, jak jsou implementovány v konečné verzi kalkulátoru. Třída Main obsahuje čtyři atributy pole String. Tato pole slouží jako zdroj pro generování tlačítek. Princip generování tlačítek bude rozebrán později. Pole funcnazev obsahuje všechny popisky tlačítek, tedy text, který se na tlačítkách zobrazuje. Pole fkode obsahuje textový seznam numerických hodnot pro přiřazení k ID jednotlivým tlačítkům. Obě pole pak tvoří výchozí seznamy pro generování tlačítek. Stejný princip je použit i pro generování tlačítek pro numerickou klávesnici. Atributy cislo a fcislo jsou proměnné pro procházení jednotlivými poli. Posledním atributem třídy Main je comp. Je to instance třídy Comp. Přes tento atribut se přistupuje k metodě Zpracuj a k atributu Disp. Třída Main musí přistupovat přímo k atributu Disp proto, aby byl zobrazen na obrazovce kalkulátoru.

Třída Comp má na starosti veškerou obsluhu událostí. Obsluhuje zmáčknutá tlačítka a provádí výpočty. Atributy cislo1 a cislo2 jsou proměnné pro samotné numerické výpočty. Datový typ Number byl vybrán, jelikož zastupuje v JavaFX datový typ s plovoucí desetinnou čárkou. Tady je vhodné zmínit, že by šel použít datový typ Long, ale JavaFX má občas problémy s převody mezi datovými typy JavaFX a Java. Nicméně datový typ Number plně dostačuje na běžné výpočty, i co se týče mocnin a odmocnin. Tento kalkulátor navíc nemá být vědeckým kalkulátorem, takže i z tohoto důvodu datový typ Number plně dostačuje. Atribut pamet slouží k uchování čísla v paměti. Atributy prvnicislo a descarka jsou příznaky. Pokud je prvnicislo nastavené na hodnotu true, bude se kalkulátor chovat tak, že při dalším stisku numerického tlačítka se začne displej přepisovat. Atribut descarka značí, zda je už zadána desetinná čárka nebo ne. Pokud je nastaven na false, pak je možné přidat desetinnou čárku. Pokud je nastaven na true, už dále není možné znova připsat desetinnou čárku k číslu na displeji. Atribut operace uchovává hodnotu zvolené operace v případě, že je k jejímu provedení potřeba dva operandy. Jestliže jsou potřebné dva operandy, pak podle diagramu vnitřní logiky (obrázek 12) se požadovaná operace provede až po stisku tlačítka „rovná se“. Po stisku tohoto tlačítka se testuje hodnota tohoto atributu a podle jeho hodnoty se provádí příslušná operace. Atribut Cislo slouží k uchování numerické hodnoty ID tlačítka. Identifikaci stisknutého tlačítka bude věnován odstavec níže. S číselnou hodnotou se pracuje lépe než s hodnotou textovou, proto je zde tento atribut. Atribut cislic udržuje přehled o počtu číslic zadaných přes numerickou

klávesnici. Jak bude napsáno níže, displej kalkulátoru je realizován textovým řetězcem a tak by teoreticky bylo možné do něj zapsat více znaků, než můžeme zobrazit. Proto bylo nutné tímto atributem omezit počet zobrazovaných číslic. Funguje to tak, že při každém zmáčknutí numerického tlačítka se tento atribut zvětší o jedna a při dalším zmáčknutí numerického tlačítka se pak zkontroluje, zda už není překročen limit pro zobrazené číslice. Pokud je tento limit překročen, další číslice se již nezobrazí. Posledními dvěma atributy jsou operand a Disp. Disp je instance třídy Javafx.scene.text.Text. Tento atribut vlastně obstarává zobrazení displeje na kalkulátoru. K vytváření samotného textu/čísel displeje je použit atribut operand. Tento atribut je bindován do atributu text v Disp. Co to je data binding si rozebereme vzápětí. Třída Comp má ještě dvě metody. První z nich je Zpracuj, která obstarává veškerou vnitřní logiku. Vstupní parametr je typu MouseEvent, toto podrobněji rozeberu v části identifikace tlačítek. Druhou metodou je Nuluj a tato procedura má za úkol nastavit pokaždé descarku na false, prvnicislo na true a cislic na 0. Tato procedura je volána pokaždé, když se provádí tyto dvě operace zároveň. Z diagramu vnitřní logiky (obrázek 12) je možné vyčíst, že toto se děje opravdu často, proto vznikla i tato procedura.

4.3.1 Data binding

Tzv. bindování dat slouží v JavaFX k průběžné aktualizaci dat. Pokud chceme za chodu programu často měnit nějakou proměnnou a tyto změny sledovat, je potřeba použít data binding. Pro větší názornost, jak data binding funguje, si uvedeme příklad zdrojového kódu přímo z aplikace kalkulátoru, který tuto vlastnost využívá.

```
public var Disp:Text=Text{
x:10
y:100
stroke:Color.RED
content:bind operand;
font:Font{size:100 name:"Arial"}
fill:Color.WHITE;}
```

Tímto kouskem kódu je deklarován displej zodpovědný za zobrazování čísel na displeji. Povšimněte si klíčového slova bind operand v atributu content. Proměnná operand je textový řetězec, který obsahuje všechna zmáčknutá numerická tlačítka. V momentě, kdy zmáčkneme numerické tlačítko, chceme, aby se na displeji hned tato číslice zobrazila. Po zmáčknutí numerického tlačítka se tedy jeho hodnota, nebo spíše řečeno číslice, kterou představuje, vloží v podobě textového znaku do proměnné operand. Tím došlo k jeho změně. V tuto chvíli přichází na řadu binding, jelikož Disp má v atributu content řečeno, že má bindovat proměnnou operand. Proto tedy program sleduje proměnnou operand a dojde-li ke změně hodnoty této proměnné, tato změna se projeví i v atributu content a tím dojde zároveň i ke změně na displeji kalkulátoru. Toto je celý princip data bindingu v JavaFX. Drobná poznámka na konec. Data binding je užitečná vlastnost, ale příliš mnoho bindovaných proměnných v programu by mohlo zpomalovat chod aplikací na mobilním zařízení.

4.3.2 Princip fungování displeje

Jak jsem napsal v oddíle výše: Disp binduje operand a tím dochází k zobrazování číslic na displeji kalkulátoru. Nyní rozeberu podrobněji, jak vlastně funguje zobrazování číslic na displeji a následná práce s nimi. Jak je vidět na obrázku číslo 12, vyskytuje se zde aktualizace displeje. Podstatou aktualizace je přidání číslice do textového řetězce. Tato operace se provádí následovně:

```
operand=operand.concat(Cislo.toString());
```

Tento řádek má na svědomí zobrazení číslice odpovídající zmáčknutému numerickému tlačítku. Doslova se k textovému obsahu proměnné nakonec připojí na textovou hodnotu převedený obsah proměnné Cislo. Tím je tedy dáno, že obsah displeje je textový řetězec. Pro práci kalkulátoru ale potřebujeme tento řetězec převést do číselného typu Number. To provedeme tímto řádkem kódu:

```
cislo2=cislo2.valueOf(operand);
```

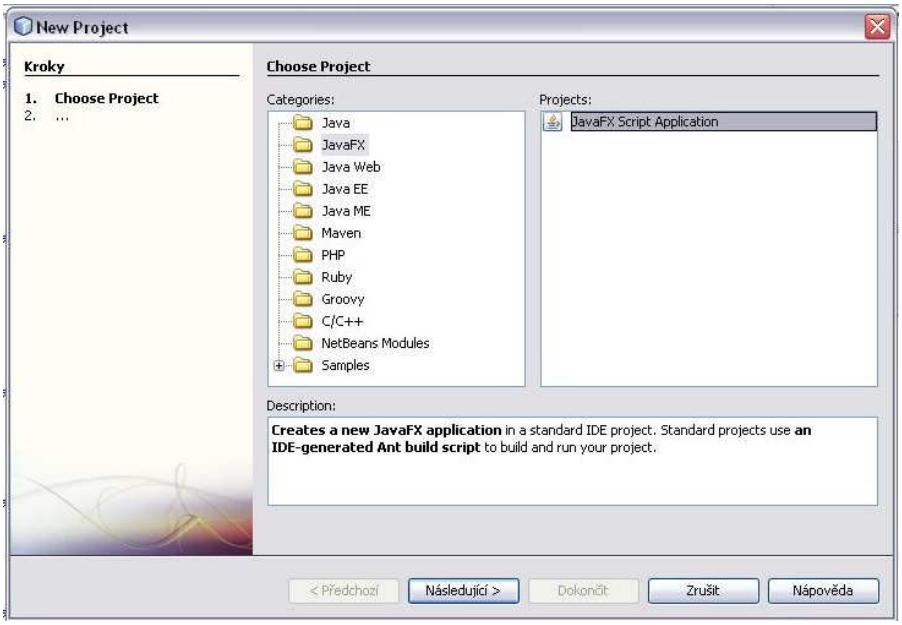
Touto operací se do proměnné cislo2 zapíše na číselnou hodnotu převedený textový řetězec obsažený v proměnné operand. Po výpočtu je samozřejmě nutné zobrazit výsledek, a proto musíme provést tuto konverzi:

```
operand=cislo1.toString();
```

Proměnná operand, jak bylo napsáno výše, je bindovaná, proto veškeré změny této proměnné se okamžitě zobrazují na displeji kalkulátoru.

4.3.3 Vývoj aplikace

Po analyzování chodu vnitřní logiky kalkulátoru a zjištění možností programovacího jazyka JavaFX skript přistoupíme k samotnému vývoji aplikace. Nejprve je potřeba ve vývojovém prostředí NetBeans založit nový projekt pro JavaFX.



Obrázek 14 – Založení nového projektu JavaFX

Na obrázku číslo 14 je vidět, že založení nového JavaFX projektu je jednoduché. Z nabídky dostupných jazyků vybereme JavaFX a zde jedinou možnost JavaFX Script Application.

Oproti založení projektu pro Java ME je založení projektu JavaFX jednodušší. Má pouze dva kroky. V tomto druhém máme možnost nastavit jméno projektu, místo, kam se nám projekt uloží (toto je nastaveno na výchozí umístění do složky NetBeansProjects, viz adresářová struktura), a jestli chceme vytvořit projekt prázdný nebo z již existujícího zdrojového kódu. Následují ještě dvě zaškrtávací políčka. Prvním z nich nastavíme nový projekt jako výchozí a druhým políčkem se nám vytvoří soubor Main, ve kterém je hlavní část programu. Těmito dvěma kroky máme založený nový projekt.

4.3.4 Struktura programu

Program v JavaFX má jednu Stage{}, což je hlavní kontejner pro každou aplikaci v JavaFX. Stage má několik parametrů. Uvedu zde dva důležité. Prvním z nich je title, což je titulek celé aplikace, který se zobrazuje v hlavním rámu okna. Druhým a tím nejdůležitějším je scene. Tento parametr osbahuje kontejner Scene, kterým se referuje hlavní okno aplikace. Kontejner Scene obsahuje veškeré grafické prvky aplikace.

V kontejneru Scene se nastavují hlavně rozměry zobrazovaného okna a jeho obsah. Scene má atribut content, což je výčet grafických prvků, které bude výsledná aplikace obsahovat. Základní struktura programu vypadá:

```
Stage{
      title: „Kalkulator“
      scene: Scene{
      height: 800
      width: 480
      content: [ ]
}}
```

V content jsou uvedeny všechny další kontejnery pro zobrazování grafického prostředí aplikace.

Aplikace kalkulátoru vyžaduje implementovat velké množství tlačítek. Toho se dá dosáhnout třeba tak, že postupně napíšeme kód pro jednotlivá tlačítka. Přitom nesmíme zapomenout na fakt, že okno aplikace se bude vykreslovat od shora dolů, tedy v tom pořadí, v jakém jsou jednotlivá tlačítka ve zdrojovém kódu zadeklarovaná. S tímto faktem musíme počítat, jelikož by nám některé později vykreslené grafické prvky mohly překrýt ty již dříve zobrazené.

Postupné vkládání tlačítek do programu je velice neefektivní vzhledem k tomu, že aplikace kalkulátoru bude těchto tlačítek potřebovat velké množství. Kód tlačítka přitom obsahuje tyto prvky:

```
      Button {
      translateX: //x souřadnice posunu tlačítka vzhledem k počátku
      obrazovky
```

```
translateY://y souřadnice posunu tlačítka vzhledem k počátku
obrazovky
text: „Tlacitko“ //textový popisek tlačítka
font: Font{size:20} //udává velikost použitého fontu
height:30 //výška v pixelech
width:100// šířka v pixelech
onMousePressed: function( e: MouseEvent ):Void {
 comp.Zpracuj(int Cislo); //obsluha události tlačítka.
}}
```

Jak je z výše uvedeného zdrojového kódu zřejmé, bylo by potřeba tento kód vkládat pro každé tlačítko zvlášť a u každého separátně nastavovat obsah atributu text tak, aby každé tlačítko mělo svůj správný popisek. Stejně tak bychom museli každému tlačítku přiřadit jiný int kód pro zpracování odezvy tlačítka. Už jen napsání kódu pro samotná numerická tlačítka by bylo velice zdlouhavé. A to i přes to, že NetBeans by nám pomocí palety prvků mohlo značnou část kódu tlačítek vygenerovat samo. Proto bylo nutné vymyslet způsob, jak tato tlačítka „generovat“ co nejjednodušeji a na co nejméně řádků.

Pro řešení tohoto problému proto v hlavním programu vznikla pole s textovým obsahem, který je pak přiřazován jednotlivým tlačítkům. Samotné generování tlačítek tak probíhá ve dvou vnořených cyklech, kdy první z nich generuje jednotlivé řádky a druhý, vnořený, pak tlačítka v onom řádku. Tento postup výborně fungoval pro popisky tlačítek, nikoliv už pro jejich přiřazené kódy, které se pak předávají třídě Comp pro zpracování odezvy. Zde se vyskytl problém v tom, že veškerá tlačítka, ač měla přiřazená správné popisky, reagovala všechna stejně, jelikož jejich číselná hodnota se nezaznamenávala, a proto ji všechna tlačítka měla stejnou.

4.3.5 Identifikace zmáčknutého tlačítka

Jak bylo uvedeno výše, bylo potřeba vyřešit problém, jak generovat tlačítka co nejúsporněji a přitom vyřešit problém, jak předávat správnou hodnotu pro zpracování. Jelikož třída Comp už pracovala s integerovou hodnotou zmáčknutého tlačítka, bylo by velmi pracné ji předělat, aby tlačítka odesílala textovou hodnotu svého kódu. Práce s textovou hodnotou je ale dost komplikovaná a pro samotná numerická tlačítka by bylo lepší, kdyby předávala kód v číselné podobě. Proto bylo potřeba vyřešit tento problém.

Řešením tohoto problému bylo přidání atributu ID každému tlačítku. Hodnota ID je textová, takže ji je možné správně přiřazovat v cyklech. Proč byla hodnota ID řešením? Tato hodnota se mimo jiné předává instanci třídy MouseEvent. MouseEvent je kontejner zachycující akce vyvolané uživatelem při pohybu myší. Její instance se proto vytvářejí v momentě kliknutí na grafický prvek. Jak je vidět ve výše zmíněném ukázkovém zdrojovém kódu pro tlačítko, má každé tlačítko obslužnou funkci onMousePressed, která právě vytváří a předává instanci třídy MouseEvent. Instance třídy MouseEvent mimo jiné obsahuje ID objektu, na kterém byla tato akce vyvolána. Tím je tedy možné zjistit, jaké tlačítko bylo zmáčknuto. A tím pádem je i vyřešen problém s generováním tlačítek v cyklech, jelikož hodnota ID je textová a proto je i správně přiřazena jednotlivým tlačítkům. Výsledný kód pro generování všech numerických tlačítek vypadá následovně:

```
for(radek in [0..3]){
for(sloupec in [0..3]){
cislo++;
Button{
translateX:10+(sloupec*120)
translateY:450+(radek*70)
text:nazev[cislo]
font:Font{size:50}
id:kode[cislo]
height:60
width:100
onMousePressed: function( e: MouseEvent ):Void {
comp.Zpracuj(e);
}}}}
```

Těmito pár řádky vygenerujeme celou numerickou klávesnici. Stejným principem pak vygenerujeme funkční tlačítka, jen jejich popisy a ID vezmeme z jiných polí.

4.3.6 Instalace Kalkulátoru do zařízení.

Instalace výsledné aplikace do zařízení je velice jednoduchá. Nejprve musíme v NetBeans „vybudovat“ spustitelný archív s příponou .jar. Tento krok provedeme následovně: Nastavíme konfiguraci pro mobilní zařízení a zmáčkneme tlačítko Clear and Build. Tím se nám spustí vyčištění adresářů projektu od dočasných souborů a do složky dist se uloží výsledný archív. V našem případě to bude Kalkulator.jar. Tento soubor následně přeneseme do zařízení. Možností na jeho přenesení máme několik. Nejjednodušší je ho nechat přenést pomocí BlueTooth. Následně stačí „spustit“ tento archív a prostředí JavaFX v tomto zařízení nainstalované se samo postará o spuštění instalace. Zobrazí se nám úvodní obrazovka s dotazem, zda chceme tuto aplikaci nainstalovat. Zde zvolíme volbu Yes. V dalším kroku budeme dotázáni na to, kam chceme aplikaci nainstalovat. Nejdříve musíme zvolit vnitřní paměť zařízení nebo paměťovou kartu. Za druhé musíme zvolit, do které složky chceme tuto aplikaci nainstalovat. Volbu potvrdíme stisknutím OK. Následně se aplikace nainstaluje a zobrazí se nám okno s dotazem na spuštění aplikace. Buď potvrdíme a aplikace se začne načítat, nebo odmítneme a vrátíme se tím do souborového průzkumníka. Tím máme celou instalaci dokončenou a aplikaci pak můžeme spustit z nabídky aplikací po spuštění JavaFX prostředí.

Na obrázku číslo 15 si pak můžeme prohlédnout výslednou aplikaci Kalkulátoru na zařízení HTC Touch HD a spuštěnou v prostředí NetBeans. Obrázek spuštěného kalkulátoru v prostředí NetBeans je nepatrně oříznutý z důvodu vyššího rozměru okna aplikace než má rozlišení monitor, na kterém byla aplikace zobrazena. (Rozlišení monitoru: 1366x768 pixelů proti 800x480 okna aplikace)

Obrázek 15 – Kalkulátor v JavaFX na HTC Touch HD a v prostředí NetBeans

5 Závěr

Aplikace Kalkulátor pro zařízení HTC Touch HD vyřešila problém s tím, že na tomto zařízení není žádný vhodný kalkulátor. Jak bylo předesláno výše, systémový kalkulátor je opravdu velice jednoduchý. Tento mnou naprogramovaný kalkulátor je vhodný pro běžné a složitější výpočty. Pro vědecké výpočty je ovšem lepší sáhnout po kalkulátoru přímo pro tyto výpočty určenému. Aplikace tohoto typu ale bývají často zpoplatněné.

Drobnou nevýhodou tohoto kalkulátoru je delší čas potřebný na spuštění samotné aplikace. Toto je dáno, při porovnání HW konfigurace PC a tohoto komunikátoru, přece jen menším výkonem a menší dostupnou pamětí v případě komunikátoru. Nicméně běh samotné aplikace je už svižný a kalkulátor se dá ovládat jak stylusem, tak prstem.

Literatura

GOYAL, Vikram. 2005. J2ME Tutorial, Part 2: User Interfaces with MIDP 2.0. *Java.net : The Source for Java Technology Collaboration.* [Online] 03. 05 2005. [Citace: 01. 04 2010.] <http://today.java.net/pub/a/today/2005/05/03/midletUI.html>.

Oracle. 2010. Connected Device Configuration (CDC), version 1.1.2. *Developer Resources for Java Technology.* [Online] 2010. [Citace: 04. 03 2010.] <http://java.sun.com/javame/reference/apis/jsr218/>.

—. 2010. Java Platform, Standard Edition. *The Source for Java Developers.* [Online] 2010. [Citace: 13. 04 2010.] <http://java.sun.com/javase/6/docs/technotes/guides/index.html>.

—. 2010. JavaFX API 1.1. *Developer Resources for Java Technology.* [Online] 2010. [Citace: 04. 03 2010.] <http://java.sun.com/javafx/1.1/docs/api/>.

—. 2010. JavaFX Tutorial. *Developer Resources for Java Technology.* [Online] 2010. [Citace: 09. 05 2010.] <http://java.sun.com/javafx/1/tutorials/ui/>.

Sun Microsystems. 2006. CLCD 1.0 Generated Documentation. *Developer Resources for Java Technology.* [Online] 2006. [Citace: 04. 03 2010.] <http://java.sun.com/javame/reference/apis/jsr030/>.

—. 2006. CLCD 1.1 Generated Documentation. *Developer Resources for Java Technology.* [Online] 2006. [Citace: 04. 03 2010.] <http://java.sun.com/javame/reference/apis/jsr139/>.

—. 2004. Overview (Java 2 Platform SE 5.0). *Java SE at a Glance.* [Online] 2004. [Citace: 06. 04 2010.] <http://java.sun.com/j2se/1.5.0/docs/api/>.

—. 2006. Overview (MID Profile). *Java ME Technology.* [Online] 20. 06 2006. [Citace: 01. 04 2010.] <http://java.sun.com/javame/reference/apis/jsr118/>.

YUAN, Michael a SHARP, Kevin. 2005. MIDP user interface. *JAVAWORLD.* [Online] 15. 05 2005. [Citace: 31. 03 2010.] <http://www.javaworld.com/javaworld/jw-05-2005/jw-0516-midp.html?>.

Příloha A – Kontejnery pro vývoj GUI v Java ME a JavaFX

Tabulka 4 – Kontejnery pro GUI JavaFX

Balíček	Kontejner	Popis
Javafx.scene.layout	HBox	Stará se o horizontální rozložení prvků, které do tohoto kontejneru vložíme.
	VBox	Kontejner zabezpečující vertikální rozložení prvků do něj vložených.
	Tile	Rozložení pomocí klasické tabulky do sloupců a řádků.
Javafx.scene.control	Button	Klasické tlačítko pro GUI.
	CheckBox	Zaškrtávací políčko.
	Hyperlink	Objekt umožňující vkládat interaktivní hypertextové odkazy.
	Label	Klasický štítek pro textový popis v okně aplikace bez možnosti editace uživatelem.
	ProgressBar	Stavový řádek, ukazující průběh dané operace.
	ProgressIndicator	Graficky jiný indikátor průběhu prováděné operace.
	RadioButton	Rádiové tlačítko.
	ScrollBar	Posuvník pro posouvání například dlouhého textu.
	Slider	Posuvník pro pozvolné nastavování hodnoty.
	TextBox	Textové pole pro vstup/výstup textu.
	ToggleButton	Přepínací tlačítko. Graficky stejné jako klasické tlačítko, ale používá dvoustavovou logiku: Buď je zmáčknuto nebo není
	ToggleGroup	Kontejner pro definování skupiny přepínacích tlačítek, které jsou navzájem spřažené (Jde vybrat jen jedno ze skupiny.).

Balíček	Kontejner	Popis
javafx.scene.shape	Arc	Kruhová výseč. Jde definovat střed, poloměr, startovní úhel a délka oblouku.
	ArcTo	Oblouk. Definuje se střed, poloměr, začátek a délku oblouku
	Circle	Kreslí kruh se zadanými parametry
	CubicCurve	Kreslí kubickou Beziérovu křivku.
	CubicCurveTo	Podobně jako ArcTo, kreslí kubickou Beziérovu křivku od bodu do bodu.
	Ellipse	Vykreslení elipsy.
	HLineTo	Horizontální přímka z počátečního bodu do koncového bodu
	Line	Přímka z bodu do bodu
	LineTo	Přímka do cílového bodu.
	Polygon	Mnohoúhelník, který je specifikován sadou souřadnicových koordinátů.
	Polyline	Opět pomocí souřadnicových koordinátů můžeme vykreslit lomenou čáru.
	QuadCurve	Kreslí kvadratickou Beziérovu křivku.
	QuadCurveTo	Kreslí kvadratickou Beziérovu křivku z bodu do bodu.
	Rectangle	Kreslení čtyřúhelníků.
	SVGPath	Cesta, jejíž souřadnice jsou zadány v podobě textového řetězce.
	VLineTo	Vykreslí vertikální přímku do zadaného bodu.

Tabulka 5 – Kontejnery pro GUI Java ME

Objekt	Typ	Popis
Alert	ALARM	Zobrazuje okno s informacemi o události, o které chtěl být uživatel zpraven.
	CONFIRMATION	Potvrzovací vyskakovací dialog.
	ERROR	Zobrazuje chybovou zprávu.
	INFO	Informativní okno, zejména pro předání nějaké důležité informace uživateli.
	WARNING	Zobrazuje varovací okno alertu.
List	EXCLUSIVE	List umožňuje vybrat jen jednu položku ze seznamu.
	MULTIPLE	Je možnost vybrat více položek z listu.
	IMPLICIT	List po vybrání položky okamžitě zpracovává výběr zavoláním CommandListeneru.
Form	ChoiceGroup	Definuje skupinu objektů pro možnosti volby mezi nimi. Počet možností volby je dáno typem prvků, kterými je skupina implementována (radio/check box).
	DateField	Pole pro zadávání času a data do formuláře.
	TextField	Textové pole pro vstup textu do formuláře.
	StringItem	Pole zobrazující textový řetězec, který nemůže být uživatelem přímo modifikován.
	ImageItem	Slouží pro zobrazování obrázků v rámci formuláře.
	Gauge	Stavový řádek ukazující hodnotu Integer mezi 0 a maximální hodnotou.
	Spacer	Průhledné, neinteraktivní pole sloužící pro oddělování jednotlivých formulářových prvků.
	CustomItem	Slouží pro vytváření podtříd pro vytvoření nových grafických a interaktivních prvků ve formuláři.