VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

VÝVOJ APLIKACÍ PRO MOBILNÍ ZAŘÍZENÍ PDA A TELEFONY

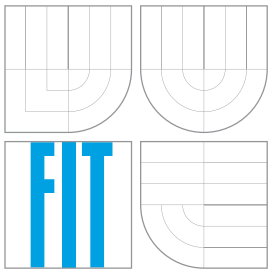
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MIROSLAV MURÍN

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

VÝVOJ APLIKACÍ PRO MOBILNÍ ZAŘÍZENÍ PDA A TELEFONY

THESIS TITLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MIROSLAV MURÍN
AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN HRUBÝ, Ph.D.
SUPERVISOR

BRNO 2007

Abstrakt

Táto bakalárska práca sa zaoberá vývojom aplikácií na platforme JAVA pre mobilné zariadenia PDA a telefóny. Práca obsahuje Midlet pre mobilné zariadenie napísaný v jazyku JAVA, konkrétne J2ME(Java 2 Platform, Micro Edition). Výsledkom práce je hra Piškvorky určená pre mobilné zariadenia s podporou platformy Java.

Klíčová slova

Java, J2ME, hra , telefon, mobil, PDA

Abstract

This bachelor thesis deals with a design of applications on the Java platform for PDAs and mobile phones. This work comprises of midlet for mobile devices written in java programming language, particulary J2ME. The result is a tic-tac-toe game designed for mobile devices with the java platform support.

Keywords

Java, J2ME, hra , cellphone, mobile, PDA

Citace

Miroslav Murín: Vývoj aplikací pro mobilní zařízení PDA
a telefony, bakalářská práce, Brno, FIT VUT v Brně, 2007

Vývoj aplikací pro mobilní zařízení PDA a telefony

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Hrubého, Ph.D.

.....
Miroslav Murín
10. května 2007

Poděkování

Rád by som poďakoval vedúcemu práce Ing. Martinovi Hrubému, Ph.D. za odbornú pomoc a za všeobecný dohľad pri vytváraní tejto bakalárska práca.

Obsah

Kapitola 1

Úvod

Mobily majú dlhú históriu, ktorá siaha k počiatkom 70. rokov 20. storočia. Pre nízke zavádzacie náklady a rýchle rozmiestnenie mobilných sietí sa rýchlo rozšírili po svete a predstihli pevné telefóny. Dnes už vlastní mobilný telefón takmer každý dospelý, dospievajúci ako aj deti. Mobil sa stal nepostrádateľnou súčasťou každého z nás. Naučili sme sa využívať technologické vymoženosti, ktoré nám mobil prináša. Medzi tieto vymoženosti patrí aj hranie hier, chatovanie po internete z mobilu a mohol by som menovať mnoho ďalších vecí. Jednou z nich, na ktorú som sa ja zameral je prostredie JAVA nachádzajúce sa v mobilnom zariadení. Nemusí to byť práve spomínaný mobilný telefón ale napríklad aj PDA zariadenie. V týchto zariadeniach sa nachádza programové prostredie nazývané Java 2 Micro Edition, alebo tiež J2ME. Je to najmenšia z platforiem Java. Táto obmedzená sada virtuálneho stroja a API umožňuje vytvárať a spúšťať programy určené pre spomínané zariadenia.

V tejto bakalárskej práci som sa zameral na oblasť zábavy a vypracoval som hru, ktorej tradícia siaha ešte pred vznik prvého počítača a aj napriek tomu je stále veľmi obľúbenou. Je to hra Piškvorky. Hra je určená pre zariadenia podporujúce J2ME a konfiguráciu CLDC 1.0 opísanú v kapitole 2.3 a profil MIDP 2.0(kapitola 2.7), ako aj MIDP 1.0.

Pri štúdii možných nástrojov používaných pri vytváraní aplikácie som prišiel na základný a podstatný poznatok a to, že J2ME je jediná platform, v ktorej sme schopní programovať aplikácie v jazyku Java. Samozrejme ak by sme chceli vyvíjať aplikácie pre mobilné zariadenia ako sú napríklad Pocket PC, SmartPhone založené na architektúre Windows Mobile je nám k dispozícii platforma .Net od spoločnosti Microsoft. V neposlednom rade tu máme ešte platformu Palm OS a Symbian určenú taktiež pre PDA zariadenia, nazývané tiež palmtopy. V posledných dvoch rokoch sa platforma Symbian a v malom množstve aj Windows Mobile začína presadzovať aj na trhu s mobilnými telefónmi.

Samotné vypracovávanie programu bolo založené na tom, aby výsledný program bol spustiteľný na rôznych typoch mobilných zariadení a menej som sa venoval algoritmom samotnej hry. Preto sa možno hra bude z pohľadu hrateľnosti zdať príliš jednoduchá.

Kapitola 2

Technológia J2ME

J2ME je verzia produktu Java spoločnosti Sun Microsystems určená pre trh s elektronickými zariadeniami, akými sú napríklad mobilné telefóny, pagery, osobné digitálne asistenty(PDA), prídavné zariadenia a ďalšie malé zariadenia. Vývoj J2ME je zaistený projektom nazývaným Java Community Process(JCP), http://jcp.org/en/home/index, ktorý umožňuje komukoľvek s pripojením do internetu, aby sa na tomto vývoji podieľal [1].

2.1 História a vývoj J2ME

Na začiatku 90.rokov spoločnosť Sun Microsystems vytvorila nový programovací jazyk nazývaný OAK ako súčasť projektu Green Project [8]. Tento jazyk mal byť určený pre aplikácie do výrobkov spotrebnej elektroniky. Prvým zariadením kde sa použil tento jazyk bol prenosný ovládač STAR7. Zariadenie obsahovalo dotykový LCD display, vstavanú bezdrôtovú sieť a infračervené rozhranie [8]. OAK bol vyvinutý na základe skúseností vývojového týmu s jazykom C++. Dopyt po tomto zariadení bol však minimálny a preto sa neskôr spoločnosť rozhodla premenovať jazyk OAK na jazyk JAVA. Na trh taktiež prišla s novým produktom a tým bol prenositeľný webový prehliadač HotJava. V roku 1998 Sun Microsystems uvoľnila produkt nazývaný PersonalJava, alebo tiež pJava. Bol určený pre mobilné zariadenia všeobecne. PersonalJava zahŕňala súčasti JDK1.1.8 z J2SE a obsahoval tiež nové súčasti určené špeciálne pre mobilné zariadenia. Samotná J2ME bola na trh uvedená v roku 1999.

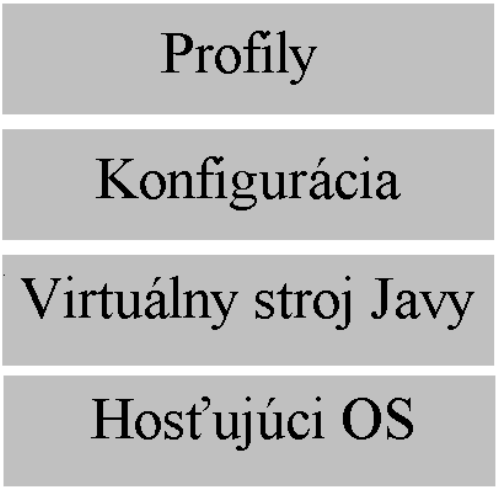
2.2 Štruktúra J2ME

Zo všeobecného pohľadu J2ME definuje nasledujúce komponenty:

- sériu javovských virtuálnych strojov, z nich každý sa uplatňuje u iného typu malého zariadenia a s odlišnými nárokmi
- skupinu knižníc a API, ktoré možno spustiť na každom virtuálnom stroji. Nazývajú sa Konfigurácia a Profily
- rôzne nástroje pre vývoj a nastavenie zariadenia

Obrázok 2.1 ukazuje prehľad vzťahov v prostredí J2ME. Centrom je javovský virtuálny stroj, ktorý pracuje na hostiteľskom operačnom systéme. Nad ním je špecifická konfigurácia

J2ME skladajúca sa z programovacích knižníc. Vrchol tvorí jeden, alebo viac J2ME profilov. Ide o doplnkové programovacie knižnice využívajúce príbuzné funkcie v podobných zariadeniach [1].



Obrázek 2.1: Prehľad architektúry pracovného prostrediaJ2ME

2.3 Konfigurácia CLDC

Mobilné zariadenia sa líšia svojou formou, funkciami a vlastnosťami. Majú však podobné procesory a podobné množstvá pamäti, myslené vnútornej pamäti. Nepatrí sem vonkajšia pamäť, ktorú môžu tvoriť rôzne dátové karty. Konfiguráciami sa preto definuje členenie zariadení založené na dostupnom množstve pamäti a výkone procesoru. V súčasnosti existujú dve štandardné konfigurácie. Sú to CLDC(Connected Limited Device Configuration) a CDC(Connected Device Configuration).

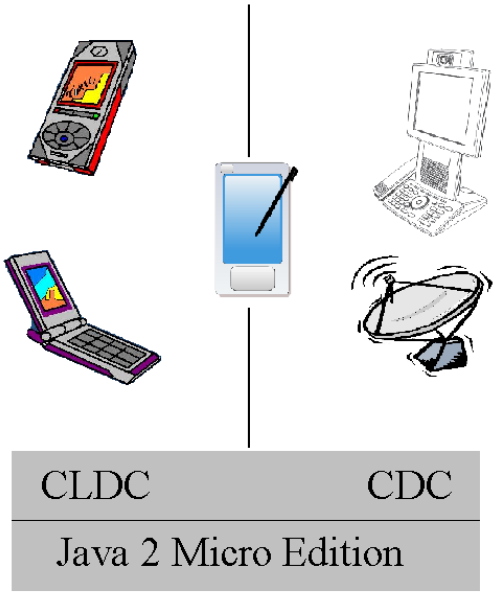
2.4 Konfigurácia CDC (Connected Device Configuration)

Je určená pre výkonné zariadenia, ktoré sú občas pripojené k sieti. Patria sem prídavné zariadenia, televízia po internete, domáce spotrebiče a navigačné systémy pre vozidlá a mnoho ďalších. CDC obsahuje virtuálny stroj podobný tomu, ktorý sa používa v J2SE. Rozdiel je v pamäti a zobrazovacích schopnostiach. Podľa oficiálnej špecifikácie [5] sú požiadavky na zdroje:

- zariadenie má 16, alebo 32-bitový procesor s minimálnou frekvenciou 25 MHz
- má 2MB a viac pamäti pre Javu. Táto čiastka zahrňuje pamäť RAM i pamäť flash, alebo ROM
- zariadenie vyžaduje plne funkčný virtuálny stroj Java 2 “Blue Book”
- zariadenie je pripojené k niektorému typu siete. Veľmi často je to bezdrôtové pripojenie s obmedzenou šírkou pásma

Konfigurácia CLDC(Connected Limited Device Configuration) je pre J2ME obvyklejšia. Udáva značne menšie nároky na zabudované zariadenie než CDC. Podľa špecifikácie [1] sú požiadavky na zdroje:

- zariadenie môže mať celkovo 160 až 512KB pamäti pre prostredie Java vrátane RAM a pamäti flash alebo ROM
- zariadenie môže mať obmedzený zdroj energie, napr. batériové napájanie
- zariadenie je pripojené k niektorému typu siete. Veľmi často je to obojsmerné bezdrôtové pripojenie s obmedzenou šírkou pásma



Obrázek 2.2: Architektúra J2ME, CLDC a CDC

Na obrázku 2.2 je vidieť, že aj napriek odlišným konfiguráciám nie je hranica medzi nimi ostrá. Predpokladá sa, že v budúcnosti sa vďaka technickému pokroku bude rozdiel naďalej zmenšovať. Treba mať však na pamäti, že rozdiel [1] [5] medzi CDC a CLDC je predovšetkým v minimálnom množstve pamäti zariadenia, v používaní, alebo nepoužívaní batérií a v prítomnosti, alebo neprítomnosti užívateľského rozhrania.

2.5 Virtuálne stroje pre CLDC a CDC

Ako som už spomenul tak CLDC a CDC stanovujú svoje vlastné skupiny podporovaných zariadení. Následkom toho sa aj Virtuálny stroj v každej tejto skupine musí líšiť. Preto má CLDC ako aj CDC svoj vlastný virtuálny stroj. Pre CLDC je to KVM(Kilo Virtual Machine) a pre CDC je to CVM.

2.5.1 KVM(Kilo Virtual Machnine)

KVM bol pôvodne vytvorený v laboratóriách spoločnosti Sun Microsystems ako výskumný projekt pod názvom “Spotless” [9]. Cieľom tohto projektu bolo vytvoriť virtuálny stroj

na základe požiadavky behu samotnej Javy na výkonných elektronických zariadeniach s obmedzenými zdrojmi.

Spoločnosť Sun zmenšila veľkosť vtedajšieho virtuálneho stroja používaného na klasických PC a taktiež zmenšila veľkosť všetkých tried. Zredukovala tak množstvo potrebnej pamäte, ktorú KVM využívala. KVM [4] samotná obsahuje v tejto dobe 40KB kódu. Je napísaná v jazyku C a vyžaduje iba 128KB dostupnej pamäte pre spustenie na 16, alebo 32-bitových procesoroch s minimálnou frekvenciou 25 MHz.

Minimálne požiadavky na zdroje podľa špecifikácie [4] sú:

- 128KB pamäti ROM , batériou zálohovanej pamäti, alebo pamäti typu flash pre trvalé uloženie VM Javy a knižníc tried, ktoré platformu CLDC tvoria
- 32KB energeticky závislej pamäti, ktorá je k dispozícii pre bežiacu aplikáciu a prípadné alokácie za behu. Táto pamäť sa používa k splneniu dynamických požiadaviek aplikácie

Virtuálny stroj KVM [1] má z dôvodu úspory pamäťového priestoru a zníženej záťaži procesora tieto hlavné obmedzenia:

- nepodporuje plávajúcu radovú čiarku
- nemožno použiť finalizáciu
- obmedzenie spracovania chýb
- JNI(Java Native Interface) nie je k dispozícii
- neumožňuje zavedenie zavádzača tried
- chýba skupina vlákien

2.5.2 CVM

Pôvodne bola CVM [1] skratka pre “Compact” Virtual Machine. Inžinieri v Sun Microsystems však zistili, že nevšímaví predavači (alebo nevzdelanci) by mohli zameniť výraz “compact” v skratke CVM s písmenom K v skratke KVM. A tak v súčasnosti C v slove CVM jednoducho nič neskracuje. CVM je vytvorené pre väčšie zariadenia používajúce CDC [9]. CVM má oproti KVM [4] viac možností:

- podporuje plávajúcu rádovú čiarku
- JNI
- možnosť definovať zavádzače tried
- serializácia objektov
- reflexi

2.6 Profily

J2ME umožňuje definovať javovské prostredia pre rôzne produkty rovnakej úrovne tým, že zavádza profily. Profil je vlastne sada programových rozhraní (API) tvoriacich nadstavbu konfigurácie(obr. 2.3).

MIDP(Mobile Information Device Profile)
Profil navrhnutý pre prácu s CLDC poskytujúci sadu programových rozhraní použiteľných v mobilných zariadeniach, ako sú mobilné telefóny a obojsmerné pagery [1] [9]. Tieto zariadenia musia spĺňať nasledujúce parametre:

- displej musí mať minimálne 96 x 54 pix a 2 farby
- mať klávesnicu, alebo dotykovú obrazovku
- 32KB pamäti pre prácu s Javou, 128KB stálej pamäte pre komponenty MIDP a 8KB stálej pamäte pre dlhodobé ukladanie dát
- možnosť obojsmerného sieťového spojenia

Tento profil je najznámejší profil J2ME. Malým aplikáciám, ktoré bežia pod MIDP sa hovorí Midlety bližšie opísané v kapitole 5.1.

PDA profil
Profil PDA je veľmi podobný profilu MIDP. Je určený pre organizéry PDA. Tie majú lepší a väčší display a viac dostupnej pamäte ako majú mobilné telefóny.

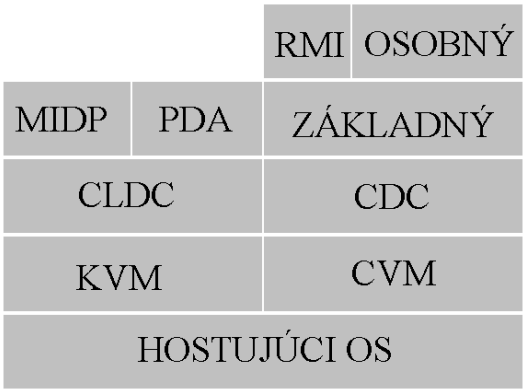
Základný profil
Základný profil rozširuje programové rozhranie, ktoré poskytuje CDC, ale nedodáva žiadne API pre užívateľské rozhranie. Už podľa názvu je možné zistiť, že tento profil má slúžiť ako základný profil pre ďalšie profily, napríklad pre Osobný profil a RMI profil.

Osobný profil
Osobný profil rozširuje možnosti Základného profilu o grafické užívateľské rozhranie(GUI), na ktorom možno spustiť applety pre Java Web.

RMI profil
Profil RMI pridáva k Základnému profilu knižnicu pre vzdialené volanie metód J2SE. Podporuje sa iba klientská strana tohto API.

2.7 MIDP 2.0

Špecifikácia MIDP 2.0 [2] je rozšírenou kópiou špecifikácie MIDP 1.0. Pri tvorbe svojej bakalárskej práce som použil práve vyššie spomínanú špecifikáciu MIDP 1.0. Dôvod prečo som sa rozhodol práve pre túto špecifikáciu bol ten, že na svete je ešte stále mnoho ľudí, ktorý vlastnia starší mobilný telefón , ktorý nepodporuje najnovšiu špecifikáciu. Naproti tomu, telefón, ktorý vlastním obsahuje veľa technických vymožeností a taktiež má "dostatok" pamäte pre prácu v prostredí J2ME. V programe som sa však šikovne vyhol programovacím technikám a rozšíreniam tak aby výsledný program bol plne funkčný aj na staršom mobilnom telefóne so špecifikáciou MIDP 1.0. Ako príklad môžem uviesť to, že v celom programe nepracujem s plávajúcou radovou čiarkou.



Obrázek 2.3: Profily v prostredí J2ME

Špecifikácia MIDP 2.0 bola uvedená v novembri roku 2002 a na jej tvorbe sa podieľali aj výrobcovia mobilných telefónov. Výsledkom je značné rozšírenie a odstránenie nedostatkov [9], ktoré staršia verzia obsahovala.

MIDP 2.0:

- nevyžaduje žiadnu konkrétnu verziu CLDC, ale predpokladá, že väčšina zariadení bude obsahovať verziu CLDC 1.1
- zahŕňa podporu plávajúcej rádovej čiarky
- pokiaľ ide o pamäť má o 128KB viac RAM a o 98KB viac pre beh JRE
- v oblasti bezpečnosti sa zachoval pôvodný bezpečnostný režim midletu bežiaci v sandboxe, ktorý sa tam nachádzal pre nedôveryhodné sady midletov

Okrem všetkých noviniek, ktorých je omnoho viac ako som tu spomenul, je treba spomenúť aj nedostatky, alebo vlastnosti, ktoré by sa tu hodili a práve v MIDP 1.0 sa nachádzali. Patrí sem napríklad prístup k telefónnemu zoznamu, posielanie SMS správ, zahájenie telefónneho hovoru, alebo prístup k dátam v telefóne ako k súborovému systému. Tieto funkcie sa dajú nájsť iba u telefónov niektorých výrobcov v ich vlastnom rozhraní, ktoré by časom malo byť nahradené rozhraním definovaným v rámci JSR(Java SpecificationRequest).

Kapitola 3

Vývojové a testovacie nástroje

V dnešnej dobe existuje mnoho vývojových nástrojov, v ktorých môžeme písať kód. Z hľadiska užívateľskej prívetivosti by som nástroje rozdelil na dve hlavné skupiny. Prvou skupinou sú editory typu PSPad, WordPad od spoločnosti Microsoft nachádzajúci sa v každej edícii Windows od verzie Windows 95. Ďalej sem patrí v poslednom období obľúbený OpenSource nástroj AbiWord a mnoho ďalších ako je uvedené na stránke WikiPedie [6]. Výhodou, ktorá hrá v prospech týchto editorov je, že na vás neútočia množstvom funkcií a rôznych nastavení takže stačí si ich jednoducho nainštalovať a začať v nich okamžite pracovať. V novších verziách niektorých editorov sa už objavuje aj možnosť automatického doplňania niektorých častí kódu. Často to však spôsobuje viac problémov ako osohu. V mojom prípade som testoval textový editor PSPad v.4.5.1. Druhou skupinou editorov, alebo by sa skôr patrilo nazývať ich vývojovými prostrediami(IDE) určenými k programovaniu aplikácií v konkrétnom, alebo rôznych jazykoch ako je tomu napríklad vo Visual Studiu 2005, alebo najnovšom VS 2007 od spoločnosti Microsoft. Tento produkt nie je voľne šíriteľný a preto ak ho chceme používať musíme si ho zaplatiť.

To ma privádza k ďalšiemu deleniu týchto nástrojov a to na voľne šíriteľné tzv. freeware, alebo platené. K najznámejším a najpoužívanejším patria Eclipse a vývojový nástroj NetBeans. Oba tieto nástroje sú určené k tvorbe aplikácií v jazyku Java. Flexibilný návrh oboch týchto nástrojov dovoľuje však rozšíriť zoznam podporovaných programovacích jazykov za pomocou pluginov, napríklad o C++ alebo PHP. Spoločným menovateľom týchto nástrojov je, že oba sú vytvorené v jazyku Java a za oboma stojí spoločnosť Sun Microsystems. Treba preto dodať, že obsahujú mnoho rovnakých funkcií a smer ich vývoja je veľmi podobný.

Hlavnou výhodou pri použití týchto nástrojov je ich natívná schopnosť čo najviac uľahčiť užívateľovi prácu pri tvorbe aplikácií, jednoduchým užívateľským rozhraním a intuitívnym ovládaním. Môžeme spomenúť funkcie, ktoré sa stávajú štandardom ako automatické doplňanie a formátovanie zdrojového kódu, ktoré je celkom inej úrovni ako je to pri spomínaných klasických textových editoroch. Odhaľovanie syntaktických chýb v zdrojovom kóde už počas písania je taktiež veľkou výhodou. V čom sa však najviac odlišujú od ostatných nástrojov je možnosť otestovanie funkčnosti aplikácie, vyhľadávanie chýb tzv. bugov.

Pokiaľ ide o testovacie nástroje až tak veľký výber sa nám neponúka. Tieto prostredia neumožňujú editáciu zdrojových kódov, ale je možné ich integrovať do niektorých vývojových prostredí ako je napríklad vyššie spomínaný Eclipse. Nástroje na testovanie obsahujú referenčné implementácie CLDC a MIDP a taktiež pomocné programy pomocou, ktorých je možné napríklad nahrať aplikáciu do mobilného zariadenia v prípade, že testovanie prebehlo správne. V opačnom prípade sa neodporúča nahrávať aplikáciu do zariade-

Eclipse	NetBeans
-je to platforma -lepšia podpora pluginov -štruktúrovanejšie vyhľadávanie v programovej hierarchii -lepšie refractorovanie -rozšíriteľná podpora pre kompiláciu a modelovanie programu -obsahuje grafický framework SWT -závislosť na OS	-čistokrvné IDE -platformová nezávislosť grafického rozhrania -chýbajúci AST model a "open compiler" -dva hlavné produkty, vývojové prostredie NetBeans IDE a NetBeans Platform

Tabulka 3.1: Eclipse vs NetBeans

nia. Jediným výsledkom by bola chybová správa systému. K fungovaniu takých nástrojov je potrebná inštalácia J2SDK, ktorú si môžete stiahnuť na stránkach spoločnosti Sun Microsystems http://java.sun.com.

3.1 J2MEWTK(J2ME wireless toolkit)

Asi najpoužívanejším testovacím prostredím je J2MEWTK(J2ME wireless toolkit), ktorý je možné si stiahnuť zo stránky http://java.sun.com/products/sjwtoolkit/index.html. Umožňuje vytvorenie nového alebo existujúceho projektu, vytváranie, spustenie a kompiláciu MIDLETov, ladenie aplikácií , vytvorenie balíka JAR a zmenu atribútov MIDLETu. Je tu možné spustiť aplikáciu v rôznych emulátoroch mobilných telefónov. Je to výhodné k otestovaniu prenositeľnosti danej aplikácie medzi rôznymi typmi telefónov. Minimálne požiadavky pre inštaláciu sú:

- 100 MB voľného priestoru na pevnom disku
- 128 MB pamäti RAM
- 800MHz Pentium 3 CPU

Ako nevyhnutný softvéru [7] sa uvádza Apple QuickTime player (k prehrávaniu AMR súborov v prostredí Windows). Ak chceme prehrávať AMR(Adaptive Multi-Rate) súbory v prostredí Linux musíme mať nainštalovanú podporu 3GPP.

3.2 Sony Ericsson SDK 2.2.4

Sony Ericsson SDK 2.2.4 je najnovšia verzia softvéru, ktorá obsahuje testovacie prostredie, emulátor(KToolbar) a nástroj pre nahrávanie aplikácie do mobilného zariadenia(Device Explorer). Tak ako J2MEWTK aj SE SDK 2.2.4 obsahuje množstvo emulátorov rôznych typov mobilov avšak v tomto prípade sa jedná len o zariadenia značky Sony Ericsson. Minimálne požiadavky pre inštaláciu sú:

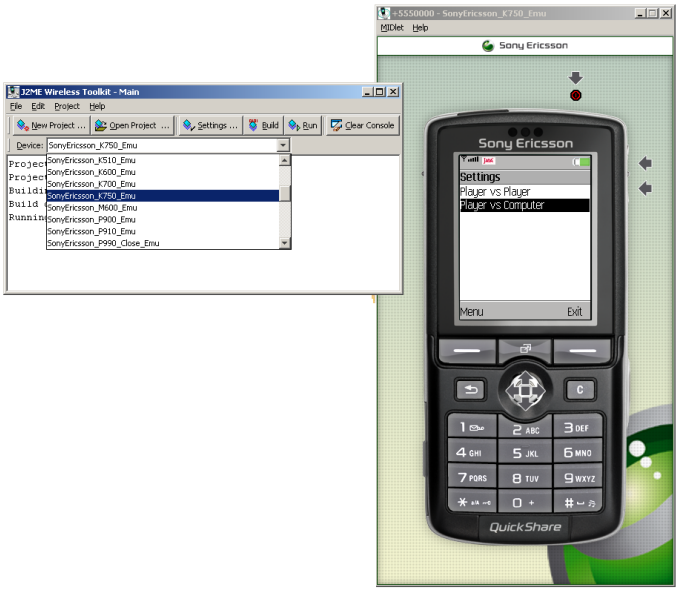
- 140 MB voľného priestoru na pevnom disku
- 256 MB pamäti RAM

- 500MHz CPU

Požiadavky na softwer:

- Microsoft Windows 2000/XP
- Java SE Development Kit (JDK) 1.4.1 alebo viac (doporučuje sa JDK 1.4.2)
- DirectX 8.1 alebo viac

Tento balík nástrojov J2ME plne podporuje Java MIDP 1.0, MIDP 2.0, Java 3D API vrátane Javadoc pre Sony Ericsson prídavné zariadenia. Dostupná je aj podpora priameho ladenia aplikácie na mobilnom zariadení.



Obrázek 3.1: Sony Ericsson SDK 2.2.4

Kapitola 4

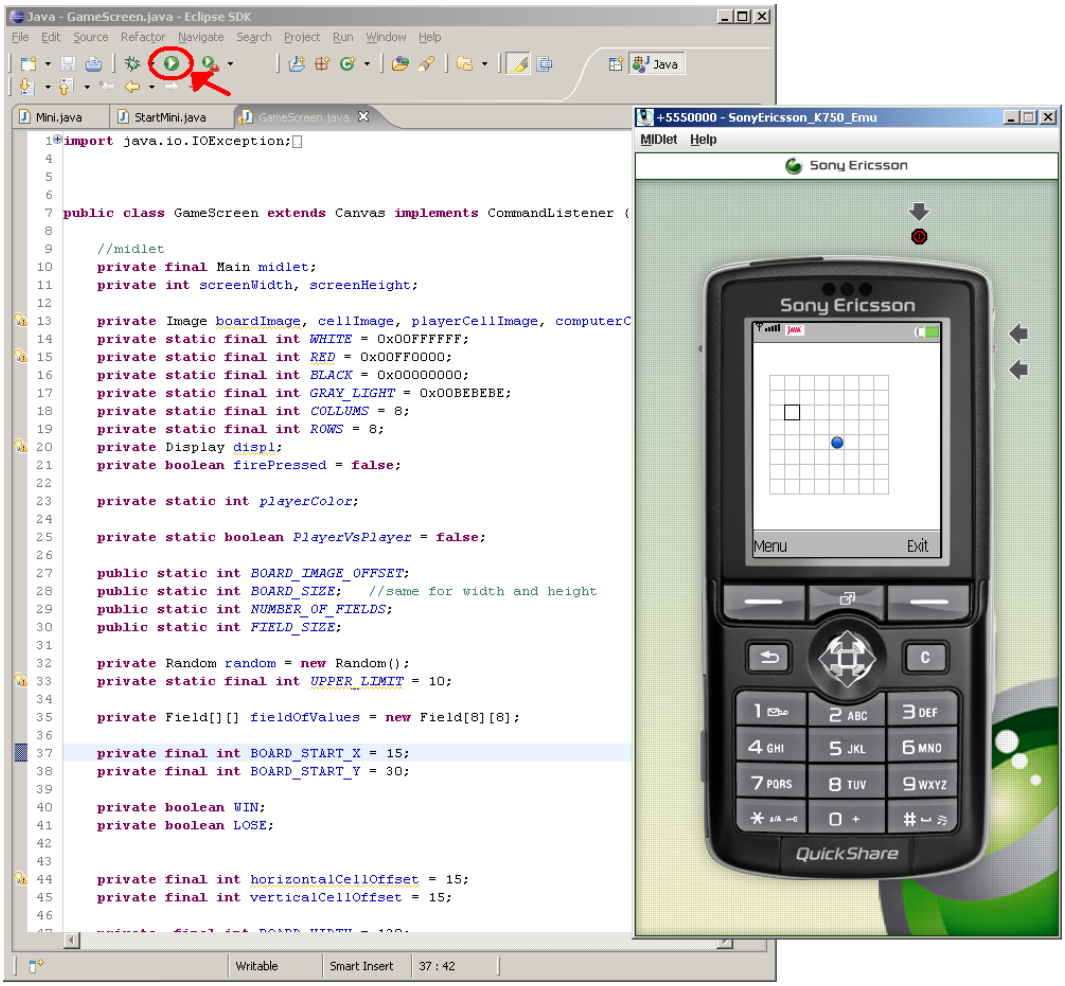
Práca s Eclipse 3.2

Ak budete svoj projekt vypracovávať v tomto vývojovom nástroji tu je návod ako na to. Je určite dobrou voľbou použiť nástroj ako je program Eclipse, pretože automaticky vytvorí mnoho vecí za užívateľa.

Vytváranie J2ME aplikácie
Toto prostredie za nás samo nastaví všetky potrebné hodnoty a premenné, ktoré je nutné nastaviť na začiatku vytvárania každého midletu. Pri vytváraní postupujeme podľa nasledujúcich krokov:

1. V menu File vyberieme New Project
2. Vybrali sme položku z kategórie J2ME a pokračujeme tlačidlom Next. Po od kliknutí sa nám zobrazí sa nové okno, v ktorom nás žiadajú aby sme udali názov nášho nového projektu
3. Ak sme zadali názov a následne ho potvrdili zobrazí ďalšie okno, ktoré nás informuje o tom, aký typ zariadenia má byť použitý ako predvolený pri vytváraní samotného programu. Je to z dôvodu, že nie všetky mobilné telefóny majú rovnaké parametre.
4. Ak všetko prebehlo ako má tak sme práve vytvorili nový projekt a môžeme vytvoriť hlavný súbor, alebo tiež Midlet. Vytvoriť Midlet môžeme dvoma spôsobmi. Ten prvý je, že sa tam preklikáme podobne ako pri vytváraní projektu v sekcii 1, alebo použijeme klávesovú skratku. Je to moja obľúbená možnosť, pretože nám Eclipse intuitívne ponúkne vytvorenie nového Midletu a nemusíme sa preklikávať všetkými kategóriami, ktoré ponúka. Klávesová skratka ALT + SHIFT + N nám teda ponúkne vytvorenie nového Midletu
5. Posledné na našej ceste k vytvoreniu nového Midletu je zadať názov Midletu. V dolnej časti okna, ktoré máme pred sebou sa nachádzajú tri zaškrtávacie políčka, ktoré za nás vytvoria potrebné metódy nevyhnutné pre správny beh programu.

Preklad a spustenie v prostredí Eclipse
Potom ako sme vytvorili novú aplikáciu a naprogramovali nadišiel čas na spustenie aplikácie. Stačí použiť jediné tlačidlo Run v panely nástrojov, ktoré za nás odvedie všetku prácu. Eclipse následne preloží a spustí aplikáciu v externom emulátore. Externý emulátor nie je súčasťou základného balíčku Eclipse 3.2. Je možné doinštalovať si ho ako plugin. Kde si ho stiahnuť a ako sa inštaluje nájdete na adrese http://eclipseme.org/.



Obrázek 4.1: Vývojové prostredie Eclipse 3.2

Kapitola 5

Triedy MIDP a ich využite v programe

Pretože je MIDP vrcholom CLDC, zameriava sa na nasledujúce v CLDC neriešené oblasti [1]:

Správa priebehu aplikácií
MIDP obsahuje balíček javax.microedition.midlet s triedami a metódami pre zahájenie, prerušenie a vymazanie aplikácií z hostiteľského prostredia.

Užívateľské rozhranie a udalosti
MIDP taktiež poskytuje balíčky javax.microedition.lcdui, ktoré zahrňujú triedy a rozhrania k tvorbe komponent grafického užívateľského rozhrania v aplikáciách.

Pripojiteľnosť k sieti
MIDP poskytuje rozhranie HttpConnection a súčasne inštalačnú sadu pre HTTP protokol čím rozširuje rozhranie ContentConnection.

Ukládanie dát v zariadení
Balíček javax.microedition.rms zavádza databázový záznamový systém, ktorý umožňuje ukladať dáta.

Balíček	Popis
javax.microedition.lcdui	Komponenty a udalosti grafického rozhrania
javax.microedition.midlet	Priebeh aplikácie
javax.microedition.rms	Ukladanie dát

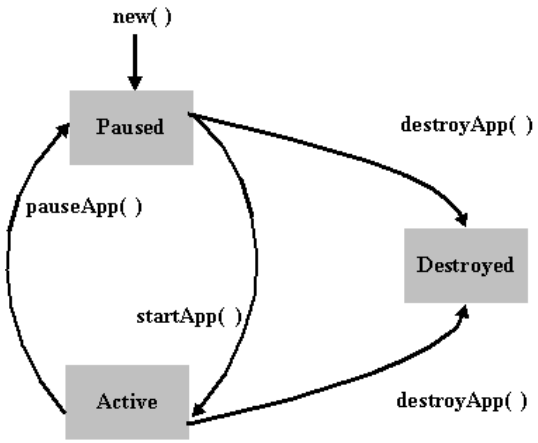
Tabulka 5.1: Balíčky v MIDP

5.1 Midlet

Vo všeobecnosti je Midlet program napísaný v jazyku Java určený pre vstavané elektronické zariadenia. Sú to hry, alebo aplikácie, ktoré majú byť spustiteľné hlavne na mobilnom

telefóne. Z programátorského hľadiska je Midlet trieda z balíka javax.microedition.midlet. Je to jediná trieda v tomto balíku a okrem nej sa tu nachádza aj výnimka MIDletStateChangeException.

Midlet sa môže nachádzať v troch stavoch tak ako to opisuje obrázok 5.1. Ak spustíme Midlet po prvýkrát nachádzať sa bude v stave prerušenia(*Paused*). V prípade, že je pripravený tak ho kontrolný program privedie do stavu aktívneho(*Active*). V tomto bode Midlet beží a užívateľ s ním pracuje. Celý proces môže byť prerussený buď užívateľom, alebo systémom MIDP. V stave zrušenia(*Destroy*) by mal Midlet uvoľniť systému MIDP všetky zdroje, ktoré v tú chvíľu používal [1].



Obrázek 5.1: Prechody medzi stavmi Midletu

5.1.1 Ako vytvoriť Midlet

Pred vlastnou tvorbou midletu je vhodné si celý projekt rozvrhnúť do niekoľkých vzájomne nadväzujúcich častí. Najľahším a asi aj najobecnejším rozdelením Midletu je:

1. Splash screen alebo úvodná obrazovka
2. Menu
3. Engine vlastnej aplikácie, alebo tiež hlavný motor aplikácie. Spracováva užívateľské dáta, vytvára dátové štruktúry, prevádza výpočty, ukladá dáta a podobne
4. Výstupné rozhranie aplikácie. Slúži k predaniu výsledkov behu aplikácie užívateľovi. Či už je to vizuálnou, akustickou, alebo kinetickou formou

Midlet ako program môžeme písať v rôznych editoroch, o ktorých ste si už určite prečítali v kapitole 3. Ja osobne som pracoval v prostredí Eclipse 3.2 a preto vám ponúknem návod ako na to v kapitole 4.

5.1.2 Programové využitie

V mojej aplikácii trieda *Main* je zdedená od abstraktnej triedy Midlet a teda zdedila jej príslušné tri metódy. Metóda

```
protected void startApp() throws MIDletStateChangeException
```

obsahuje kód programu, ktorý vytvorí užívateľské menu s troma položkami *Play*, *About* a *Exit*. Trieda implementuje rozhranie *CommandListener* a jeho metódu

```
public void commandAction(Command c,Displayable d)
```

V tejto metóde zisťujem, ktorá položka z menu bola vybraná . Následne sa vytvorí inštancia, objekt triedy *Settings*.

```
public void commandAction(Command c, Displayable d) {
    List l = (List)displ.getCurrent();
    switch (l.getSelectedIndex()) {
    case 0: new Settings(this); break;
```

5.2 LCDUI

Všetky triedy MIDP GUI sú umiestnené v balíčku javax.microedition.lcdui. Tento balíček obsahuje štyri rozhrania a dvadsaťtri tried. GUI sa v prípade MIDP skladá z nízkoúrovňového a vysokoúrovňového API [1].

API grafického rozhrania MIDP
Triedy *Graphics* a *Canvas* realizujú nízkoúrovňové API. Dané API poskytuje iba malé množstvo abstrakcie. Je navrhnuté pre aplikácie vyžadujúce presné rozmiestnenie a kontrolu grafických prvkov a prístup k nízkoúrovňovým vstupným udalostiam. Toto API dáva plnú kontrolu nad tým, čo sa má zobraziť na obrazovke zariadenia. Nemal by som tiež zabudnúť podotknúť, že Midlety využívajúce toto nízkoúrovňové API nezaručujú stopercentnú prenositeľnosť a to aj z dôvodu, že disponuje mechanizmami pre prístup k detailom, ktoré sú špecifické pre každé zariadenie. Naproti tomu vysokoúrovňové API je vytvorené pre aplikácie kde sa kladie väčší dôraz na prenositeľnosť aplikácie medzi mobilnými zariadeniami. Aby to všetko bolo zaručené využíva API vysokú úroveň abstrakcie. Užívateľ má tak veľmi málo priestoru pre kozmetické úpravy podľa vlastného gusta. Všetky triedy, ktoré realizujú vysokoúrovňové API sú potomkami triedy *Screen* z balíku javax.microedition.lcdui [1].

5.2.1 Programové využitie

V mojej aplikácii využívam tak nízkoúrovňové API ako aj vysokoúrovňové API. Trieda, v ktorej prebieha celé vykresľovanie hry sa nazýva *GameScreen*. Táto trieda dedí triedu *Canvas*, ktorá realizuje nízkoúrovňové API. Tá obsahuje metódu

```
protected void paint(Graphics g)
```

, ktorej parametrom je tzv. grafický objekt pomocou, ktorého sa vykresľuje. Celkovo využívam niekoľko metód na vykreslenie objektov nachádzajúcich sa na obrazovke. Patrí sem napríklad metóda:

```
public void drawGameScreen(Graphics g) {
    g.setColor(WHITE);
    g.fillRect(0, 0, screenWidth, screenHeight);
```

```
        drawBoard(g);
        if (firePressed == false)
            drawCursor(g);
        else
            firePressed = false;
        drawGame(g);
        if (WIN)
            drawGameOver(g, true);
        else if (LOSE)
            drawGameOver(g, false);
    }
```

Medzi ďalšie triedy, ktoré som použil pri vývoji aplikácie patria:

- Command - Trieda zapuzdruje informácie o význame určitej akcie. Zahrňuje však iba informáciu o príkaze, nie o skutočnej funkcii, ktorá sa zadaním príkazu prevedie.
- Display - Trieda, ktorá predstavuje manažér displeja a vstupných zariadení.
- Form - Obrazovka obsahujúca ľubovoľné položky(napr. obrázky, texty).
- Image - Trieda obsahujúca metódy pre prácu s obrázkami.
- List - Obrazovka obsahujúca zoznam možností.

Trieda Canvas obsahuje metódu

```
protected void keyPressed(int keyCode)
```

Táto metóda pracuje s udalosťami, ktoré vyvoláva stlačenie tlačítka na klávesnici zariadenia. Týmto spôsobom je ošetrený spôsob akým užívateľ komunikuje s aplikáciou. Metóde

```
public void doMoveCursor(int horizontalMove, int verticalMove)
```

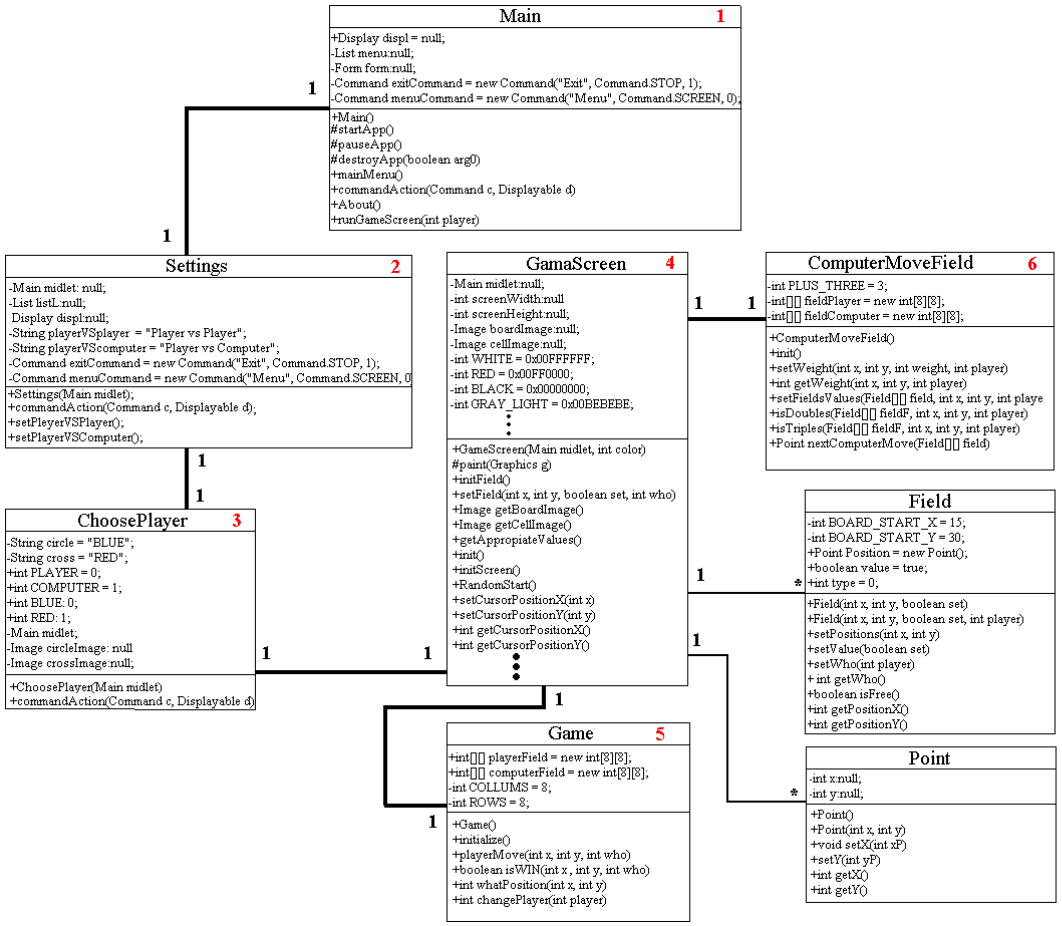
tak predávam hodnoty aktuálnej pozície kurzora.

5.2.2 Programové triedy Game a NextComputerMove

Myslenie samotnej aplikácie, alebo lepšie povedané umelá inteligencie aplikácie je implementovaná v týchto dvoch triedach. V tejto etape vývoja aplikácie som dával doraz na jednoduchosť použitých algoritmov. Počas implementácie som narazil na problém , pri ktorom by som sa rád pozastavil. Vychádzal som z algoritmu, ktorý použil istý autor hier pre mobilné telefóny, ktorý si hovorí Eric. Bohužiaľ táto prezývka, alebo meno je jediná informácia, ktorú sa mi podarilo získať o tomto autorovi. Snažil som sa využiť bitové operácie pri ukladaní a zisťovaní aktuálneho stavu hry. Keďže hracie pole má rozmery 8 x 8 obsahovalo teda 64 políčok. Tie mohli mať stav FREE, alebo USE. V prípade, že bolo políčko FREE tak sa na jeho pozícii v reťazci nachádzala 0 inak 1. Pomocou statických premenných, ktoré označovali výhru som využil binárnej operácie AND a znamienkového bitového posunu. Problém, na ktorý som narazil spočíval v prekrývaní stavov USED v prípade, že boli obsadené políčka v dvoch radoch za sebou. Tento problém som nedokázal riešiť v danom momente a pri použití daného algoritmu. Preto som sa rozhodol pre použitie polí. Základom

bolo dvojrozmerné pole typu Integer o rozmeroch 8 na 8. Každý index obsahoval údaj o tom či je dané políčko (ne)prázdne, kto je vlastníkom, či hráč, alebo počítač. Kontrola aktuálneho stavu prebieha po každom ťahu. Jej základom je kontrola susedných políčok. Ako pomocné triedy sem patria trieda *Point* a *Field* (diagram 5.2). Trieda Point uchováva pozície políčka a trieda Field absolutnú pozíciu políčka na obrazovke v pixloch a stav políčka. To znamená či je obsadené, alebo nie. Ak áno tak kým.

Celý algoritmus umelej inteligencii je založený na obrane. Zvolil som vhodnú ohodnocovaciu funkciu, ktorou som ohodnotil políčka susediace s už obsadenými políčkami. Hodnoty som sčítal a políčko s najväčšou hodnotou bolo následne obsadil.



Obrázek 5.2: Diagram tried

5.2.3 Problém rozlíšenia obrazoviek

Na trhu v dnešnej dobe existuje veľké množstvo mobilných zariadení s rôznymi parametrami. V takomto prípade sa musí aplikácia prispôsobiť danému zariadeniu. To v mojom prípade znamená, že veľkosť objektov vykresľovaných na obrazovku sa musí meniť v závislosti na type mobilného zariadenia, respektíve veľkosti rozlíšenie jeho obrazovky. Tu som využil metódy z triedy Displayable

```
int getHeight(), int getWidth()
```

, ktoré vrátia rozmery obrazovky daného zariadenia. Podľa toho sa v metóde

```
public Image getBoardImage()
```

vygeneruje konkrétny objekt, obrázok vyhovujúcich rozmerov.

Obrázek 5.3: Sony Ericsson D750i

Kapitola 6

Testovacie zariadenie

Zadaním mojej bakalárskej práce bolo navrhnúť a implementovať aplikáciu pre mobilné zariadenia PDA a telefóny. Ja som si vybral zariadenie typu mobilný telefón. Bolo to aj z dôvodu, že mobilné zariadenie PDA nevlastním. Vlastním však mobilný telefón Sony Ericsson D750i, ktorý môžete vidieť na obrázku 5.3. Toto zariadenie plne podporuje aplikácie na platforme Java a jeho bližšia špecifikácia je uvedená v kapitole 6.1.

6.1 Špecifikácia SE D750i

- Rozmery: 100x49x19,5 mm
- Celková hmotnosť: 99 g
- Farebný display: ANO
- Počet farieb: 262 144
- Rozlíšenie displeja: 176x220 px
- Pásma: 900/1800/1900 MHz
- Fotoaparát: 2,0 Megapixel (1632x1224 px)
- Vnútorná pamäť: 34MB
- Externá pamäť: Memory Stick Duo 64MB
- Java: ANO, verzia 2.0

6.2 Testovanie Midletu

Predchádzajúce kapitoly vysvetlili čo je to Midlet a ukázali základné odlišnosti v programovaní Midletu od aplikácií programovaných na báze platformy J2SE, alebo J2EE. Pri tvorení Midletu v prostredí Eclipse 3.2, opísaného v kapitole 4 a po nainštalovaní externého emulátora(kapitola 3.2) máme na výber otestovať Midlet identickými spôsobmi avšak s malou obmenou u oboch. Prvým sposob je spustenie aplikácie v prostredí Eclipse cez klasické menu Run → Run as → Java aplication ako môžeme vidieť na obrázku 4.1. Eclipse následne spustí externý emulátor a nahrá do neho všetky potrebné súbory pre správne fungovanie aplikácie. Tento spôsob mi prišiel ako najjednoduchší. Po niekoľkých spusteniach

	WTK1	WTK2
MIDP	MIDP 1.0	MIDP 2.0
CLDC	CLDC 1.04	CLDC 1.0 or 1.1
podpora	P800 series T610 series T630 series Z600 series	P900/P910 series S700/S710 series V800 F500i J300 series K300/K310 series K500/K510 series K600/K610 series K700/K750 series

Tabulka 6.1: Porovnanie WTK1 a WTK2

keď som testoval aplikáciu sa mi však emulátor nespustil. Dôvod tohto nečakaného zlyhania komunikácie programu Eclipse a externého emulátora som neodhalil. Po následnom reštarte programu Eclipse bežalo všetko ako má. Z tohto dôvodu som sa rozhodol, že nebudem spúšťa aplikáciu týmto spôsobom ale pustím si externý emulátor ako ďalšiu aplikáciu. Ja osobne som využíval Sony Ericsson SDK 2.2.4(kapitola 3.2) prostredie pre testovanie aplikácie. Nástroj obsahuje WTK1 API/emulation a WTK2 API/emulation. Rozdiel medzi oboma môžete vidieť v tabuľke 6.1.

Práca s SE SDK 2.2.4
Ak už vieme, ktorý typ mobilného telefónu chceme použiť zvolíme si WTK1 alebo WTK2. Po štandardnom nainštalovaní v prostredí Windows sa preklikáme ponukou
Štart → Programy → Sony Ericsson WTK1/WTK2 a spustíme program nazývaný KToolbar. Následne sa otvorí okno kde vytvoríme nový projekt New Project a zadáme názov projektu a názov Midlet Class Name. Ten sa musí zhodovať s názvom našej triedy, ktorá bola zdedená od triedy Midlet. V prípade, že sme úspešne vytvorili nový projekt je za potrebné skopírovať zdrojové súbory do zložky s názvom projektu. Tá sa nachádza po štandardnej inštalácii v

```
C:\SonyEricsson\JavaME_SDK_CLDC\PC_Emulation\WTK2\apps
```

Po otvorení zložky skopírujete do zložky *src* zdrojové súbory .java a do zložky *res* všetky externé súbory ako obrázky, hudba a iné. V spodnej časti menubaru sa nachádza choicebox s návestím Device, kde si zvolíme typ nášho mobilu. Následné stačí skompilovať súbory(tlačítko Build) a spustiť aplikáciu(Run).

6.3 Nasadenie Midletu

Existuje viacero spôsobov ako nahrať výslednú aplikáciu z počítača do mobilného zariadenia. Buď je to pomocou dátového kábla, prostredníctvom technológie bluetooth, alebo infraportu. Ak už máme k dispozícii ktorúkoľvek zo spomínaných foriem prenosu dôležité je aby mal nahrávaný balík správne parametre.

Tvorba Midletu v tomto prípade pozostáva z 5 fáz:

- napísanie Midletu

- skompilovanie zdrojového kódu
- preverifikovanie class súborov
- zabalenie aplikácie do JAR súboru
- vytvorenie JAD súboru a Manifestu

Nasledujúce riadky kódu ukazujú príkazy potrebné k vytvoreniu balíka JAR [3]. Príklady sú ukážkou ako na to v prípade, že celá aplikácia obsahuje iba jeden súbor a to Midlet.java. Ak by naša aplikácie obsahovala viac ako jeden súbor je potrebné uviesť všetky súbory. Najlepším riešením je namiesto názvu každého z nich zadať *.java.

Ku kompilácii zdrojového kódu sa používa príkaz javac

```
javac -bootclasspath c:\j2me\midp-fcs\classes Midlet.java
```

Týmto príkazom sme vytvorili súbor Midlet.class v aktuálnom adresári. Ďalším krokom je preverifikácia class súboru pomocou príkazu preverify:

```
preverify -classpath c:\j2me\midp-fcs\classes;. -d . MIDlet
```

V tomto kroku sa vytvorí preverifikovaná trieda Midlet.class a zapíše sa do podadresára *output*. Aplikáciu je nutné zabaliť do JAR súboru, aby sme umožnili dynamické zavádzanie MIDP aplikácie. K vytvoreniu JAR použijeme príkaz jar:

```
jar cvfm MIDlets.jar MANIFEST.MF MIDlet.class
```

Ak v našej aplikácii používame obrázky, hudbu alebo iné je potrebne ich uviesť na konci príkazu *jar* aby výsledný JAR súbor obsahoval dané súbory.

Posledným krokom je vytvorenie JAD súboru. JAD súbor musí obsahovať [1] [3] aspoň nasledujúce položky, pričom sa rozlišujú malé a veľké písmená: Midlet-Name, Midlet-Version, Midlet-Vendor, Midlet-JAR-Size, Midlet-JAR-URL. JAR súbor musí taktiež obsahovať *manifest*. Ten musí znova obsahovať určité položky: Midlet-Name, Midlet-Version, Midlet-Vendro. Maximálna dĺžka JAD a JAR súboru je 16 znakov. Veľkosť JAR súboru musí byť presne uvedená v JAD súbore.

6.3.1 Problém JAR a JAD

Pri vytváraní JAR a JAD súboru podľa [3] som mal problém správne skompilovať zdrojové súbory. Pre použitie príkazu *preverify* je potrebné nastaviť systémovú premennú *PATH* aby príkaz fungoval konzolovo z akéhokoľvek adresára, v ktorom sa práve nachádzame. V j2sdk1.4.2_14, ktoré som si nainštaloval a používal na kompiláciu sa daný preverifikátor *preverify.exe* nenachádzal. Po prehľadaní pevného disku som zistil, že daný súbor preverify.exe sa nachádza v adresári externého emulátora, ktorý ho využíva na preverifikovanie súborov pri spúšťaní aplikácie. V tomto spočíva spôsob akým som sa vyhol týmto nastaveniam a nadbytočnému písaniu príkazov do konzoly zakaždým keď som chcel nahrať aplikáciu do mobilu. Namiesto toho som nahral zdrojové kódy do externého emulátora, v ktorom som aplikáciu otestoval a ten za mňa vytvoril potrebné class, JAD a MANIFEST súbory. Potom stačí vytvoriť JAR súbor a zmeniť v súbore JAD veľkosť súboru JAR, ktorý sme vytvorili. Ak máme vytvorené všetky tieto súbory môžeme pristúpiť k poslednému kroku a tým je nahratie aplikácie do mobilného zariadenia. K tomuto slúžia programy ako je napríklad

Ftp Explorer Mobile 2007, alebo voľne dostupný My Phone Explorer 1.4.5. V poslednom období firma Sony Ericsson ponúka všetkým svojim zákazníkom softvér z vlastnej produkcie. Pri nasadení aplikácie do mobilu som osobne využil možnosti, ktoré tento softvér ponúka. Rozdiel medzi softvérom od firmy SE a inými je, že tento podporuje iba telefóny vlastnej značky. Naproti tomu My Phone Explorer 1.4.5 má dostatočne veľkú databázu mobilných zariadení, s ktorými spolupracuje. V prípade, že žiaden software použiť nechcete, alebo nemáte možnosť si ho zakúpiť existuje tu posledná možnosť. Je to priame nakopírovanie súborov do mobilného zariadenia.

Kapitola 7

Záver

Cieľom práce bolo preštudovať voľne dostupné nástroje pre vývoj aplikácií na báze Java a naimplementovať zvolenú aplikáciu. Z môjho pohľadu som splnil očakávané. To znamená aplikácia je plne funkčná a ja som sa plne oboznámil s vývojovým prostredím a prostriedkami. Ak by som sa mohol vrátiť späť v čase určite by som si dal väčší pozor pri návrhu aplikácie a vyhol sa určitým záludnostiam, lepším prepracovanejším návrhom určitých častí aplikácie. Samotné štádium implementácie prebehlo bez väčších problémov. Problémy menšieho kalibru nastali pri nasadení a testovaní samotného Midletu. Najčastejším problémom bolo zlé vytvorenie súborov JAR a JAD, ktorého riešenie som popísal v kapitole 6.3.1. V takom prípade sa nepodarilo aplikáciu vôbec spustiť a bolo zložité na chybu naraziť pretože som nemal k dispozícii žiaden výpis chybových správ, ako je to napríklad pri spúšťaní aplikáciu na počítači.

V prípade rozšírení a ďalšej práce na aplikácií v budúcnosti určite plánujem doplniť kvalitnejšiu umelú inteligenciu v podobe algoritmu *MiniMax*. Aplikácia momentálne podporuje dva rozdielne hracie módy. Mód Player vs Player by sa určite mohol hrať na dvoch rozdielnych mobilných zariadeniach a komunikácia by prebiehala pomocou technológie bluetooth.

Literatura

[1] Qusay H. Mahmoud. *Naučte se Java 2 Microedition*. Grada Publishing, a.s., 2002. ISBN 80-247-0444-7.

[2] WWW stránky. Jsr-000118 mobile information device profile 2.0. `http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html`.

[3] WWW stránky. J2ME and MIDP development, article midlet packaging with j2me. `http://www.onjava.com/pub/a/onjava/2001/04/26/midlet.html?page=1`.

[4] WWW stránky. J2ME building block for mobile devices. `http://java.sun.com/products/cldc/wp/KVM.pdf`.

[5] WWW stránky. J2ME for home appliances and consumer electronic devices. `http://developers.sun.com/techtopics/mobility/configurations/articles/cdc/`.

[6] WWW stránky. List of word processors. `http://en.wikipedia.org/wiki/List_of_word_processors`.

[7] WWW stránky. Sun java wireless toolkit for cldc. `http://java.sun.com/products/sjwtoolkit/download-2_5_1.html`.

[8] WWW stránky. What is J2ME. `http://www.codepadia.com/1/J2ME`.

[9] Kim Topley. *J2ME v kostce*. Grada Publishing, a.s., 2004. ISBN 80-247-0426-9.