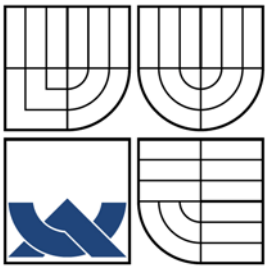
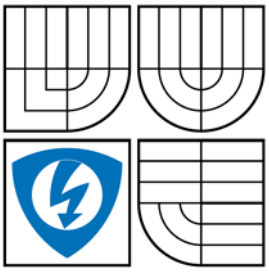


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

PŘÍJEM A POSÍLÁNÍ SMS ZPRÁV POMOCÍ APLIKACE URČENÉ PRO PLATFORMU JAVA ME

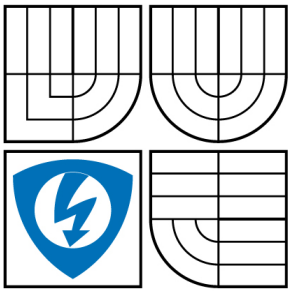
SMS APPLICATION FOR JAVA ME PLATFORM

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. LUKÁŠ RŮČKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR KOVÁŘ
SUPERVISOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Růčka Lukáš Bc. ***ID:*** 47476
Ročník: 2 ***Akademický rok:*** 2007/2008

NÁZEV TÉMATU:

Příjem a posílání SMS zpráv pomocí aplikace určené pro platformu JavaME

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte možnosti platformy JavaME pro příjem a odesílání SMS zpráv. Na základě teoretických poznatků navrhněte a vytvořte aplikaci na této platformě, která bude využitelná pro rozesílání i příjem SMS zpráv. Realizujte spojení aplikace s ovládacím rozhraním na PC, vytvořeném na platformě JavaSE.

DOPORUČENÁ LITERATURA:

[1] HAROLD, E.R.: Java Network Programming, O'Reilly, 2004, 760s., ISBN 0-596-00721-3
[2] MAHMOUD, Q. H., Naučte se Java2 Micro Edition. Grada Publishing a.s., 2002. ISBN 80-247-0444-7

Termín zadání: 11.2.2008 ***Termín odevzdání:*** 28.5.2008

Vedoucí práce: Ing. Petr Kovář

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

UPOZORNĚNÍ:

Vysoké učení technické v Brně — Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

POPISNÝ SOUBOR ZÁVĚREČNÉ PRÁCE

Autor: Bc. Lukáš Růčka

Název závěrečné práce: Příjem a posílání SMS zpráv pomocí aplikace určené pro platformu JavaME

Název závěrečné práce ENG: SMS Application for JavaME Platform

Anotace závěrečné práce: Úkolem diplomové práce je prostudovat možnosti platformy Java ME pro příjem a odesílání SMS zpráv a následné využití poznatků k realizaci aplikace na této platformě. V práci je po úvodním seznámení s jazykem Java a jeho vlastnostmi rozebrána problematika týkající se specifických vlastností platformy Java ME a vlastností zařízení podporujících tuto platformu. Následně je na základě teoretických poznatků proveden návrh aplikace vytvořené na platformě Java ME, která umí přijímat, zpracovávat, uchovávat a odesílat SMS zprávy, a která je schopna komunikace se serverovou aplikací vytvořené na platformě Java SE. Vytvořená Java ME aplikace je zamýšlena jako automatizovaný hlasovací SMS server, který lze provozovat na mobilním zařízení s podporou Java ME, a který je možno po spojení pomocí Internetu ovládat z aplikace vytvořené na platformě Java SE. V závěru práce jsou uvedeny poznatky a výsledky z testování vytvořených aplikací na skutečných mobilních zařízeních.

Anotace závěrečné práce ENG: The aim of this diploma thesis is to study the potential of Java ME platform for receiving and sending of SMS and to utilize obtained information subsequently in implementation of an application for this platform. The first part of the diploma thesis aims at explaining general terms of the Java language and its basic features. The next part deals with specific properties of the Java ME platform and properties of devices that support this platform. A design of the Java ME application based on theoretical knowledge is created then. This application has to be able to receive, process, store and send SMS and communicate with

Vysoké učení technické v Brně | Fakulta elektrotechniky a komunikačních technologií
Ústav telekomunikací

POPISNÝ SOUBOR ZÁVĚREČNÉ PRÁCE

the server application based on the Java SE platform. The created Java ME application serves as an automated SMS voting server, which can be used in mobile devices that support the Java ME platform and can be remote-controlled via Internet from the Java SE platform application. The final chapter presents the results and conclusions of testing of the created application on real mobile devices.

Klíčová slova: SMS, služba krátkých textových zpráv, Java ME, Java SE, JSR 120, JSR 205, WMA, volitelný balíček, MIDlet, MIDP 2.0, CLDC 1.1 klient, server

Klíčová slova ENG: SMS, short message service, Java ME, Java SE, JSR 120, JSR 205, WMA, optional package, MIDlet, MIDP 2.0, CLDC 1.1, client, server

Typ závěrečné práce: diplomová práce

Datový formát elektronické verze: pdf

Jazyk závěrečné práce: čeština

Přidělovaný titul: Ing.

Vedoucí závěrečné práce: Ing. Petr Kovář

Škola: Vysoké učení technické v Brně

Fakulta: Fakulta elektrotechniky a komunikačních technologií

Ústav / ateliér: Ústav telekomunikací

Studijní program: Elektrotechnika, elektronika, komunikační a řídicí technika

Studijní obor: Telekomunikační a informační technika

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Bc. Lukáš Růčka

Bytem: Dolní Bečva 178

Narozen/a (datum a místo): 13.4.1983, Valašské Meziříčí

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

se sídlem Údolní 244/53, 602 00, Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

prof. Ing. Kamil Vrba, CSc.

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - □ diplomová práce
   - □ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP: Příjem a posílání SMS zpráv pomocí aplikace určené pro platformu JavaME

Vedoucí/ školitel VŠKP: Ing. Petr Kovář

Ústav: Ústav telekomunikací

Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v*:

- □ tištěné formě – počet exemplářů ……..2
- □ elektronické formě – počet exemplářů ……..1

* hodící se zaškrtněte

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

Článek 2
Udělení licenčního oprávnění

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

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

Článek 3
Závěrečná ustanovení

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

V Brně dne: …….

…….. ……
Nabyvatel Autor

Anotace

Úkolem diplomové práce je prostudovat možnosti platformy Java ME pro příjem a odesílání SMS zpráv a následné využití poznatků k realizaci aplikace na této platformě. V práci je po úvodním seznámení s jazykem Java a jeho vlastnostmi rozebrána problematika týkající se specifických vlastností platformy Java ME a vlastností zařízení podporujících tuto platformu. Následně je na základě teoretických poznatků proveden návrh aplikace vytvořené na platformě Java ME, která umí přijímat, zpracovávat, uchovávat a odesílat SMS zprávy, a která je schopna komunikace se serverovou aplikací vytvořené na platformě Java SE. Vytvořená Java ME aplikace je zamýšlena jako automatizovaný hlasovací SMS server, který lze provozovat na mobilním zařízení s podporou Java ME, a který je možno po spojení pomocí Internetu ovládat z aplikace vytvořené na platformě Java SE. V závěru práce jsou uvedeny poznatky a výsledky z testování vytvořených aplikací na skutečných mobilních zařízeních.

Abstract

The aim of this diploma thesis is to study the potential of Java ME platform for receiving and sending of SMS and to utilize obtained information subsequently in implementation of an application for this platform. The first part of the diploma thesis aims at explaining general terms of the Java language and its basic features. The next part deals with specific properties of the Java ME platform and properties of devices that support this platform. A design of the Java ME application based on theoretical knowledge is created then. This application has to be able to receive, process, store and send SMS and communicate with the server application based on the Java SE platform. The created Java ME application serves as an automated SMS voting server, which can be used in mobile devices that support the Java ME platform and can be remote-controlled via Internet from the Java SE platform application. The final chapter presents the results and conclusions of testing of the created application on real mobile devices.

Klíčová slova

SMS, služba krátkých textových zpráv, Java ME, Java SE, JSR 120, JSR 205, WMA, volitelný balíček, MIDlet, MIDP 2.0, CLDC 1.1 klient, server

Keywords

SMS, short message service, Java ME, Java SE, JSR 120, JSR 205, WMA, optional package, MIDlet, MIDP 2.0, CLDC 1.1, client, server

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma Příjem a posílání SMS zpráv pomocí aplikace určené pro platformu Java ME jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 22.5.2008 Bc. Lukáš Růčka

Na tomto místě bych chtěl poděkovat všem lidem, kteří poskytli svůj mobilní telefon pro testování vytvořených MIDletů.

Dále bych chtěl poděkovat vedoucímu mé diplomové práce Ing. Petr Kovářovi, za užitečnou metodickou pomoc, připomínky a cenné rady při zpracování diplomové práce.

Bc. Lukáš Růčka

RŮČKA, L. *Příjem a posílání SMS zpráv pomocí aplikace určené pro platformu Java ME.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 66 s. Vedoucí diplomové práce Ing. Petr Kovář.

Použité symboly a zkratky

AMS	–	Application Management Software
API	–	Application Programming Interface
CBS	–	Cell Broadcast Service
CDC	–	Connected Device Configuration
CLDC	–	Connected Limited Device Configuration
CVM	–	Compact Virtual Machine
IDE	–	Integrated Development Environment
GCF	–	Generic Connection Framework
GSM	–	Global System for Mobile Communications
GUI	–	Graphical User Interface
JAD	–	Java Application Deskriptor
JAR	–	Java Archive
Java ME	–	Java Micro Edition (dřívější označení J2ME)
Java SE	–	Java Standard Edition (dřívější označení J2SE)
Jave EE	–	Java Enterprise Edition (dřívější označení J2EE)
JCP	–	Java Community Process
JDK	–	Java Development Kit (vývojové prostředí)
JIT	–	Just In Time compilation (compiler právě včas)
JNI	–	Java Native Interface
JP	–	Java Platform
JVM	–	Java Virtual Machine (abstraktní počítač neboli javovský virtuální stroj)
JVM	–	KVM Debug Wire Protocol
KVM	–	Kilo Virtual Machine
MAPI	–	Media API
MID	–	Mobile Information Device
MIDlet	–	Java ME aplikace pro MIDP
MIDP	–	Mobile Information Device Profile
MMAPI	–	Mobile Media API
MMS	–	Multimedia Messaging Service
O-T-A	–	Over The Air
PDA	–	Personal Digital Asistent (osobní digitální asistent)
PIM	–	Personal Information Manager
RI	–	Referenční Implementace
RMI	–	Remote Method Invocation
RMS	–	Record Management System
sandbox	–	bezpečnostní model "pískoviště" (sandbox) platformy Java
SDK	–	Software Development Kit
SMS	–	Short Message Service
TCK	–	Test Compatibility Toolkit
UDP	–	User Datagram Protocol
URI	–	Uniform Resource Identifier
WMA	–	Wireless Messaging API
WWW	–	World Wide Web

OBSAH

Seznam obrázků

1. Úvod

Úkolem diplomové práce je prostudovat možnosti příjmu a odesílání SMS zpráv na platformě Java ME. Poté na základě teoretických poznatků navrhnout a vytvořit aplikaci na této platformě, která bude využitelná pro rozesílání i příjem SMS zpráv. Následně realizovat spojení aplikace s ovládacím rozhraním na PC, vytvořeném na platformě Java SE.

Práce je rozdělena do 4 tématických částí. V první tématické části je po úvodu a teoretickém seznámení s jazykem Java a podrobněji rozebrána architektura platformy Java Micro Edition (Java ME) s ohledem na zařízení s omezenými zdroji (CLDC). Především jsou rozebrány pojmy javovský virtuální stroj (JVM), konfigurace, profil, MIDlet, volitelný balíček. Pojmy MIDlet a volitelný balíček jsou rozebrány obšírněji.

Ve druhé tématické části je proveden návrh řešení pro vlastní zpracování, na základě teoretických poznatků a zjištění současné situace na trhu. Toto řešení využívá platformy Java ME a volitelného balíčku Wireless Messaging API pro odesílání a přijímání SMS zpráv. Návrh obsahuje také možnost propojení aplikací z platformy Java ME s ovládacím rozhraním, které je vytvořeno na platformě Java SE.

Ve třetí tématické části je popsán postup realizace navržených řešení. Je zde uvedena vlastní realizace jednotlivých aplikací, popis jejich ovládání, praktické testy těchto aplikací a dosažené výsledky.

Poslední tématická část je závěr práce, ve kterém je shrnutí výsledů, nástin možného řešení některých problémů a nástin možného rozvinutí práce dalším směrem.

1.1. SMS a platforma Java ME obecně

Mobilní telefon a SMS (Short Message Service) jsou již neodmyslitelnou součástí života moderního člověka. Velmi velké procento (přes 50%) celé populace ČR vlastní mobilní telefon. SMS zprávy jsou velice oblíbenou formou komunikace, hlavně mezi mladými uživateli. V podstatě každý dnes na trhu dostupný mobilní telefon podporuje práci s SMS zprávami. Zásadní rozdíly mezi jednotlivými telefony jsou však v možnostech práce s nimi. Mnohé telefony pro svou jednoduchost nepodporují pokročilé funkce práce s SMS zprávami jako jsou koncepty, šablony, archivace SMS zpráv, skupiny pro komunikaci pomocí SMS, hromadné odesílání SMS a mnoho dalších. Tento stav však lze změnit s pomocí aplikace, která by byla vytvořena v platformě Java ME a následně do mobilního telefonu doinstalována.

Java ME platforma umožňuje do mobilního zařízení doplnit o další funkce v podobě aplikace naprogramované v jazyce Java. Dá se obecně říci, že všechny dnešní mobilní telefony střední třídy a výše obsahují podporu platformy Java ME (byť je tato podpora u jednotlivých mobilních telefonů rozdílná).

1.2. Problematika aplikací pro mobilní zařízení

V prvopočátcích rozvoje elektronických zařízení (mobilních telefonů včetně) bylo prakticky nemožné rozšířit tyto zařízení o jiné funkce, než které se rozhodl výrobce v daném zařízení implementovat. To platilo až do roku 2000, kdy společnost Sun Microsystems, Inc. uvedla na trh platformu Java ME, která byla určena pro elektronické zařízení (jako mobilní telefony, PDA, komunikátory, set-top boxy apod.).

Od svého uvedení na trh se platforma Java ME neustále vyvíjí, zdokonaluje a přizpůsobuje požadavkům současné doby. Především díky cenám jsou dnes prakticky pro všechny zcela dostupná malá, ale výkonná elektronická zařízení. Dnes asi nejrozšířenějším typem těchto zařízení jsou mobilní telefony. Dnešní mobilní telefony disponují relativně výkonnými procesory, řádově megabajty až desítky megabajty paměti, plně grafickými displeji, možností přístupu na Internet a mnoha dalšími rozšířeními (jako jsou Bluetooth, Wi-Fi, dotykový displej atd.). Tento stav umožňuje vyvíjet nové aplikace, které plně interagují s uživatelem a doplňují mobilní zařízení dalšími funkcemi. Mezi dnes nejrozšířenější aplikace patří především mobilní hry a jednoduché aplikace.

2. Seznámení s jazykem Java

2.1. Jak vznikla Java

Programovací jazyk *Java* je objektově orientovaný programovací jazyk vycházející z jazyka C++, který vyvinula firma Sun Microsystems, Inc. představila 23. května 1995. Java je určena především pro tvorbu aplikací, kde je podmínkou nezávislost na architektuře (přenositelnost) a spolehlivost řešení. Díky své přenositelnosti je Java používána pro programy, které mají pracovat na různých systémech počínaje čipovými kartami (platforma Java Card), přes mobilní telefony a různá zabudovaná zařízení (platforma Java Micro Edition), aplikace pro desktopové počítače (platforma Java Standard Edition) až po rozsáhlé distribuované systémy rozprostřené po celém světě (platforma Java Enterprise Edition). Tyto technologie se jako celek nazývají *platforma Java*.

2.2. Co je Java

Java je:

- objektově orientovaný jazyk,
- nemá ukazatele a z toho plynoucí přímý přístup do paměti,
- nemá globální proměnné,
- podporuje multitasking (multithreading) a synchronizaci,
- bezpečnost aplikací je zajištěna programovou kontrolou,
- syntaxe jazyka je jednoduchá a srozumitelná,
- prostřednictvím vyjímek lze zachytit chyby a neočekávané stavy,
- u interpretovaných programů se projevuje malá rychlost běhu programu (proti programům kompilovaným).

V The Java Language: A White Paper [2] charakterizuje SUN jazyk Java následujícím způsobem:

> *Java: jednoduchý, objektově orientovaný, distribuovaný, interpretovaný, robustní, bezpečný, nezávislý na architektuře, přenosný, vysoce výkonný, víceprocesní a dynamický jazyk.*

Na tomto místě je také záhodno uvést, že Java není vhodná pro řešení každého problému. Již výše bylo uvedeno několik nevýhod Javy.

Nevýhody jazyka Java – proti programovacím jazykům, které provádějí tzv. statický překlad (např. C/C++), je start programů psaných v Javě pomalejší, protože prostředí musí program nejprve přeložit a potom teprve spustit. I běh javovských programů je několikrát pomalejší, oproti programům se statickým překladem. Další nevýhodou projevující se hlavně u jednodušších programů je větší paměťová náročnost při běhu způsobená nutností mít v paměti celé běhové prostředí.

2.3. Technologie Java a Java platforma

Java platforma zastřešuje různé varianty použití programovacího jazyka Java pro vývoj a provoz různých typů aplikací. Java platforma se skládá ze dvou hlavních částí:

- Javovský virtuální stroj (Java Virtual Machine – JVM).
- Aplikačního programového rozhraní (Java Core API).

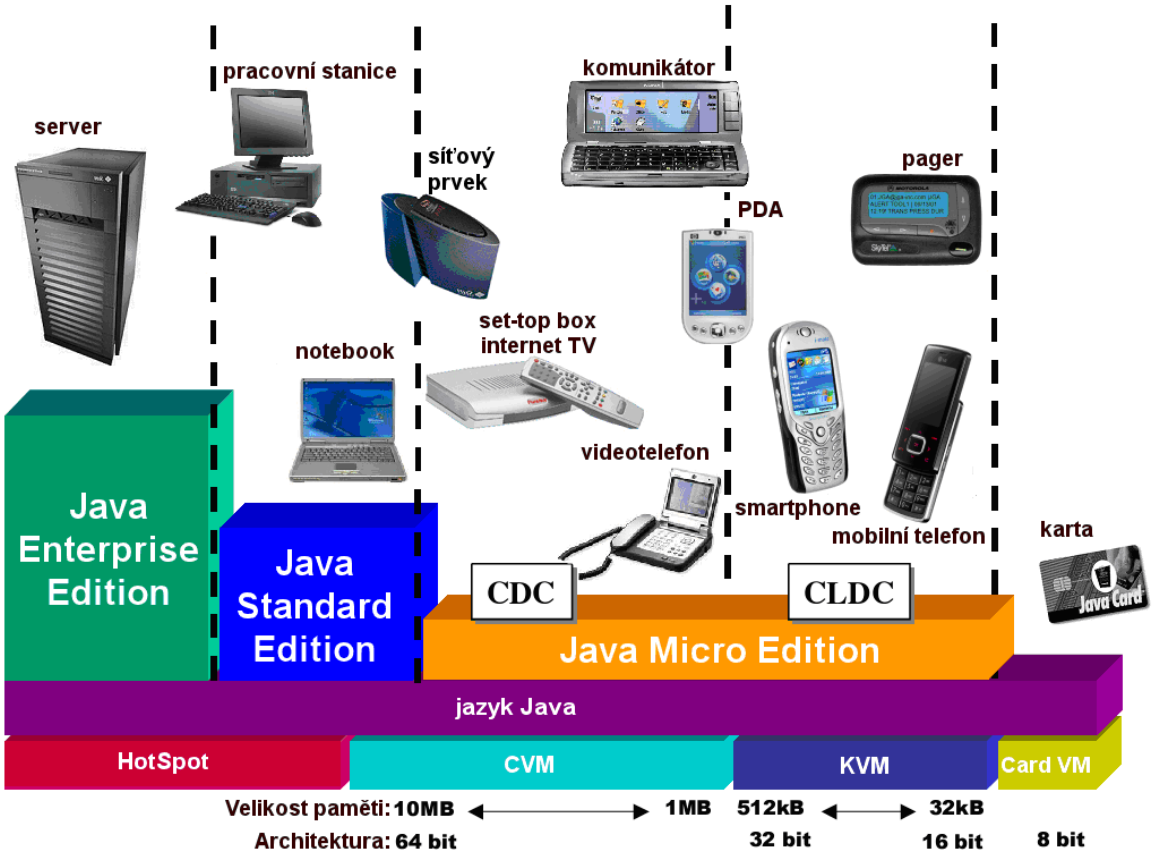
Javovský virtuální stroj (Java Virtual Machine) – virtuální stroj se skládá z *runtime systému*, což je část realizující vazbu na hardware, a *interpretu*, který vykonává bytový kód (pro urychlení může být interpret volitelně nahrazen tzv. JIT – Just-In-Time překladačem, případně technologií zvanou HotSpot compiler).

Aplikační programové rozhraní (Java Core API) – Java Core Application Programming Interface jsou základní knihovny pro psaní programů. Výhodou je, že tyto knihovny nemusí být s programem distribuovány. Java Core API skrývá značné množství knihovních tříd, které jsou považovány za standardní, čili musí se vyskytovat v každém prostředí, kde se Java používá. Prakticky to znamená, že programy pak obsahují pouze kód napsaný programátorem a jejich soubory pak proto mají relativně malou velikost.

Java platforma je poskytován firmou Sun Microsystems v následujících čtyřech dílčích platformách:

- **Java Card** – pro aplikace provozované v rámci tzv. „chytrých“ karet (např. platební a kreditní karty atp.). Technologie Java Card umožňuje bezpečný běh aplikací (Java Card aplety), na paměťových kartách a dalších, jim podobných, zařízeních s omezenou pamětí.
- **Java Micro Edition** (Java ME) – pro aplikace provozované na mobilních zařízeních (mobilní telefony, PDA, pagery atp.). Java ME má ve srovnání s Java SE vytvořen menší virtuální stroj a omezeny API funkce.
- **Java Standard Edition** (Java SE) – je základní Java prostředí. Implementace Java SE určuje aplikační programové rozhraní (API) a třídy umožňující vývoj a běh standardních klientských a serverových aplikací a apletů (aplet – aplikace běžících ve webových prohlížečích).
- **Java Enterprise Edition** (Java EE) – je skupina několika Java aplikačních rozhraní a nejavovských technologií. Hlavní využití je pro tvorbu vícevrstvých aplikací s možností distribuovaných aplikací. Spojuje vícevrstvé heterogenní aplikace v jeden celek. Využívá se pro podnikové aplikace a rozsáhlé informační systémy. V podstatě se jedná o nadstavbu nad platformou Java SE.

Jazyk Java prošel od první verze mnoha změnami a jeho standardní knihovny byly obohaceny o mnoho nových tříd a knihoven. Vedle všech změn jazyka proběhlo během několika let vývoje také mnoho dalších změn v javovské knihovně tříd. Ta se tak rozrostla z původních pár stovek tříd až na více než tři tisíce tříd (ve verzi Java SE 6.0.) Dále byla představena celá nová API a mnoho z původních tříd a metod z první verze bylo zrušeno či zcela přepracováno.



Obr. 1 – přehled a použití Java platformy

3. Java Micro Edition (Java ME)

Tento produkt představila v červnu 1999 firma Sun Microsystems, Inc. na konferenci Java-One jako mladšího bratříčka produktů Java SE a Java EE. Jak již bylo řečeno výše, Java byla původně vyvíjena pro spotřební elektroniku [3]. Důvodů proč pro malá zařízení nebyla použita Java SE je několik. Hlavním důvodem bylo, že zařízení, pro která je Java ME určena, mají specifické vlastnosti.

Obecně software pro tato zařízení musí mít minimalistické požadavky. Paměť alokovaná v zařízeních pro Java aplikace, Java třídy a javovský virtuální stroj je v řádu stovek kilobajtů. Dále mají tyto zařízení rozdílné ovládání, přistup k uživatelskému rozhraní, zobrazovací zařízení a interakci s uživatelem (např. klávesnice a displeje mobilních telefonů, dotykové displeje PDA, pagery apod.). Toto neumožňuje použití klasických Java SE API (Swing, AWT). Java ME je navržena tak, že klade určité minimální požadavky na dané zařízení. Splní-li dané zařízení požadavky, anebo je dokonce přesahuje, pak je zaručeno, že na tomto zařízení bude Java ME aplikace, pro kterou jsou tyto splněné požadavky dostačující, fungovat korektně.

3.1. Architektura Java ME

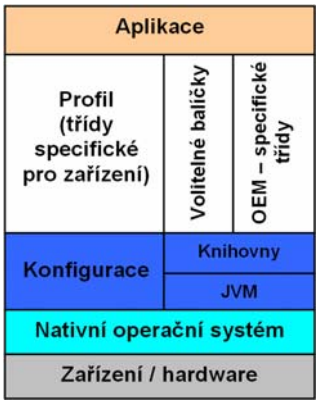
Z obecného pohledu Java ME definuje následující komponenty [4]:

- sada javovských virtualních strojů, z nichž každý se uplatňuje u jiného typu zařízení s různými nároky,
- skupina knihoven a API (konfigurace a profily), které lze spustit na každém virtuálním stroji,
- různé nástroje pro vývoj a nastavení zařízení.

Na Obr. 2 je zobrazen přehled vztahů v pracovním prostředí. Nejníže je samotný hardware zařízení. Nad ním pracuje operační systém. Nad operačním systémem pracuje javovský virtuální stroj a dále specifická konfigurace daného zařízení. Nad toto je postaven jeden nebo několik profilů spolu s volitelnými balíčky a specifickými třídami. Nejvýše stojí samotná Java aplikace.

Konfigurace a *profily* jsou hlavní elementy zahrnující Java ME modulární návrh. Tyto dva elementy umožňují podporu pro široké spektrum zařízení, které obsahují možnost běhu Java ME programů.

Java ME konfigurace specifikuje minimální sadu rysů pro určitou kategorii zařízení. Všechny zařízeni v této kategorii mají podobné požadavky na paměť a procesorový výkon. Konfigurace je specifikace, která identifikuje úroveň systému zařízení a zároveň definuje rysy jazyka Java, charakteristiky a rysy javovského virtuálního stroje a Java knihoven, které jsou minimálně podporovány. Díky tomu mohou vývojáři software očekávat jistou úroveň systémové podpory dostupnou pro kategorii zařízení, která používají danou konfiguraci. Výrobci zařízení implementující určitý profil umožňují reálné platformě dané kategorie zařízení, být kompatibilní s jinými zařízeními, než daná konfigurace specifikuje [8].



Obr. 2 – přehled architektury pracovního prostředí Java ME

Java ME profil specifikuje rozhraní aplikační úrovně, pro podobnou třídu zařízení. Implementace profilu obsahuje sadu tříd a knihoven jazyka Java, které poskytují rozhraní aplikační úrovni. Profil tak může teoreticky specifikovat všechny druhy funkcionalit a služeb zařízení. Typicky profil obsahuje knihovny, jež jsou mnohem více specifické pro charakteristiku kategorie daných zařízení [8].

Aplikace, která v architektuře stojí úplně na vrcholu, může používat pouze knihoven a tříd, jež jí poskytují pod ní ležící úrovně (profil, konfigurace). Profilů může být v zařízení použito několik, avšak konfigurace může být použita pouze jedna (viz Obr. 2).

3.2. Konfigurace

Výše uvedený text o profilech, konfiguracích a platformních definicích je poněkud abstraktní. Následující část mnohem konkrétněji popisuje charakteristik nynějších zařízení.

Konfigurací se definuje horizontální členění produktů založené na dostupném množství paměti a výkonu procesoru jednotlivých zařízení. Jakmile má tyto informace k dispozici, konfigurace zjistí následující údaje [4]:

- podporované rysy programovacího jazyka Java,
- podporované rysy virtuálního stroje Javy,
- podporované základní javovské knihovny a API.

Java ME v současné době definuje pouze dvě konfigurace pro nejrozšířenější zařízení, kterými jsou CDC (Connected Device Configuration) a CLDC (Connected Limited Device Configuration). Hranice mezi těmito dvěmi konfiguracemi není ostrá. Je možné, aby jedno zařízení mohlo podporovat obě konfigurace. Díky technickému pokroku a snaze výrobců o komplexní zařízení, se bude současná hranice ještě více rozmazávat.

Ve velice hrubém rozdělení lze říci, že určující rozdíl mezi CLDC a CDC spočívá především v nutném minimálním množství paměti, kterým musí zařízení disponovat a v tom, zda je v zařízení přítomno či nepřítomno uživatelského rozhraní a baterie.

3.2.1. Connected Limited Device Configuration (CLDC)

Tato konfigurace je určena pro malá zařízení, mezi které patří mobilní telefony, pagery, PDA atd. CLDC byl navržen jako nejmenší společný jmenovatel, který lze nalézt u mnoha různých zařízeních. U CLDC neexistují žádné volitelné rysy. U zařízení, která CLDC podporují lze využít vše, co konfigurace CLDC poskytuje.

CLDC je specifikováno ve dvou verzích. A to CLDC 1.0 (JSR 30) a CLDC 1.1 (JSR 139).

CLDC 1.0 specifikace – předpokládá, že se javovský virtuální stroj, konfigurační knihovny, knihovny profilů a aplikace musí vejít do celkové velikosti paměti 160 až 512 kB. Zahrnuje zařízení, u kterých se předpokládají tyto vlastnostmi [9]:

- min. 128 kB stálé paměti (např. ROM, při vypnutí zařízení zůstávají data zachována) dostupné pro javovský virtuální stoj a CLDC knihovny,
- min. 32 kB dočasné paměti, která je k dispozici pro běh programu,
- 16-ti nebo 32-ti bitový procesor s minimální taktovací frekvencí 25MHz,
- zařízení je primárně napájeno z baterie (nízká spotřeba energie),
- zařízení je propojitelné s některým typem sítě často bezdrátovým přerušovaným dvousměrným připojením a s omezenou šířkou pásma (často 9600 bps a méně),
- zařízení může mít i poměrně propracované uživatelské rozhraní, ale není to povinností.

Z javovských virtuálních strojů CLDC byly odstraněny mnohé rysy jazyka Java (z důvodu paměťové náročnosti nebo zátěže procesoru). Kvůli tomu je třeba mít na paměti následující omezení virtuálních strojů CLDC 1.0. V těchto chybí následující rysy: podpora plovoucí řádové čárky, finalizace, omezené možnosti zpracování chyb, JNI (Java Native Interface) není k dispozici, uživatelské zavaděče tříd (class-loadery), podpora reflexe není k dispozici, skupiny vláken a podprocesy typu démon, slabý typ reference (weak reference).

CLDC 1.1 specifikace – CLDC 1.1 je revizí CLDC 1.0 specifikace. V podstatě se jedná o navazující vydání, které je plně kompatibilní se specifikací CLDC 1.0. Pouze byly přidány nové funkcionality [10]: podpora plovoucí řádové čárky, podpora typu slabých referencí (weak reference), přidána jedna nová třída pro obsluhu výjimek a různé minoritní změny v knihovnách.

Oproti CLDC 1.0 jsou hlavní rozdílné charakteristiky CLDC 1.1 tyto:

- nejméně 192 kB celkové velikosti paměti dostupné pro Java platformu,
- min. 160 kB stálé paměti (např. ROM, při vypnutí zařízení zůstávají data zachována) dostupné pro javovský virtuální stoj a CLDC knihovny,
- min. 32 kB dočasné paměti, která je k dispozici pro běh programu.

3.2.2. Connected Device Configuration (CDC)

Tato konfigurace je zaměřena na zařízení s větším paměťovým prostorem (více než 2 MB), s 32 bitovým procesorem a síťovým připojením. Může se jednat o různé "vylepšené" spotřebiče (např. TV s internetem). Zařízení pro tuto konfiguraci disponují lepším systémovým vybavením než zařízení pro konfiguraci CLDC. Patří sem videotelefony, set-top boxy, přídavná zařízení, televize po internetu, domácí spotřebiče a navigační systémy pro vozidla.

CDC je specifikováno ve dvou verzích. A to CDC 1.0 (JSR 36) a CDC 1.1 (JSR 218). Více viz literatura [11], [12].

3.3. Javovské virtuální stroje (Java Virtual Machine)

CDC a CLDC stanoví své vlastní skupiny podporovaných rysů z javovského virtuálního stroje. Proto je u každého z nich zapotřebí jiný javovský virtuální stroj. Jelikož CLDC podporuje mnohem méně rysů jazyka Java, je jeho virtuální stroj mnohem menší než virtuální stroj CDC. Virtuální stroj pro CDC se nazývá CVM. Pro CLDC je možno použít jeden ze dvou virtuálních stojů. A to buď KVM (první generace), anebo virtuální stroj s formálním názvem Monty (druhá generace – dnes nejpoužívanější) [13].

3.3.1. Virtuální stroj KVM (Kilo Virtual Machine)

KVM je kompletní javovské pracovní prostředí. Je kompaktní, přenositelné a od základu navrhováno pro malá zdroji omezená zařízení s potřebou několika málo stovkami kilobajtů celkové paměti. Od toho je odvozen název „Kilo“ Virtual Machine. KVM je napsané v programovacím jazyku C, takže jej lze snadno přenést na různé platformy, které disponují překladačem jazyka C.

KVM je popisován takto [14]:

- určeno pro 16-bitové a 32-bitové CISC/RISC procesory s minimální frekvencí 25 Mhz,
- s nároky na statickou paměť od 40 do 80 KB (v závislosti na nastavení překladu a použitých knihoven),
- modulární a přizpůsobivé,
- za daných podmínek co možná nejrychlejší a kompletní bez nutnosti obětovat jiné, výše zmíněné vlastnosti designu.

3.3.2. Virtuální stroj Monty

Projekt Monty je vysoce výkonný virtuální stroj pro embedded zařízení. Jedná se o druhou generaci virtuálního stroje pro CLDC. Vychází z architektury HotSpot. Virtuální stroj Monty byl navrhován s ohledem na budoucí vývoj u zařízení s omezenými zdroji. Monty zachovává vlastnosti a rysy KVM s mírně zvýšenými nároky na prostředky. Avšak díky nové koncepci je proti KVM mnohonásobně rychlejší.

Při návrhu se vycházelo z těchto předpokládaných vlastností nových zařízení [15]:

- typ CPU – nejvíce používané ARM,
- rychlost CPU 30-400 MHz,
- 1-4 MB paměti RAM,
- 8-24 MB paměti ROM/Flash,
- 1 MB a méně paměti pro běh Javy.

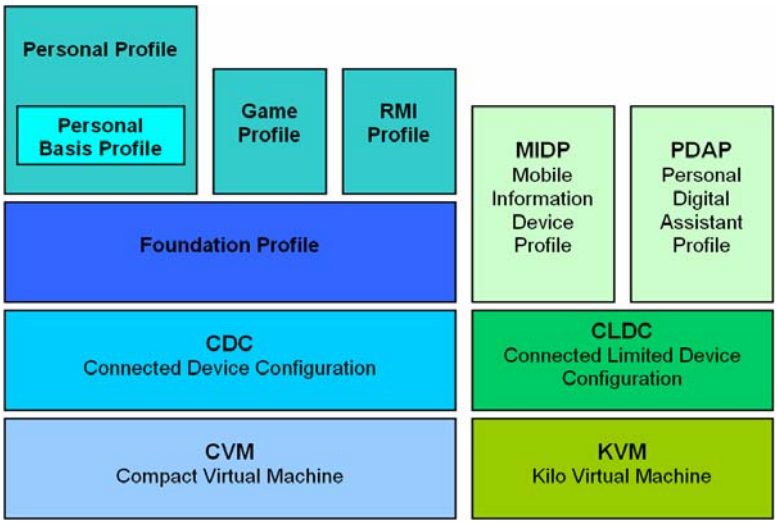
3.4. Profily Java ME pro CLDC

Profily doplňují konfiguraci tak, že výsledná aplikace je lépe přizpůsobena vlastnostem dané skupiny zařízení.

Personal Digital Assistant (PDA) Profile – tento profil je specifikován v JSR 75. Skládá se ze dvou volitelných balíčků pro společné použití na PDA. Poskytuje API pro uživatelské rozhraní. Jeden balíček slouží k implementaci přístupu k datům z PIM (Personal Information Manager). Druhý slouží pro přístup k souborovému systému.

Mobile Information Device Profile (MIDP) – protože je konfigurace CLDC určena pro velmi rozmanitou škálu zařízení, existuje zde velmi vysoký potenciál pro vytvoření specifických profilů pro každý druh zařízení. Asi nejpopulárnějším a nejznámějším profilem je Mobile Information Device Profile (MIDP), který je určen pro mobilní telefony a pagery. V současné době existují specifikace MIDP 1.0 (JSR 37) a MIDP 2.0 respektive jeho revize MIDP 2.1 (JSR 118). MIDP vyžaduje referenční implementaci CLDC a poskytuje třídy pro tvorbu stažitelných aplikací, které fungují na mobilních zařízeních. Také umožňuje stahování nových služeb zákazníkům, jako jsou mobilní hry, komerční aplikace a personalizované služby.

Celkový pohled na Java ME architekturu a zobrazení vzájemných vztahů mezi javovskými virtuálními stroji, konfiguracemi a profily ukazuje Obr. 3.

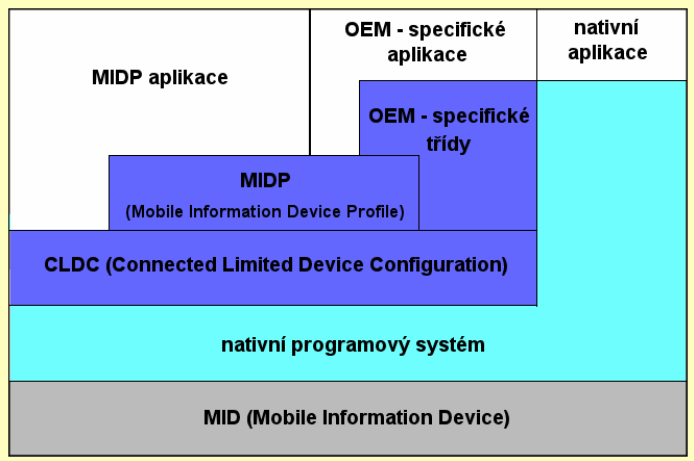


Obr. 3 – celkový pohled na členění a architekturu Java ME

3.5. *Mobile Information Device Profile (MIDP) podrobně*

Ze základního členění (viz Obr. 3) Java ME vyplývá, že MIDP je vrcholem CLDC a definuje otevřené prostředí pro vývoj aplikací. Jak již bylo uvedeno výše, v současné době existují specifikace MIDP 1.0, MIDP 2.0 respektive jeho nejnovější revize MIDP 2.1. Na tyto se nyní podíváme podrobněji.

Jak je vidět na následujícím Obr. 4, kromě MIDP implementace může výrobce zařízení dodávat také rozšiřující API (tzv. OEM – specifické třídy). Aplikace, které tyto třídy využívají, jsou však závislé na daném zařízení, pro které byly vytvořeny. Například Siemens dodával k mobilním telefonům SL45i také API, které umožňovalo zasílání krátkých textových zpráv SMS (což tehdejší specifikace MIDP samo o sobě nenabízela). Na jedné straně je to výhodné, protože toto API umožňuje využít funkce telefonu, které by normálně MIDP API neumělo obsloužit. Na druhé straně však takto vytvořené aplikace nejsou přenositelné na jiný telefonní přístroj, což je poněkud v rozporu s filozofií jazyka Java. Proto dnes vznikají určitá standardizovaná API, při jejichž specifikaci spolupracují přední výrobci zařízení. Spolu související úrovně architektury MIDP jsou znázorněny stejnou barvou.



Obr. 4 – pohled na MIDP architekturu

3.5.1. MIDP 1.0

Standard MIDP definuje mobilní informační zařízení, jako druh zařízení s následujícími minimální vlastnostmi [16]:

- displej – minimální rozlišení 96 x 54 pixelů, minimální bitová hloubka barev 1b, poměr stran obrazu přibližně 1:1,
- vstupní zařízení – tlačítková nebo dotyková klávesnice,
- paměť – 32 kB dynamické paměti pro práci Javy, 128 kB statické paměti pro komponenty MIDP, 8 kB statické paměti pro ukládání dat z aplikací,
- páce v síti – obousměrný (přerušovaný) síťový provoz, s omezenou šířkou pásma.

MIDP se zaměřuje na následující oblasti, jež CLDC řešilo obecně: správa průběhu aplikací, uživatelské rozhraní a události, připojitelnost k síti, ukládání dat v zařízeních.

3.5.2. MIDP 2.0

MIDP 2.0 specifikace je rozšířením MIDP 1.0 specifikace. Zpětná kompatibilita je zachována (obsahuje stejné třídy jako MIDP 1.0), přičemž přidává některé vylepšení a novinky [17]. Proti MIDP 1.0 doznaly nároky na minimální vlastnosti těchto změn:

- paměť – 128 kB dynamické paměti pro práci Javy, 256 kB statické paměti pro komponenty MIDP,
- zvuk – schopnost přehrávat tóny, ať už hardwarově či softwarově.

Nové vlastnosti MIDP 2.0 jsou především: podpora HTTPS, práce s multimédii (Media API – MAPI), uživatelské rozhraní, Game API, podpora práce s RGB obrázky, ověřování důvěryhodnosti MIDletu, Push architektura, sdílené úložiště dat mezi MIDlety (MIDP 1.0 neumožňovalo MIDletu číst data jiného MIDletu), O-T-A 2.0 (Over The Air).

Pozn. v současné době (2008) je ve stádiu návrhu a diskuze verze MIDP 3.0 (JSR 271) [18].

3.6. MIDlet podrobněji

Java ME aplikace pro MIDP se nazývají MIDlety. Úkolem následujícího textu je rozebrat strukturu MIDletu, popsat jeho jednotlivé součásti a vysvětlit pojmy.

3.6.1. Správa průběhu aplikací

Jak již bylo řečeno, Java aplikace pro MIDP se nazývají *MIDlety*. MIDlety jsou tvořeny jednou nebo více třídami Javy. V zařízení se s MIDletem pracuje jako s celkem. MIDlet musí rozšiřovat speciální třídu `MIDlet` a v telefonu běží z bezpečnostních důvodů v tzv. sandboxu, neboli na svém vlastním „pískovišti“, které nemůže opustit. Podobně jako javovské applety také MIDlety mají svůj průběh, když pracují na mobilním zařízení. MIDlet obsahuje třídy a metody pro řízení průběhu aplikace. MIDlet má svou specifickou kostru, kterou musíme při programování dodržet. Jak ukazuje Obr. 5 MIDlet se může nacházet ve třech stavech [4]: *přerušený*, *aktivní* a *zrušený*.

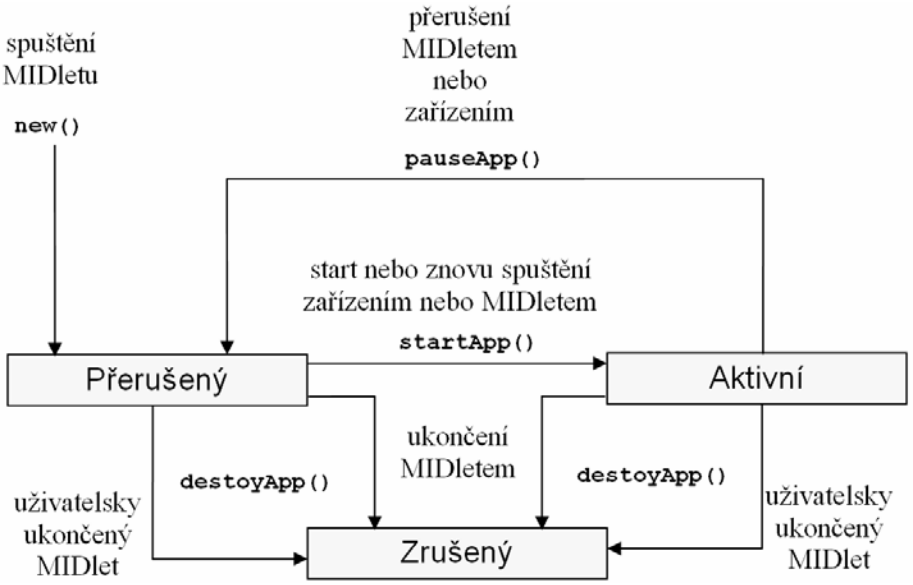
Běh aplikace a její přechod mezi stavy řídí aplikační manažer AMS (Application Management Software). Když je MIDlet nahrán, je jeho počáteční stav *Přerušený* – jsou volány statické inicializátory a poté také jeho veřejný konstruktor. Když dojde k vyvolání výjimky během průběhu konstruktoru, přejde MIDlet do stavu Zrušený (nespustí se).

Po provedení konstruktoru přejde MIDlet do stavu *Aktivní* – je volána metoda `StartApp()`. Tato metoda indikuje, že se MIDlet přesouvá ze stavu Přerušený do stavu Aktivní. MIDlet inicializuje všechny objekty potřebné pro stav aktivity a nastaví odpovídající obrazovku.

V tuto chvíli MIDlet běží a uživatel s ním může pracovat. Aplikaci může přerušit buď systém, anebo sám program. MIDlet se přesouvá do stavu *Přerušený* – volá se metoda `pauseApp()`. Znamená to, že se přeruší všechna vlákna, která jsou v daném okamžiku

aktivní, a může se nastavit obrazovka, která se objeví, bude-li MIDlet znovu aktivován. Potřebná data lze uložit a později opakovaně použít, jakmile bude MIDlet opět aktivní.

MIDlet může přejít do stavu *Zrušený* ze stavu Aktivní, nebo ze stavu Přerušený. Stav Zrušený je vyvolán buď uživatelsky, anebo jej vyvolá sám program MIDlet. Při přechodu do stavu Zrušený – se volá metoda `destoyApp()`. Metoda by měla uvolnit, anebo uzavřít všechny zdroje, které byly při průběhu MIDletu využívány. Dále by metoda měla uložit všechna data, která je zapotřebí uchovat pro pozdější použití.



Obr. 5 – přechody mezi stavy MIDletu

3.6.2. Grafické prostředí MIDletu

Pro práci s grafickým prostředím MIDletu lze použít balík `javax.microedition.lcdui`, který poskytuje vysokoúrovňové API. Toto API ulehčuje definici vzhledu aplikace, avšak výsledný vzhled grafického výstupu je závislý na konkrétní implementaci v zařízení (tzn. v každém zařízení může vypadat jinak). Přístup ke klávesnici je také zprostředkován za pomocí tohoto API. Třídy vysokoúrovňového API jsou odvozeny (zděděny) od třídy `Screen`. Vysokoúrovňového API se využívá především tam, kde není u aplikace důležitý přesný vzhled, ale funkčnost poskytovaná aplikací.

Druhou možností pro práci s grafickým prostředím možno použít balík nízkoúrovňového API `javax.microedition.lcdui.game`. Pomocí nízkoúrovňového API lze dosáhnout větší kontroly výsledného grafického zobrazení. Toto API umožňuje přesněji definovat výstup a v důsledku toho by se zobrazení na různých zařízeních mělo shodovat. Využitím tohoto se však omezuje přenositelnost (ne každé zařízení má stejný počet a funkci kláves, využívá jiný hlavní ovládací prvek apod.) Tohoto API se využívá především u her. Třídy nízkoúrovňového API jsou odvozeny (zděděny) od třídy `Graphics` nebo `Canvas`.

Třída `Display` slouží jako rozhraní manažeru aplikace a umožňuje aplikaci ovládat displej. Manažer aplikace umožňuje zobrazovat MIDletu nové obrazovky, pokud je ve stavu aktivní.

K ovládání aplikace jsou definovány příkazy, jež jsou reprezentací stisku klávesy. Každé obrazovce MIDletu jsou přiřazeny příkazu typu `Command`. Záleží na implementaci v zařízení, které klávese bude příkaz přiřazen (což bývá někdy nepříjemné při návrhu uživatelského rozhraní pomocí vysokoúrovňového API). Každému příkazu lze přiřadit popis (jednoduchý název), který se zobrazí na displeji zařízení. Reakce na události těchto příkazů zajišťují posluchači zaregistrovaní pomocí rozhraní `CommandListener`.

Mobilní zařízení poskytuje také další funkce, o které se ovšem nestará samotná aplikace, ale tyto funkce zajišťuje implementace MIDP.

3.6.3. Application Management System (AMS)

Manažer aplikací (AMS) někdy také označován jako (MIDlet management software) je na MIDP zařízeních předinstalovaný program, který je zodpovědný za životní cyklus každé aplikace. Aplikační manažer běží neustále v systému MIDletů. Vypnutí aplikačního manažeru nastane pouze tehdy, kdy je vypnut celý javovský virtuální stroj. Konkrétně je manažer aplikací zodpovědný za [19]:

- instalaci MIDletů,
- aktualizaci MIDletů,
- odstranění MIDletů,
- spuštění MIDletů,
- zobrazení informací o MIDletech,
- aktualizaci nastavení MIDletů.

Manažer aplikací musí pro všechny MIDlety poskytovat provozní správu. Ta je požadována specifikací MIDP profilu. Dnes nejrozšířenější profil MIDP 2.0 požaduje:

řízení aplikací – interakci s uživatelem k řízení aplikací nainstalovaných na zařízení,

odinstalátor – interakci s uživatelem pro odinstalování aplikací (někdy je tato funkce součástí řízení aplikací),

grafický instalátor – umožnění interakce s uživatelem při instalaci aplikací,

zjištění aplikací – umožňuje uživateli zjistit pomocí O-T-A, že jsou k dispozici nové aplikace a automaticky vyvolat grafický instalátor, v případě uživatelova rozhodnutí, že si aplikaci stáhne,

persistentní úložiště – uložení aplikací a jejich dat napříč mnohonásobnému užití zařízení,

MIDP běhové prostředí – běh instalovaných MIDletů (což je v podstatě zajišťuje Java běhové prostředí).

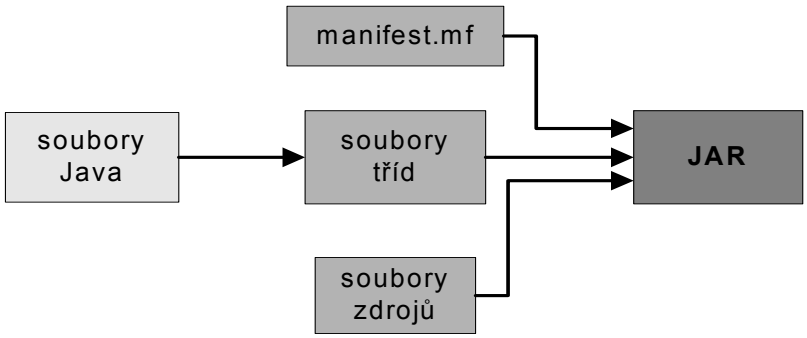
Manažer aplikací poskytuje pro instalátor také služby jako jsou autentizace a autorizace digitálně podepsaných aplikací (podepsaných pomocí certifikátů X.509), instalační služby (parsování JAD souborů, extrakce informací z JAD souborů, extrakce zdrojů z JAR souborů, srovnávání informací z JAD a JAR souborů, porovnávání verzí MIDletů během aktualizace), uložení MIDletů do persistentního úložiště, automatickou inicializaci MIDletů (tzv. Push), automatické testy.

3.6.4. Java Archive (JAR)

Formát souboru *Java Archive* (JAR) umožňuje zabalit několik souborů do jednoho archivu. Typicky se v souboru s touto příponou nacházejí vlastní data aplikace (zdrojové kódy, informace o MIDletu) a zdroje (např. ikonka aplikace, obrázky atd.). Jeden soubor JAR může obsahovat i více aplikací. Soubory jsou zkomprimovány (JAR soubor je zabalen pomocí ZIP formátu). Konkrétně JAR soubor obsahuje:

- Manifest soubor,
- Java třídy (CLASS) pro MIDlety,
- ostatní zdroje (RES) jako například obrázky.

Na Obr. 6 je graficky znázorněna skladba JAR souboru.



Obr. 6 – schéme vzniku JAR souboru

Formát souboru JAR poskytuje mnoho výhod. Mezi ně patří: bezpečnost (lze digitálně podepsat obsah JAR souboru), komprese (efektivnější využití paměťového prostoru), zkrácení času stahování (v JAR soubor jsou všechny části MIDletu), přenositelnost, balící techniku pro rozšíření (lze přidat funkcionality, které nejsou obsaženy v jádře Java platformy jako např. Java3D, JavaMail).

3.6.5. JAR MANIFEST (MANIFEST.MF)

JAR MANIFEST obsahuje definici vlastností MIDletu. Tyto vlastnosti popisují obsah JAR souboru, který používá program pro správu aplikací k identifikaci a instalaci soupravy MIDletů. Jedná se o textový soubor s touto příponou MF. Většinou má název MANIFEST.MF. Povinné údaje musí MANIFEST soubor obsahovat. Nepovinné mohou být vynechány [4].

3.6.6. Java Application Deskriptor (JAD)

Deskriptor aplikace JAD (Java Application Deskriptor) je textový souboru podobný MANIFEST. Na rozdíl od něj však není komprimovaný v souboru JAR. Deskriptor aplikace má příponu jad. Při instalaci sady MIDletů se nejprve do telefonu přenese deskriptor aplikace. Manažer aplikací zkontroluje podle popisu, zda danou aplikaci lze spustit na daném zařízení (podporované verze profilů, dostatek místa atd.). Poté se soubor JAR nahrává do zařízení podle atributů v něm obsažených. Povinné údaje musí JAD soubor obsahovat. Nepovinné mohou být vynechány [4]. Kromě povinných atributů může deskriptor obsahovat další libovolné atributy, které nezačínají řetězcem MIDlet. K těmto atributům má programátor přístup v třídě MIDlet přes metodu `getAppProperty(String key)`.

3.6.7. Bezpečnostní model MIDP

Bezpečnostní modely MIDP 1.0 a MIDP 2.0 se výrazně liší. Snahou bezpečnostních modelů je ochránit uživatele před škodlivým softwarem. To je zařízeno tak, že MIDlety jsou kontrolovány v přístupu k citlivým rozhraním a jejich přístup k nim řízen pomocí bezpečnostních domén. Ve specifikaci MIDP 1.0 byl nastavený model velice restriktivní [16]. Velice zjednodušeně se dá říci, že MIDlet běžel v jakémsi sandboxu odkud neměl skoro nikam umožněn přístup. Takto se vývojáři platformy Java ME snažili chránit uživatele proti vnějším hrozbám a k jakémukoli MIDletu bylo přistupováno jako k nedůvěryhodnému (tzv. nedůvěryhodná doména).

Bezpečnostní model MIDP 2.0 zůstává zpětně kompatibilní s MIDP 1.0, ale oproti němu přináší možnost přesunout MIDlet z nedůvěryhodné domény do domény důvěryhodné. Toto je možno při digitálním podepsání MIDletu pomocí X.509 certifikátu. Pokud takto podepsaný MIDlet při instalaci projde v pořádku všechny ověřovací mechanizmy a mobilní zařízení má zařazenu certifikační autoritu, jež vydala certifikát, jako důvěryhodnou, je MIDlet přesunut do důvěryhodné domény.

Na základě domény je poté rozhodováno, jak aplikační manažer reaguje na požadavky aplikace při přístupu ke kontrolovaným rozhraním. MIDP 2.0 definuje otevřený systém přístupových práv [28]. Pro vytvoření jakéhokoli spojení, musí mít MIDlet příslušné přístupové práva. Přístupové práva definované v MIDP 2.0 odpovídají síťovým protokolům, ale architektura umožňuje volitelným balíčkům definovat jejich vlastní přístupové práva. Každé přístupové právo má unikátní jméno. Jsou definována následující přístupová práva MIDP 2.0:

- javax.microedition.io.Connector.http,
- javax.microedition.io.Connector.socket,
- javax.microedition.io.Connector.https,
- javax.microedition.io.Connector.ssl,
- javax.microedition.io.Connector.datagram,

- javax.microedition.io.Connector.serversocket,
- javax.microedition.io.Connector.datagramreceiver,
- javax.microedition.io.Connector.comm,
- javax.microedition.io.PushRegistry.

Přestože jména přístupových práv vypadají podobně jako jména tříd, nejsou to dané třídy (např. `javax.microedition.io.Connector.https`, přístupové právo dává možnost MIDletu vytvořit HTTPS spojení za použití třídy `javax.microedition. .io.Connector`). MIDlet nenabývá přístupová práva explicitně v kódu, ale na základě bezpečnostní domény.

3.6.8. Bezpečnostní domény

Bezpečnost v MIDP se sestává ze dvou částí:

- přístupových práv, která jsou standardně povolena (jsou MIDletu garantovány) a z přístupových práv, která musí být konzultována s uživatelem,
- kritérií nutných k zařazení do konkrétní bezpečnostní domény.

Každému MIDletu je při instalaci přidělena některá z následujících bezpečnostních domén :

- manufacturer (výrobce telefonu),
- operator (operátor),
- trusted 3rd party (důvěryhodná aplikace),
- untrusted 3rd party (nedůvěryhodná aplikace).

Každá bezpečnostní doména má určité úrovně přístupových práv ke kontrolovaným rozhraním (např. v doméně výrobce telefonu mohou být práva k nějakému kontrolovanému rozhraní nastavena tak, že jsou vždy povolena, kdežto v nedůvěryhodné doméně může být přístup k tomuto rozhraní zakázán). Přístupová práva jsou seskupena ve funkčních skupinách [29]. Např. skupiny přístupu k síti, zasílání zpráv (SMS, MMS, CBS), automatický start aplikace, komunikace s kartou (více viz [29]). V mnoha případech poté co vývojář digitálně podepíše MIDlet certifikátem (který mobilní zařízení zná), dosáhne toho, že MIDlet bude spadat do důvěryhodné domény. Následkem toho bude, že MIDlet bude méně omezován v přístupu ke kontrolovaným rozhraním.

3.6.9. Typy přístupových práv

MIDlet má přístup definovaný pro každou funkční skupinu, která je podporovaná v telefonu. Podle definované polity bezpečnostní domény v zařízení může být nastaveno jedno z následujících přístupových práv [29]:

- vždy povolit,

- zeptat se poprvé / zeptat se jednou za spuštění,
- zeptat se pokaždé,
- nepovoleno.

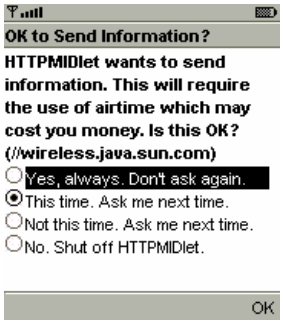
Každá z bezpečnostních domén obsahuje pro určité akce přístupová práva, které jsou vždy povolena (kde není třeba zásahu uživatele) a přístupová práva, které musí být konzultovány s uživatelem. S uživatelem konzultované přístupová práva jsou ve třech variantách, co se týká délky jejich trvání (ve specifikaci nazývány jako módy interakce). Poté co se MIDlet uživatele zeptá zde povolit či zakázat úkon, má uživatel možnost vybrat z následujících variant délky doby platnosti:

jednorázové povolení (oneshot) – MIDP implementace si výsledek rozhodnutí uživatele nikde neukládá a ptá se kdykoli je třeba udělit povolení,

povolení pro sezení (session) – MIDP implementace si výsledek uživatelova rozhodnutí pamatuje, dokud není MIDlet ukončen,

trvalé povolení (blanket) – MIDP implementace si výsledek uživatelova rozhodnutí pamatuje trvale a je platné, dokud není MIDlet odinstalován.

Příklad: Mějme MIDlet který spustíme v emulátoru programu Sun Java Wireless Toolkit for CLDC a který spadá do nedůvěryhodné domény. MIDlet chce komunikovat pomocí HTTP spojení. Ve chvíli kdy MIDlet zavolá metodu `Connector.open()` bezpečnostní model způsobí to, že se emulátor zeptá na povolení (určení) přístupových práv (viz Obr. 7)



Obr. 7 – dotaz emulátoru na přístupová práva MIDletu

Nyní má uživatel možnost povolit, anebo zakázat připojení. V případě, že uživatel povolí přístup, pokračuje MIDlet normálně v běhu programu podle programového kódu. V případě, že uživatel odepře přístup, programový kód `Connector.open()` způsobí výjimku `SecurityException` Tuto výjimku lze v programovém kódu odchytit a zpracovat tak, aby nedošlo k narušení stability MIDletu a jeho případném ukončení v důsledku této výjimky. Náhlé ukončení MIDletu bez jakékoli zprávy uživateli je pro uživatele matoucí.

3.6.10. Instalace MIDletu na zařízení

Většinou existuje více způsobů, jak lze MIDlet na zařízení nainstalovat. Tyto způsoby záleží na výrobci daného zařízení, které možnosti instalace umožní. Jeden způsob instalace MIDletu by měly podporovat všechny zařízení implementující profil MIDP (více viz. [16], [17]). Jedná se o instalaci přes Internet (Over-The-Air – O-T-A) někdy také nazývanou instalací přes vzduch. Každé zařízení s profilem MIDP musí mít prohlížeč, jež tento druh instalace zvládne. V drtivé většině zařízení se jedná o nativní prohlížeč wapových a internetových stránek, který je součástí firmware zařízení. Uživatel pomocí tohoto prohlížeče zadá adresu, kde se nachází deskriptor aplikace (JAD soubor), prohlížeč tento soubor stáhne a mechanismy platformy Java ME v zařízení implementované (toto je také jedna z činností aplikačního manažeru) se postarají o stažení a nainstalování. Ale to pouze v případě, že zařízení splňuje požadavky aplikace (podporovaná verze konfigurace, profilu, volitelných balíčků, velikost volné paměti apod.). V případě nesplnění požadavků je uživatel informován o nemožnosti instalace MIDletu. Výhodou takovéto instalace je, že zařízení má možnost aktualizace aplikace (buď automatické, anebo vyvolané uživatelsky), kdy se při aktualizaci stáhne nová verze a současná verze v telefonu se přeinstaluje. A to tak, že veškerá MIDletem dříve uložená data zůstanou zachována a jsou přístupná nové verzi MIDletu.

Mnoho mobilních zařízení také poskytuje možnost instalace pomocí dalších způsobů, jako jsou nahrání MIDletu pomocí kabelu, infraportu, Bluetooth, přenesením pomocí paměťové karty. Nevýhodou instalací je, že většina mobilních zařízení nevyžaduje k instalaci aplikace deskriptor MIDletu, a tak není nikterak provedena kontrola, zda je MIDlet kompatibilní s daným zařízením a bude korektně na zařízení fungovat.

3.7. Volitelné balíčky (Optional Packages) Java ME

Původně Java ME platforma obsahovala pouze konfiguraci a profil. Jak již bylo uvedeno konfigurace definuje minimální Java běhové prostředí (kombinace JVM a sadu API) pro určitou rodinu zařízení (CDC nebo CLDC). Profil definuje sadu API, která jsou přidána ke konfiguraci, aby bylo lépe využito vlastností daného zařízení.

Nicméně jak se Java ME platforma vyvíjela, bylo čím dál tím více jasné, že vývojáři potřebují třetí kategorii Java ME komponent – *volitelné balíčky*. Java ME specifikace (uvedená jako JSR 68) definuje konfigurace, profily a volitelné balíčky [22]. Cílem této specifikace bylo předejít vytváření různých vzájemně nekompatibilních balíčků. V důsledku toho by aplikace přestaly být přenositelné.

3.7.1. Co je volitelný balíček?

Volitelný balíček je sada API, ale na rozdíl od profilu nedefinuje kompletní aplikační prostředí. Volitelný balíček je vždy použit ve spojení s konfigurací, anebo profilem (nikdy ne samostatně). Rozšiřuje běhové prostředí o podporu vlastností zařízení, které nejsou dostatečně univerzální, aby byly definovány v profilu, anebo které potřebují být sdíleny různými profily (např. Wireless Messaging API – WMA). Protože jsou volitelné balíčky specifikovány skrze Java Community Process (JCP), každý má svou *referenční implementaci* (RI) a *soubor nástrojů pro test kompatibility* (test compatibility toolkit –

TCK). Kromě toho JCP pomáhá výrobcům s implementací volitelných balíčků jako části jejich běhového prostředí. RI a TCK také zabezpečuje, že tato implementace je konzistentní a korektní, a nezáleží na jakém zařízení je používána.

Volitelných balíčků existuje velké množství. Jako příklady volitelných balíčků lze uvést Remote method invocation (RMI) definované JSR 66, Wireless Messaging API (WMA) definované JSR 120 a JSR 205, Mobile Media API (MMAPI) definované JSR 135 a další (více viz stránky java.sun.com).

3.7.2. Používání volitelných balíčků

Při vývoji aplikace se s volitelným balíčkem pracuje stejně, jako s jakoukoli sadou Java tříd s definovaným umístěním v cestách tříd Java překladače. Samozřejmě třídy nejsou zabaleny s aplikací, protože zařízení, které podporuje dané volitelné balíčky, je již obsahuje ve svém běhovém prostředí.

U volitelných balíčků lze zjistit jejich přítomnost či nepřítomnost. Test pro zjištění existence jedinečné třídy volitelného balíčku (v tomto případě balíčku WMA) by mohl vypadat takto:

```
...
public static boolean isWMAPresent(){
    try {
         Class.forName(
              "javax.wireless.messaging.MessageConnection" );
         return true;
    }
    catch( Exception e ){
         return false;
    }
}
...
```

3.8. Wireless Messaging API (WMA)

Wireless Messaging API (WMA) je volitelný balíček, který doplňuje strukturu Generic Connection Framework (GCF) o funkcionalitou *radiogramu* (bezdrátové sdělení). Radiogram umožňuje asynchronní komunikaci podobně jako datagram. WMA je především učen pro zařízení podporující CLDC nebo CDC profil, jež umožňují odesílat a přijímat radiogramy (např. mobilní telefony). Také Java SE aplikace mohou využívat výhod balíčku WMA v případě, že je implementován volitelný balíček Generic Connection Framework pro Java SE (JSR 197).

3.8.1. Generic Connection Framework (GCF)

Generic Connection Framework (GCF) poskytuje širokou podporu typů připojení pro aplikační rozhraní platformy Java ME. Původně byl GCF navržen pro Java ME platformu a profil CLDC 1.0, protože API `java.net` a `java.io` známé z Java SE byly považovány

za příliš rozsáhlé a paměťově náročné pro použití v zařízeních s omezenými zdroji. Také množství volitelných balíčků, které obohacují původní specifikaci GCF o nové rozšíření, se od doby vzniku značně zvětšilo. GCF je založen na zcela novém způsobu abstrakce, díky čemuž dokáže používat různé mechanismy komunikace. Výsledkem této abstrakce je, že GCF neposkytuje přímo implementaci pro jednotlivé protokoly [23].

Velmi důležitou roli v GCF představuje *URI* (Uniform Resource Identifier) specifikovaný v RFC 2396. URI popisuje za použití hierarchické notace umístění a přístupovou metodu zdroje v prostředí Internetu. V GCF je URI použit tak, že identifikuje koncové body a typ připojení. URI se skládá ze tří částí: *schématu*, *adresy* a volitelně *seznamu parametrů*. Obecný zápis URI je následující:

```
<schéma>:<adresa>;<parametery>
```

Schéma identifikuje přístupovou metodu protokolu. V GCF URI popisuje typ použitého připojení (socket, http, file, datagram, atd.). Adresa identifikuje cílovou adresu (neboli cestu ke zdroji). Formát a interpretace závisí na schématu (např. www.vutbr.cz, soubor.txt atd.) Adresa může definovat volitelně parametry. Parametry identifikují v závislosti na požadavcích protokolu další informace pro sestavení spojení (např. přenosovou rychlost). Tab. 1 uvádí některé URI schémata a typy připojení GCF jež jsou definovány JCP.

Tab. 1 – schéma URI u GCF a definované typy připojení

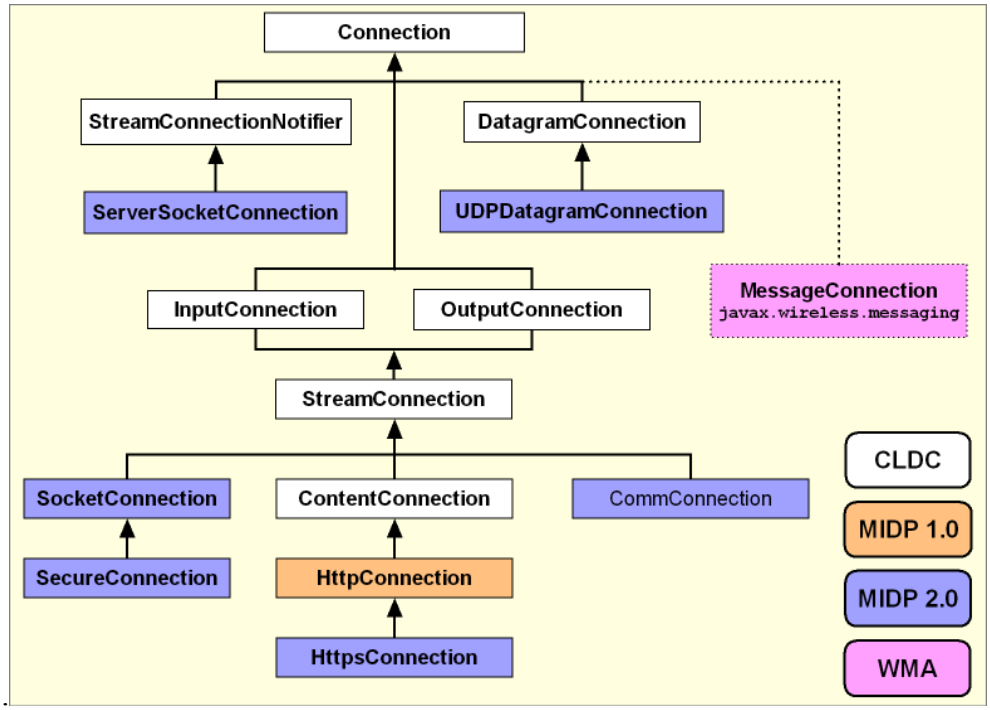
Schéma URI	Typ připojení	GCF typ připojení	Definovaný v…
`btl2cap`	Bluetooth	`L2CAPConnection`	Bluetooth API (JSR 82) - podpora volitelně
`datagram`	Datagramové	`DatagramConnection`	Všechny profily založené na CLDC a CDC, s GCF (JSR 197) podpora v Java SE - podpora volitelně
`file`	Přístup k souborům	`FileConnection`, `InputConnection`	PDA (JSR 75) - podpora volitelně
`http`	HyperText Transport Protocol	`HttpConnection`	MIDP 1.0 (JSR 37), MIDP 2.0 (JSR 118), Základní profil (JSR 46, JSR 219), Java SE (JSR 197) - podpora nutná
`https`	Zabezpečené HTTP	`HttpsConnection`	MIDP 2.0 (JSR 118) - podpora nutná
`comm`	Sériový I/O	`CommConnection`	MIDP 2.0 (JSR 118) - podpora volitelně
`sms`	Služba krátkých textových zpráv	`MessageConnection`	WMA (JSR 120, JSR 205) - podpora volitelně
`mms`	Služba multimediálních zpráv		
`cbs`	SMS vysílaná v rámci buňky sítě		
`socket`, `serversocket`	Socket	`SocketConnection`, `ServerSocketConnection`	MIDP 2.0 (JSR 118) - podpora volitelně
`datagram`	UDP Datagram	`UDPDatagramConnection`	MIDP 2.0 (JSR 118) - podpora volitelně

Příklady užití URI: http://www.ctimn.com:8080, socket://localhost:8080, file:c:/mujfile.txt (pouze Windows), file:/mujfile.txt (Unix), datagram://127.0.0.1:8099, comm:0;baudrate=9600.

GCF je definován v balíčku `javax.microdedition.io` a obsahuje: jednu třídu (`Connector`), jednu výjimku (`ConnectionNotFoundException`) a osm rozhraní.

Obr. 8 ukazuje hierarchii rozhraní připojení pro GCF definované CLDC 1.0, které rozšiřují rozhraní `Connection`. Dále pak rozšiřující rozhraní definované profily MIDP 1.0 a MIDP 2.0 a nakonec rozšiřující rozhraní definované volitelným balíčkem WMA.

Jak je vidět, rozhraní bázové konfigurace jsou doplňovány rozšiřujícími rozhraními profilů a volitelných balíčků. To umožňuje flexibilně získat nové funkcionality a pomocí vhodných kombinací vlastností využit schopností konkrétního zařízení.



Obr. 8 – hierarchie rozhraní připojení

3.8.2. Verze Wireless Messaging API

WMA v současné době existuje ve třech specifikacích. A to specifikace WMA 1.0, jejíž finální verze byla vydána v srpnu roku 2002 pod označením JSR 120. Implementace WMA 1.0 přinesla rozšíření GCF o možnost použití radiogramů, které mají obdobné vlastnosti jako datagramy používané pro UDP (User Datagram Protocol). Radiogramy poskytují základ otevřeného spojení, které je založeno na adresování pomocí řetězců (dle specifikace URI). Otevření tohoto spojení může být provedeno buď v klientském módu (odesílání), anebo serverovém módu (příjem). WMA 1.0 umožnilo příjem a posílání SMS (Short Message Service) a CBS (Cell Broadcast Service) zpráv v textovém, anebo binárním tvaru.

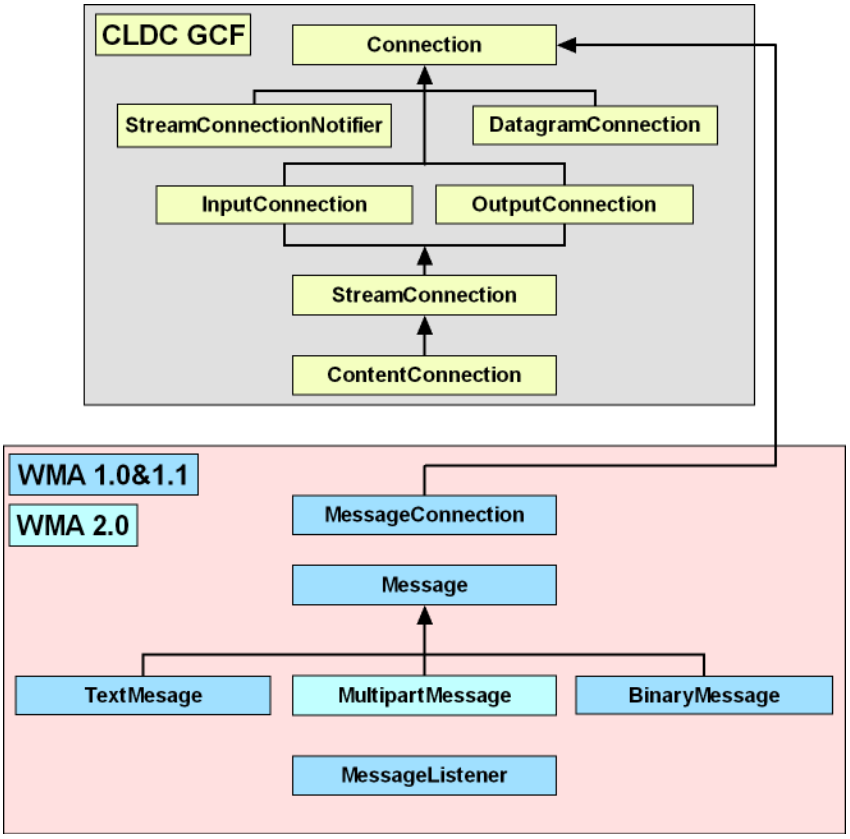
Finální specifikace WMA 1.1 byla vydána v dubnu roku 2003 pod označením JSR 120 Final Release 2. Tato upravovala původní JSR 120 zahrnutím změn, které vznikly na základě schválení architektury profilu MIDP 2.0. Změny přinesly novou verzi bezpečnostního rámce a Push mechanismy. Tato verze také umožňuje vývojářům snadněji emulovat odesílání a příjem SMS a CBS zpráv mezi rozmanitými emulátory zařízení založených na profilu MIDP 2.0 a mnoho dalšího (viz [20]).

Verze WMA 2.0 (JSR 205) je poslední verzí, jejíž specifikace byla dokončena v červnu 2004. Je rozšířením verze WMA 1.1 především o dlouhé SMS zprávy a zprávy složené z více částí a dále o multimediální zprávy (Multimedia Messaging Service – MMS) (viz [21]).

3.8.3. Struktura Wireless Messaging API

Způsob doručení radiogramu je závislý na tom, v jakém prostředí je služba použita (GSM, CDMA), případně závisí na použitém druhu služby (GSM SMS, GSM CBS, CMDA SMS nebo MMS).

Na Obr. 9 je znázorněna struktura WMA a její návaznost na GCF. Všechny komponenty WMA jsou obsaženy v jednom balíčku `javax.wireless.messaging`, který definuje všechny rozhraní potřebné pro odesílání a příjem radiogramů buď jako textová nebo jako binární data.



Obr. 9 – struktura Wireless Messaging API

WMA definuje rozšíření rozhraní GCF (balíčku `javax.microedition.io`). Tyto WMA rozšiřující třídy jsou nazývány `MessageConnection`. Stav počáteční instance `MesageConnection` závisí na tom, zda aplikace zprávy přijímá (tzv. serverový mód), anebo zprávy odesílá (tzv. klientský mód). Pro odesílání zpráv je použita třída `Connector`. Jestli chce být aplikace upozorněna na příchozí zprávu, musí mít nejdříve zaregistrovánu samu sebe jako posluchače (příjemce) zpráv. Aby toto dosáhla, implementuje rozhraní `MesageListener` z WMA a nechá MIDlet zaregistrovat sám sebe při vytvoření a připojí k specifickému portu, kdy také MIDlet nastaví sám sebe jako posluchače na tomto portu. Nyní je vytvořena instance `MessageConnection` a lze přijímat a odesílat radiogramy. WMA zprávy jsou vytvářeny voláním jedné z možných metod `MesageConnection`. WMA také definuje zapouzdření radiogramu, který je definován v rozhraní `Message`. Toto rozhraní definuje metody pro nastavení a získání informací v hlavičce zprávy (např. cílová adresa). Zpráva může být vytvořena za použití jednoho ze tří podrozhraní (`TextMessage`, `BinaryMessage`, `MultipartMessage`). WMA API definuje sadu rozhraní, které obsahují metody pro všechny požadované operace s radiogramy. Tyto obsahují registraci posluchače zpráv, vytvoření spojení, příjem a odesílání radiogramů atd.

3.9. WMA a podpora výrobců zařízení

Nyní se zaměřím na popis podpory WMA u jednotlivých výrobců zařízení. Nebudu se zabývat všemi druhy zařízení, jež umožňují práci s radiogramy, ale zaměřím se především na mobilní telefony. A to především na klasické mobilní telefony s uzavřeným operačním systémem, které podporují platformu Java ME. Protože mají tyto mobilní telefony uzavřený operační systém, vynikne u těchto přístrojů možnost získat nové funkce pomocí doinstalování aplikace fungující na Java ME platformě.

3.9.1. Zjištění podpory volitelného balíčku WMA na trhu

Nynější situaci u následujících výrobců mobilních telefonů: Nokia, Mototrola, Samsung, Sony Ericsson, BenQ-Siemens, LG (pozn. tyto výrobce jsem zvolil proto, že procentuelně obsazují většinu světového trhu s mobilními telefony) je takováto.

Podpora volitelného balíčku WMA u jednotlivých výrobců mobilních telefonů Nokia, Motorola, BenQ-Siemens, Sony Ericsson, Samsung, LG Electronics by se dala specifikovat následně. Obecně lze říci, že pokud mobilní telefon disponuje platformou Java ME a podporuje profil MIDP 2.0, pak je ve většině případů zahrnut volitelný balíček minimálně ve verzi WMA 1.1 (JSR 120). A to bez ohledu na to, zda je podporována konfigurace CLDC 1.0 nebo CLDC 1.1. V případě kombinace konfigurace CLDC 1.1 a profilu MIDP 2.0 nebo MIDP 2.1 je dnes (rok 2008) ve většině případů podporován volitelný balíček WMA 2.0 (JSR 205). Pokud mobilní telefon podporuje pouze profil MIDP 1.0 v kombinaci s konfigurací CLDC 1.0, pak v drtivé většině případů podpora volitelného balíčku WMA (JSR 120) zahrnuta není.

4. Návrh řešení

V textu uvedeném výše byla teoreticky rozebrána platforma Java ME s ohledem na zařízení s omezenými zdroji podporující konfiguraci CLDC a profil MIDP (s bližším zaměřením na MIDP 2.0). Byl také rozebrán způsob podpory volitelných balíčků, jejich umístění v architektuře platformy Java ME. Dále byla zaměřena pozornost na volitelný balíček Wireless Messaging API (ve všech jeho dostupných verzích).

Nyní bude přistoupeno k vlastnímu návrhu řešení, na základě požadavků zadání práce. V návrhu budou zohledněny skutečnosti, které byly zjištěny během teoretického rozboru.

4.1. Přehled stavu trhu

Současné realizace aplikací na platformě Java ME, které umožňují práci s SMS zprávami, zahrnují širokou škálu aplikací. Jsou mezi nimi aplikace, jež jsou realizovány formou jednoduchých demo MIDletů, které demonstrují možnosti platformy Java ME. Dále aplikace navržené speciálně za účelem práce s SMS případně MMS zprávami, které se snaží přinést nové funkce, případně odstraňují nedostatky výrobcem dodávaných nativních aplikací pro práci s SMS zprávami (typicky psaní SMS zpráv s diakritikou, jež zkracuje maximální délku SMS zprávy). Až k aplikacím, které využívají funkce pro práci s SMS zprávami, jako doplněk k hlavnímu zaměření aplikace (např. aplikace pro organizaci času). I přesto zde existuje prostor pro další aplikace, které budou podporovat práci s SMS zprávami a jejich využití na mnoho dalších způsobů.

4.2. Návrh řešení na základě zadání

Jedním z bodů zadání je, že by měl být proveden návrh a vytvoření aplikace na platformě Java ME, která bude využitelná pro příjem a odesílání SMS zpráv. Po dohodě s vedoucím práce a na základě studia problematiky bylo rozhodnuto, že bude vytvořeno několik aplikací. Hlavním jádrem bude MIDlet, který bude fungovat jako automatizovaný hlasovací SMS server. Tento MIDlet bude možno provozovat na mobilním zařízení podporující platformu Java ME a volitelný balíček WMA 2.0. Dále pak bude tento MIDlet umožňovat spojení s ovládací aplikací pomocí Internetu a ovládání MIDletu z této aplikace. Ovládací aplikace bude vytvořena za použití platformy Java SE. Tímto by měly být splněny body zadání.

V současné době existují mnohá řešení automatizovaných hlasovacích SMS serverů. Všechna tato řešení (při zkoumání situace se nepodařilo zjistit opak) vyžadují pro svůj běh osobní počítač a připojený mobilní telefon. Tento telefon je připojen k osobnímu počítači buďto pomocí kabelu, nebo pomocí Bluetooth. Tyto způsoby vzájemného spojení determinují, že nelze mobilní telefon od počítače příliš vzdálit. Princip těchto automatizovaných hlasovacích SMS serverů spočívá v tom, že mobilní telefon přijímá SMS zprávy, aplikace umístěná na osobním počítači k nim přistupuje, SMS zprávy si z mobilního telefonu stáhne a následně tyto SMS zprávy zpracovává. Výhodou těchto řešení je, že je lze relativně snadno realizovat (mnoho těchto programů je volně

šiřitelných). Nevýhodou je pak nutný nepřetržitý provoz osobního počítače, jenž je energeticky poměrně náročný.

4.2.1. Vlastní teoretický návrh

Z informací uvedených v kapitole 3.8.3 plyne, že aby bylo možno pomocí Java ME aplikace přijmout SMS zprávu, musí tato přijímaná SMS zpráva obsahovat číslo portu. MIDlet sloužící pro příjem SMS zpráv musí na tomto portu naslouchat a v případě, že dojde takováto SMS zpráva, je pomocí AMS (manažeru aplikací) upozorněn. Po upozornění je MIDletu SMS zpráva předána, načež může proběhnout její zpracování. V případě, že přijímaná SMS zpráva nemá určeno číslo portu (např. běžně zaslaná SMS zpráva z mobilního telefonu nebo SMS brány), anebo je číslo portu jiné, než na jakém MIDlet přijímající SMS zprávy naslouchá, je takováto SMS zpráva MIDletem ignorována. O její přijetí se postará nativní aplikace mobilního telefonu (dodávaná jako součást firmware).

Z výše uvedeného plyne, že SMS zprávy odeslané z nativních aplikací obsažených v mobilním telefonu není možno pomocí platformy Java ME přijmout, ani s nimi jakkoli pracovat (pozn. toto by umožňovaly pouze aplikace napsané pro otevřený operační systém jako je Symbian nebo Windows Mobile, ale v jiném jazyce než Java). Aby bylo možno pomocí platformy Java ME SMS zprávu přijmout, musí být SMS zprávě při odeslání dodáno číslo portu. Tohoto lze docílit s pomocí MIDletu, který pracuje v tzv. klientském módu (odesílá SMS zprávy) a používá k odesílání SMS zpráv následující URI schéma:

```
sms://<telefonní číslo >:<číslo portu>.
```

Na základě těchto zjištění bylo rozhodnuto, že bude vytvořen MIDlet (nazvaný *SMSkaHlas*), který bude využívat volitelného balíčku WMA 2.0 pro odesílání SMS zpráv (tzv. klientský módu) s určeným číslem portu. Číslo portu bylo podle doporučení specifikace určeno z rozsahu 16000 – 16999. Konkrétně byl zvolen port 16100.

Dále bylo přikročeno k návrhu funkcí MIDletu (nazvaný *SMSka*), který bude sloužit jako automatizovaný hlasovací SMS server. Při návrhu byly na MIDlet kladeny tyto požadavky: příjem (tzv. serverový mód) a odesílání (tzv. klientský mód) SMS zpráv, možnost perzistentního uložení přijatých dat (data jsou uchována i v případě ukončení aplikace nebo vypnutí mobilního zařízení).

Pro realizaci příjmu a odesílání SMS bylo rozhodnuto, že bude použito volitelného balíčku WMA 2.0. Tento balíček bude v MIDletu SMSka použit v obou módech (klientský i serverový).

Pro persistenci dat byl zvolen systém pro správu záznamů (Record Management System – RMS, více viz kapitola 4.2.3). Tento se jevil svou kapacitou jako dostatečný, a také jednoznačně nejuniverzálnější (na rozdíl od zvažovaného ukládání dat do souborového systému telefonu – všechna mobilní zařízení toto API nepodporují).

Dalším krokem byl návrh MIDletu (nazvaný *Synchronizace*) umožňujícího komunikaci s ovládacím rozhraním, které bude provozováno na PC a realizováno za použití platformy Java SE. Byla požadována možnost spojení MIDletu a aplikace, a také aby měl MIDlet Synchronizace přístup k perzistentním úložištím MIDletu SMSka.

Pro komunikaci mezi MIDletem a ovládacím rozhraním bylo z nabízených možností (spojení pomocí kabelu, pomocí Bluetooth, pomocí Internetu) zvoleno spojení pomocí Internetu. Důvodem byly největší možnosti mobility hlasovacího serveru, byť za cenu nákladů za přenesená data při vzájemné komunikaci. Konkrétně bylo do návrhu zvoleno spojení realizované pomoci socketů. Toto spojení zaručuje spolehlivý přenos (na bázi TCP/IP protokolu) a relativně snadnou implementaci do aplikací jak na platformě Java ME, tak platformě Java SE.

Pro perzistenci dat bude použit, stejně jako v MIDletu SMSka, systém pro správu záznamů (RMS). Aby nenastaly případné komplikace s přístupem k datovým úložištím, bylo rozhodnuto, že MIDlety SMSka a Synchronizace budou distribuovány jako sada MIDletů (jeden balík obsahující dva MIDlety).

Posledním krokem byl návrh ovládacího rozhraní (nazvané *SMSkaServer*). Toto ovládací rozhraní mělo být realizováno za pomocí platformy Java SE. Požadavky na toto ovládací rozhraní byly následující: možnost komunikace se sadou MIDletů sloužícím jako hlasovací SMS server a jich ovládání, možnost perzistentního uložení dat získaných z hlasovacího SMS serveru.

Jak již bylo zmíněno výše v návrhu MIDletů, byl pro vzájemné propojení aplikace a MIDletu Synchronizace zvolen Internet. Konkrétně spojení za pomoci socketů. To z důvodu relativně jednoduché implementace do aplikace, a také z důvodu spolehlivosti přenosu informací.

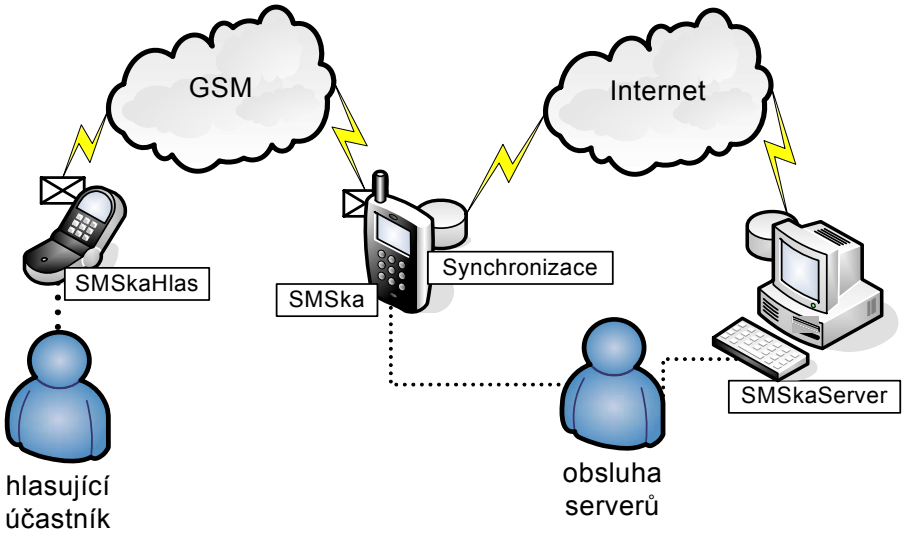
Jelikož bylo také potřeba uskutečnit ovládání hlasovacího SMS serveru, bylo rozhodnuto, že bude vytvořen jednoduchý aplikační protokol, který bude postaven na principu dotaz – odpověď. Aplikační protokol bude navrhnut takovým způsobem, že ovládací aplikace bude zasílat dotazy, MIDlet tyto dotazy zpracuje a následně zašle odpověď. Vzhledem k předpokládanému nízkému výpočetnímu výkonu hlasovacího SMS serveru (zařízení s omezenými zdroji), bude snaha protokol navrhnout tak, aby nebyl příliš výpočetně náročný.

Pro perzistenci uložení dat bylo na platformě Java SE bylo zvoleno ukládání do souboru. Jeden z důvodu tohoto rozhodnutí byl, aby s daty mohl případně pracovat i jiný program. Proto bylo rozhodnuto, že ukládaná data nebudou mít žádnou složitou vnitřní datovou strukturu, a také že ukládání do souboru bude prováděno v otevřené textové podobě (snaha o co největší čitelnost dat).

4.2.2. Vizualizace teoretického návrhu

Obr. 10 zobrazuje vizualizovaný teoretický návrh. Hlasujícím účastníkům by mělo být s pomocí MIDletu SMSkaHlas umožněno zasílat SMS zprávy v podobě hlasů skrze síť GSM (Global System for Mobile Communications). Tyto SMS zprávy by měl zpracovat

MIDlet SMSka, který plní funkci hlasovacího SMS serveru. MIDlet SMSka by měl přijaté SMS zprávy zpracovat, uložit a případně na ně reagovat odesláním SMS zprávy. MIDlet Synchronizace by měl umožnit propojení s aplikací SMSkaServer pomocí Internetu. Aplikace SMSkaServer by měla být schopna díky tomuto spojení ovládat vlastnosti MIDletů SMSka a Synchronizace, které jsou svázány jako sada MIDletů. Ovládání sady MIDletů a aplikace SMSkaServer provádí obsluha, která podle potřeby spouští MIDlety, případně pomocí aplikace SMSkaServer ovlivňuje chování sady MIDletů.



Obr. 10 – vizualizace vlastního teoretického návrhu

4.2.3. Datové úložiště a systém pro správu záznamů (RMS)

Systém pro správu záznamů (Record Management System – RMS) je určen pro dlouhodobé ukládání dat [30]. RMS je v podstatě model jednoduché databáze. RMS je dostupné již ve spacifikaci MIDP 1.0. V MIDP 2.0 byly jeho možnosti rozšířeny o další vlastnosti. Nejdůležitější z těchto vlastností je možnost více MIDletů sdílet jedno datové úložiště. Datové úložiště je pojmenovaná množina záznamů. Data uložená v datovém úložišti jsou zachována i po ukončení MIDletu, či případném vypnutí mobilního zařízení. Jednotlivá datová úložiště jsou identifikována pouze pomocí jména (max. 32 znaků dlouhého). Při vytváření datového úložiště lze určit, zda k němu bude mít možnost přistupovat i jiný MIDlet, než ten který jej vytvořil. Jednotlivé záznamy v datovém úložišti jsou identifikovány pomocí ID. ID záznamu neurčuje celkový počet záznamů v úložišti, neboť při smazání záznamu již nikdy nebude ID tohoto smazaného záznamu v daném úložišti použito. V případě, že s jedním datovým úložištěm pracuje více vláken, je na vývojáři, aby zajistil vzájemnou koordinaci přístupů těchto vláken (aby si vlákna data nepřepisovala, nedocházelo k nekonzistenci dat apod.). Pro práci s RMS se využívá třída `RecordStore`. Vzhledem k tomu, že v datovém úložišti nemusí ID jednotlivých záznamů následovat po sobě (z důvodu znovuneužití ID smazaného záznamu), slouží k procházení záznamů rozhraní `RecordEnumeration`. Toto rozhraní může využít ke svému rozšíření další dvě rozhraní `RecordFilter` (sloužící k filtrování záznamů) a `RecordComparator` (sloužící k třídění záznamů). Více se RMS věnuje literatura [30] a [31].

5. Realizace

Tato kapitola se zabývá realizací návrhu, jenž byl proveden v kapitole 4.2.1. Při realizaci byla snaha držet se vytýčených bodů návrhu. Tyto vytýčené body byly během průběhu realizace doplněny také o prvky, jež během návrhu nebyly zřejmé, ale které se v průběhu realizace ukázaly být přínosné pro celkový výsledek.

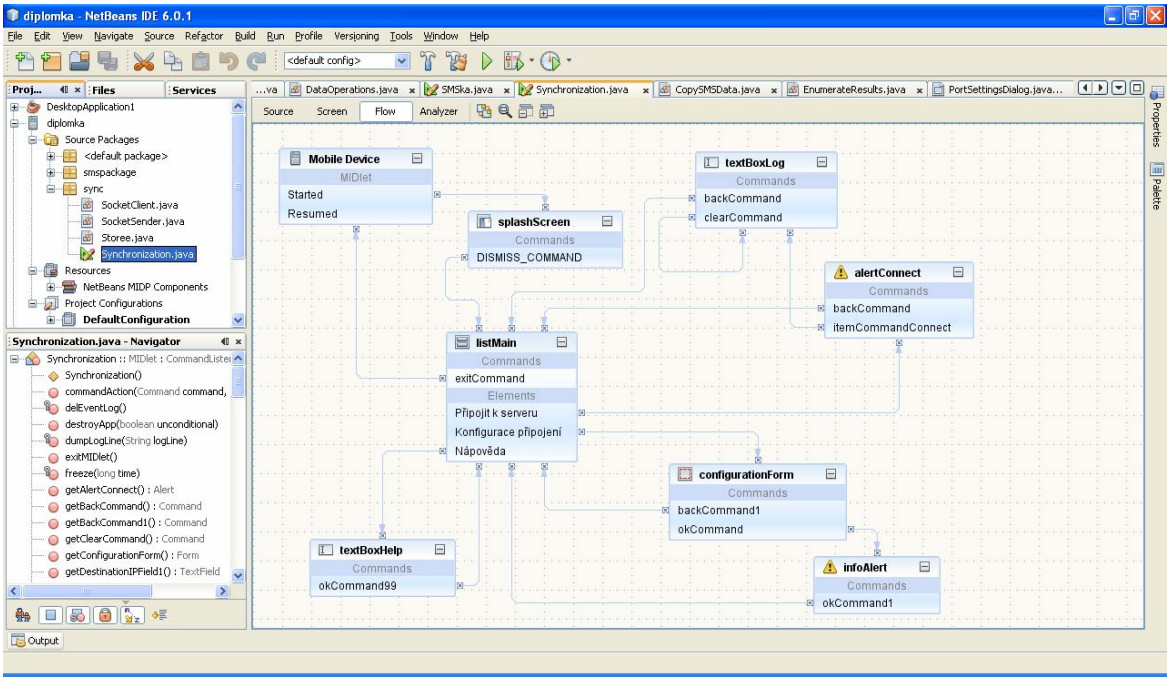
5.1. Realizace jednotlivých aplikací

Jak již bylo v teoretickém návrhu uvedeno, celé řešení se sestává ze 3 MIDletů určených pro mobilní zařízení. Těmito MIDlety jsou SMSkaHlas, SMSka a Synchronizace. Dále realizaci zahrnuje aplikace SMSkaServer, která je určena pro osobní počítače.

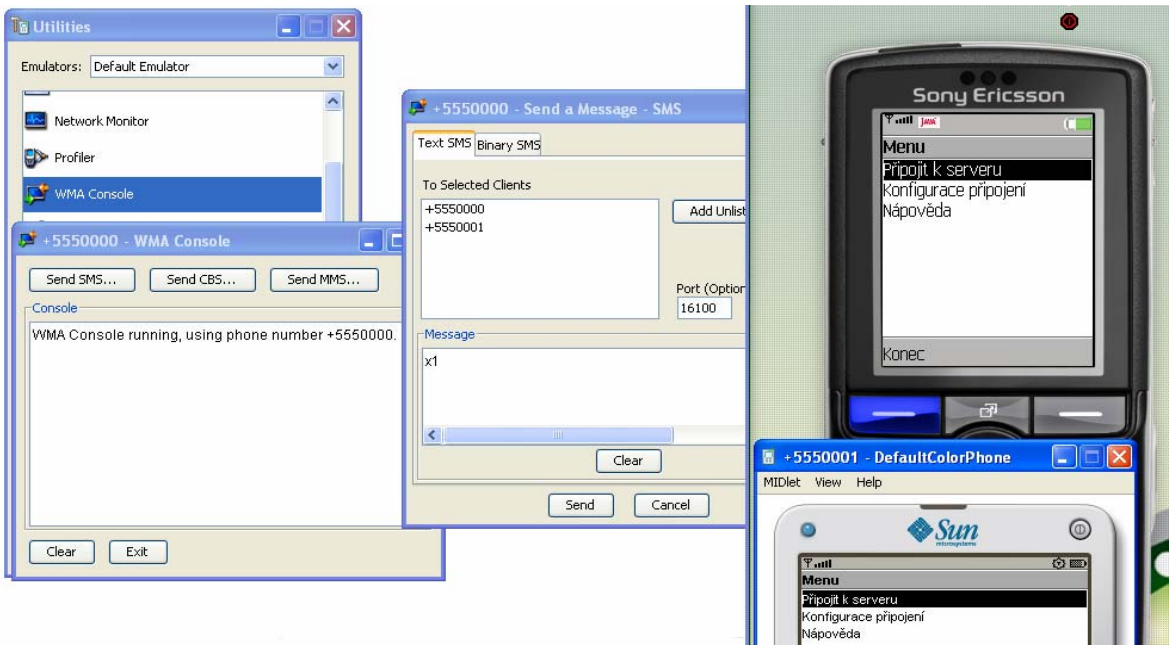
5.1.1. Nástroje použité pro realizaci

Začít vyvíjet programy pro Java SE platformu je poměrně snadné. Vývojáři stačí stáhnout a nainstalovat J2SE Software Development Kit (SDK), poté jakýkoli textový editor a může začít programovat kód. Ten si za pomocí SDK nástrojů přeloží. S vývojem pro platformu Java ME je to trochu složitější. K již nainstalovanému SDK si totiž vývojář musí ještě stáhnout a nainstalovat Sun Java Wireless Toolkit for CLDC. Kód si přeloží tentokrát za pomocí nástrojů Wireless Toolkitu. Ovšem vývoj takovýmto způsobem je poněkud zdlouhavý a nepohodlný. Drtivá většina dnešních vývojářů aplikací používá integrovaná vývojová prostředí (tzv. IDE – integrated development environment). Integrované vývojové prostředí je aplikace, která vývojáři poskytuje podporu mnoha činností při tvorbě aplikací. Základními prvky vývojového prostředí jsou např. editor zdrojového textu, překladač či interface na externí překladač, emulátory, frameworky atd. Vývojová prostředí poskytují také mnoho dalších užitečných funkcí, jako je zvýrazňování syntaxe, podporují možnosti ladění, nápovědu, využívají možností myši a mnoho dalších pro vývoj neocenitelných funkcí.

V mém případě jsem zvolil vývojové prostředí NetBeans IDE 6.1 s podporou Mobility pack pro vývoj aplikací pro CLDC zařízení. Toto vývojové prostředí poskytuje pro vývoj aplikací platformy Java ME tzv. Visual Mobile Designer verze 2, který značně zjednodušuje a urychluje vizuální návrh aplikace (zvláště v případě použití vysokoúrovňového API). Také je poměrně snadné do tohoto prostředí integrovat emulátory výrobců elektronických zařízení podporujících platformu Java ME (jako jsou emulátory mobilních telefonů apod.). Toto vývojové prostředí s rozšířením Mobility pack také podporuje a umožňuje využít utility a nástroje, které jsou obsaženy v Sun Java Wireless Toolkitu. Pro vývoj aplikací podle zadání byla asi nejpodstatnější podpora WMA Console, která umožňuje zasílat v rámci emulátorů SMS, CBS a MMS zprávy, a tak otestovat funkčnost aplikace (bez jakýchkoli nákladů) ještě před laděním na skutečném mobilním zařízení.



Obr. 11 – vývojové prostředí NetBeans IDE 6.1



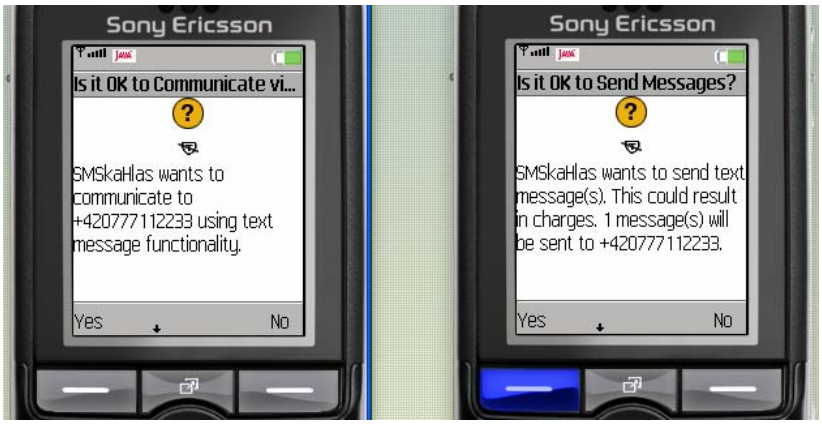
Obr. 12 – ukázka WMA Console a dvou emulátorů v NetBeans IDE

Během tvorby jednotlivých MIDletů a aplikace jsem těchto utilit, emulátorů a možnosti krokování často využil. Díky tomuto bylo značně zkráceno řešení chyb a vývoj postupoval jednoznačně rychleji.

5.1.2. Realizace MIDletu SMSkaHlas

V MIDletu *SMSkaHlas*, jako ostatně ve všech realizovaných MIDletech, byla snaha o oddělení kódu starajícího se o grafické zobrazování prvků a interakci s uživatelem a vlastního výkonného kódu, který má u tohoto MIDletu na starost odesílání SMS zpráv. Toto oddělení je realizováno pomocí tříd. Byly vytvořeny dvě třídy. Třída *SMSkaHlas* mající na starost především zobrazování a interakci s uživatelem a třída *SMSSending*, která obstarává odesílání SMS zpráv.

Nejprve jsem se pokusil vytvořit MIDlet tak, že jsem metody z třídy SMSSending pouze volal do třídy SMSkaHlas. Tato architektura se ovšem záhy ukázala být nevhodná. V každém MIDletu se totiž o zobrazování a především interakci s uživatelem stará vlákno implementující rozhraní `CommandListener`. Tomuto vláknu se také říká hlavní nebo systémové vlákno. V mém případě třída SMSkaHlas představuje systémové vlákno. Třída SMSSending využívá metod nabízených volitelným balíčkem WMA. Tento balíček spadá v bezpečnostním modelu platformy Java ME mezi kontrolovaná rozhraní. Důsledkem tohoto je, že v případě kdy se rozhraní `MessageConnection` balíčku WMA pokusí zavolat metodu `open` nebo metodu `send`, je nutno s uživatelem konzultovat přístupová práva tohoto rozhraní. Tato konzultace je realizována pomocí vyvolání obrazovky (viz Obr. 13), ve které je položen dotaz a kde jsou také uvedeny možné odpovědi.



Obr. 13 – dotaz na použití kontrolovaných rozhraní WMA balíčku při otevření rozhraní a pokusu odeslat SMS zprávu

Zobrazení této obrazovky a čekání na potvrzení volby způsobilo v systémovém vláknu uváznutí (tzv. deadlock). Řešení tohoto problému bylo relativně jednoduché. Díky tomu, že byl výkonný kód starající se o odesílání SMS zpráv ve vlastní třídě, stačilo tuto třídu upravit tak, aby běžela v samostatném vlákně. Takto její vykonávání a akce v důsledku jejího vykonávání nastanuvší, nemohou zablokovat systémové vlákno. Tento způsob řešení konstrukce MIDletu byl použit i u dalších vytvořených MIDletů.

Zjednodušený diagram průběhu MIDletu SMSkaHlas je uveden na Obr. 14. Tento diagram průběhu zobrazuje hlavní činnosti a rozhodování v toku MIDletu. Po startu MIDletu je nejdříve zobrazena obrazovka obsahující varování, které se týká informací o povolování práv kontrolovaných rozhraní. Poté následuje zobrazení hlavního formuláře. V

hlavním formuláři je možno zadat telefonní číslo příjemce a hlasovací text (znázorněno tokem *editace*). Hlavnímu formuláři jsou přiřazeny dva příkazy. Je to příkaz *Konec*, který ukončí MIDlet a příkaz *Poslat*. Po volbě příkazu Poslat jsou údaje z formuláře předány nové instanci třídy `SMSSending`, jež běží v samostatném vlákně.

Tok běhu vlákna závisí na tom, zda se MIDlet pokouší o odeslání SMS zprávy poprvé, anebo zda už byl proveden pokus o odeslání SMS zprávy. V případě, že se MIDlet pokouší o odeslání SMS zprávy poprvé, pak si vykonávání příkazu

```
smsConn = (MessageConnection)Connector.open(destinationAddress);
```

vynutí konzultaci přístupových práv s uživatelem. Tento příkaz slouží k vytvoření spojení nutnému k odesílání SMS zprávy. Rozhraní `MessageConnection` využívá metody `open` třídy `Connector`, kterému je předán parametr `destinationAddress` (jenž má podobu sms://<telefonní číslo>:<číslo portu>).

Rozhraní `MessageConnection` patří mezi kontrolované rozhraní, jehož práva musí být konzultovány s uživatelem. V důsledku toho je zobrazen dotaz na povolení práv (levá obrazovka z Obr. 13). Toto povolení nebo případné zakázání příjmu SMS zpráv, i když je zde WMA balíček užit pouze k odesílání SMS zpráv (tzv. klientský mód), si MIDlet pamatuje po celou dobu existence MIDletu v mobilním zařízení (typ trvalé povolení, viz kapitola 3.6.9). A to až do doby, dokud není MIDlet ze zařízení odinstalován. V případě, že MIDletu není uživatelem uděleno právo pro příjem SMS zpráv, bezpečnostní politika MIDP profilu neumožní MIDletu již nikdy odeslat SMS zprávu a průběh vykonávání MIDletu přejde do části, která ošetřuje nastanuvší bezpečnostní výjimku (`SecurityException`). Na displeji je zobrazena informace pro uživatele (ohledně nemožnosti odeslat SMS zprávu).

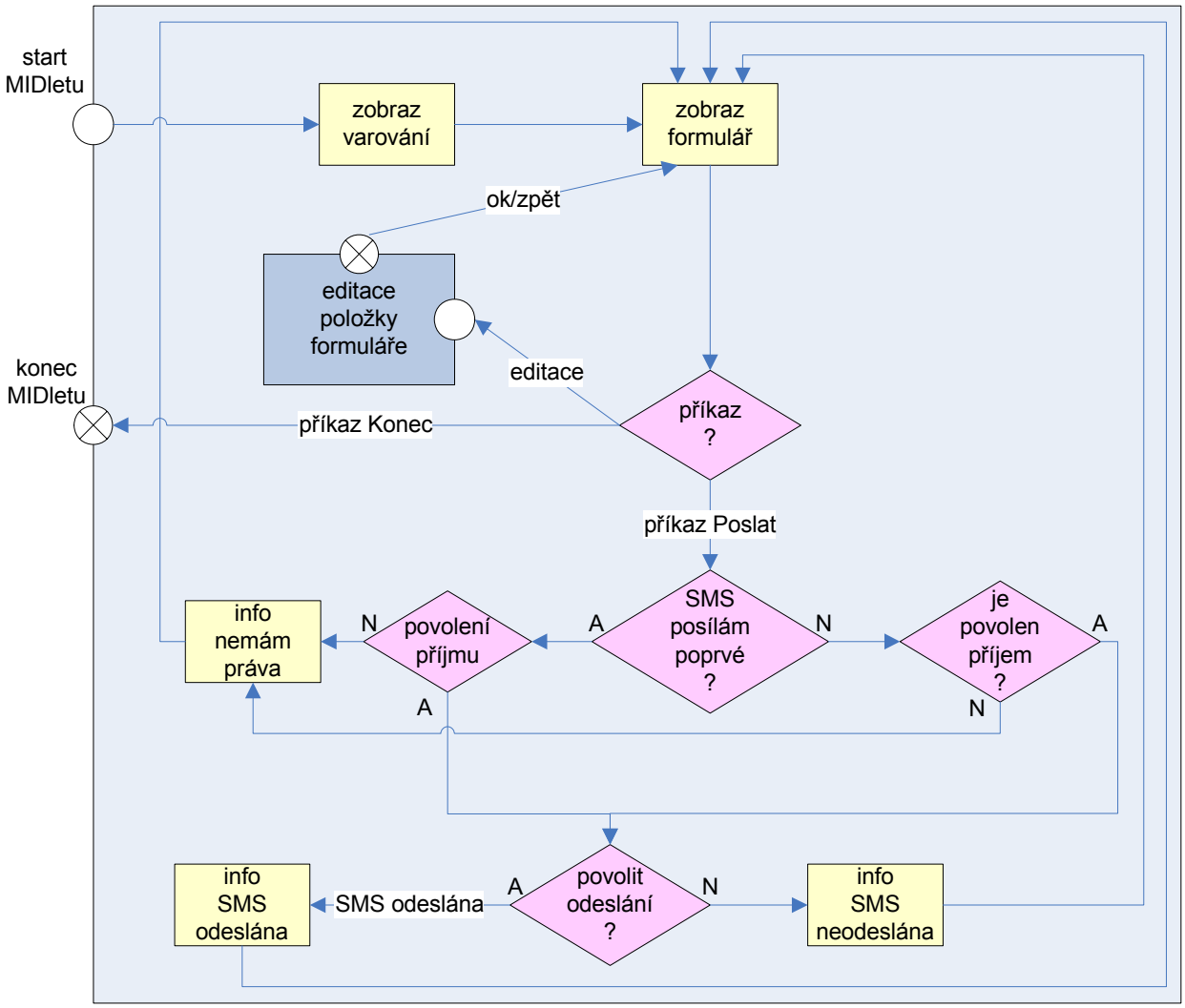
V případě, že je příjem SMS zpráv MIDletu povolen, vykovávání kódu pokračuje dále a to až do doby, kdy má být vykonán příkaz

```
smsConn.send(txtMessage);
```

Tento příkaz je v podstatě zodpovědný za odeslání vytvořené SMS zprávy (`txtMessage` jenž je parametrem metody). Metoda `send` spadá opět mezi kontrolované rozhraní balíčku WMA, jehož práva musí být konzultovány s uživatelem. Proto je položen uživateli dotaz v souvislosti s odesláním SMS zprávy (pravá obrazovka z Obr. 13).

Povolení nebo případné zakázání odeslání SMS zprávy si MIDlet nepamatuje (typ jednorázové povolení, viz kapitola 3.6.9). Proto je tento dotaz uživateli pokládán při každém pokusu o odeslání SMS zprávy. Zákaz odeslání způsobí, že průběh vykonávání MIDletu přejde do části, kde je ošetřena nastanuvší bezpečnostní výjimka (`SecurityException`), a že MIDlet neodešle SMS zprávu. Následně jsou zobrazeny informace o neodeslání SMS zprávy. V případě, že uživatel udělí povolení odeslání SMS zprávy, MIDlet odešle SMS zprávu a zobrazí informace o odeslané SMS zprávě.

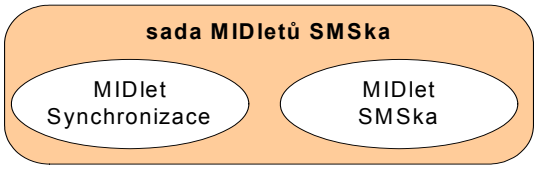
Jde-li u MIDletu o opakovaný pokus o odeslání SMS zprávy, není již MIDlet dotazován na povolení příjmu SMS zpráv, neboť si rozhodnutí na tento dotaz pamatuje z předešlého pokusu o odeslání.



Obr. 14 – zjednodušený diagram průběhu MIDletu SMSkaHlas

5.1.3. Realizace MIDletu SMSka

Sada MIDletů SMSka (sada je „balík“ obsahující více MIDletů, viz Obr. 15), zahrnuje MIDlet nazvaný *SMSka* a MIDlet nazvaný *Synchronizace*. Jak již bylo uvedeno výše v návrhu, důvodem zahrnutí těchto dvou MIDletů do jedné sady MIDletů, byla práce těchto MIDletů se stejnými datovými úložišti.



Obr. 15 – grafické znázornění sady MIDletů

MIDlet SMSka slouží jako automatizovaný hlasovací SMS server. Pro tento MIDlet byly vytvořeny třídy *SMSka*, která má na starost zobrazení a interakci s uživatelem, třída *SMSSending*, která zajišťuje odesílání SMS zpráv, třída *SMSReceiver,* která obstarává příjem a zpracování SMS zpráv a třída *Store,* která poskytuje ostatním třídám metody pro práci s datovými úložišti. Příjem a odesílání SMS zpráv je implementován za pomoci volitelného balíčku WMA.

Jádro třídy SMSSending je velice podobné jádru stejnojmenné třídy z MIDletu SMSkaHlas. Také postup povolování přístupových práv je u této třídy velice obdobný. Třída SMSReceiver využívá pro příjem SMS zpráv kontrolované rozhraní z balíčku WMA. Roli systémového vlákna v tomto MIDletu má třída SMSka. Třídy SMSSending a SMSReceiver jsou implementovány tak, že každá instance těchto tříd je spouštěna jako další vlákno.

Průběh MIDletu SMSka (viz Obr. 16) závisí na tom, zda byl MIDlet na zařízení již někdy spuštěn, anebo je spuštěn poprvé. Po spuštění MIDletu je během inicializace vytvořena instance třídy SMSSending (která implementuje rozhraní `MessageListener`) a je zavolána metoda `openSMSReceiver`. Metoda obsahuje příkaz

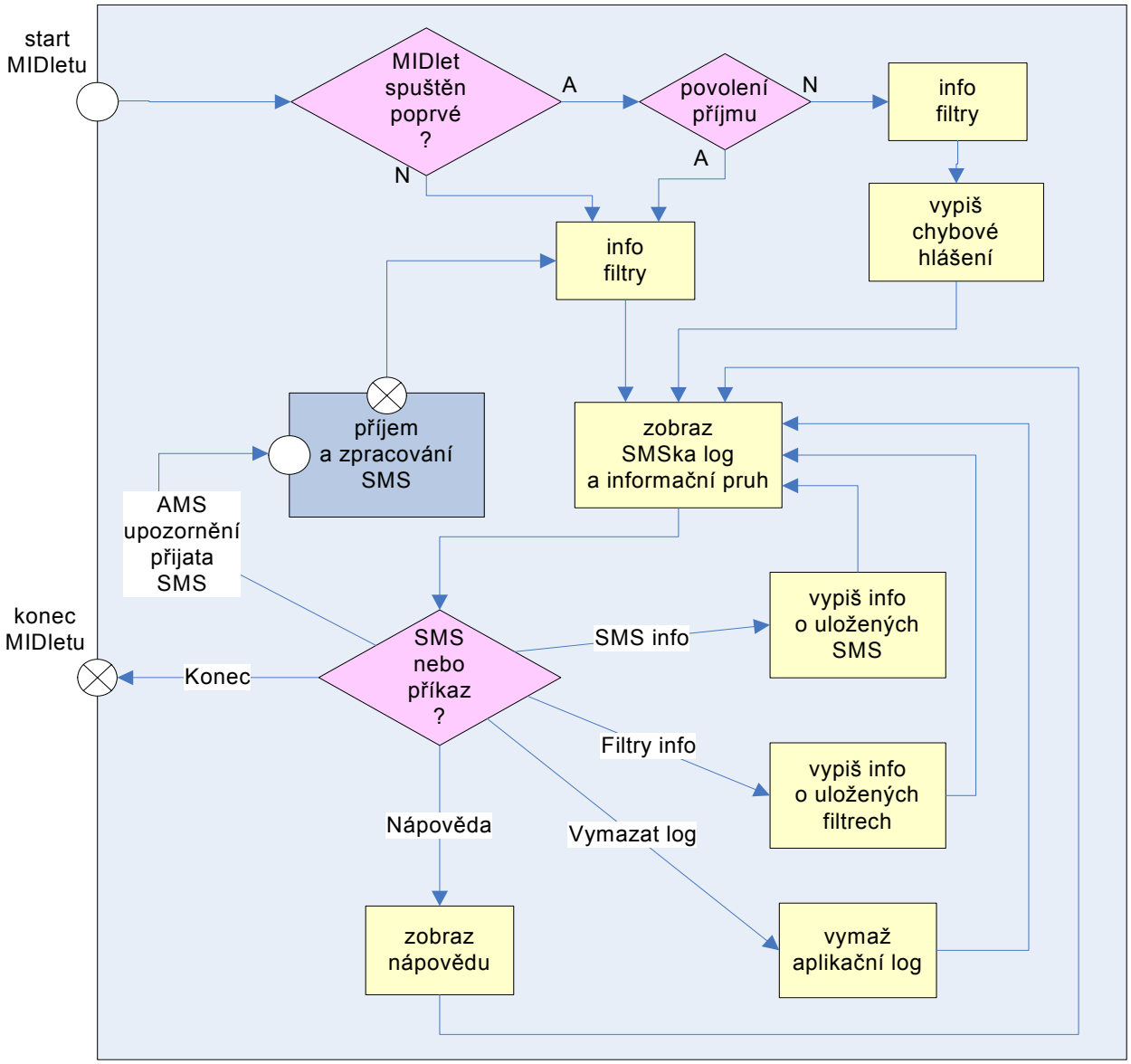
```
smsConn = (MessageConnection) Connector.open("sms://:"+smsPort);
```

Tento příkaz slouží k otevření spojení pro příjem SMS zpráv (tzv. serverový mód) na definovaném portu. Hodnota portu je získána z deskriptoru MIDletu. V případě, že je MIDlet na zařízení spuštěn poprvé, dojde při volání metody `open` třídy `Connector` k zobrazení obrazovky pro konzultaci přístupových práv s uživatelem (levá obrazovka z Obr. 13). Toto je důsledkem užití rozhraní `MessageConnection`, které patří mezi kontrolované rozhraní. Povolení nebo případné zakázání příjmu SMS zpráv si MIDlet pamatuje po celou dobu existence MIDletu v mobilním zařízení (typ trvalé povolení, viz kapitola 3.6.9). A to až do doby, dokud není MIDlet ze zařízení odinstalován. V případě, že MIDletu není uživatelem uděleno právo pro příjem SMS zpráv, bezpečnostní politika MIDP profilu MIDletu již nikdy neumožní MIDletu přijímat SMS zprávy. To má za následek, že MIDlet neplní svou hlavní funkci, kterou je příjem, zpracování a případné odesílání SMS zpráv. Na displeji je poté zobrazena obrazovka s informacemi o inicializaci filtrovacích pravidel. Následně je zobrazen hlavní obrazovka MIDletu, kde je informace o chybě aplikace v důsledku nepovolení práv pro příjem SMS zpráv. Jediným možným řešením této situace je odinstalace MIDletu, jeho opětovné nainstalování, spuštění a povolení příjmu SMS zpráv. Nejedná-li se o první spuštění MIDletu, pak je během inicializace MIDletu vytvořena instance třídy SMSSending s právy, které ji byly uděleny během prvního spuštění MIDletu.

Hlavní obrazovka MIDletu má podobu logu, ve kterém je nahoře zobrazen *informační pruh*. V informačním pruhu jsou zobrazeny informace o počtu SMS zpráv které byly přijaty, zpracovány a odeslány od startu MIDletu. Na hlavní obrazovku jsou do logu vypisovány informační zprávy.

Hlavní obrazovce je přiřazeno pět příkazů. Příkaz *Konec* ukončí MIDlet. Příkaz *Nápověda* zobrazí novou obrazovku s jednoduchou nápovědou. Příkaz *Vymazat log* vymaže všechny zobrazené informace z hlavní obrazovky. Příkaz *Filtry info* zobrazí

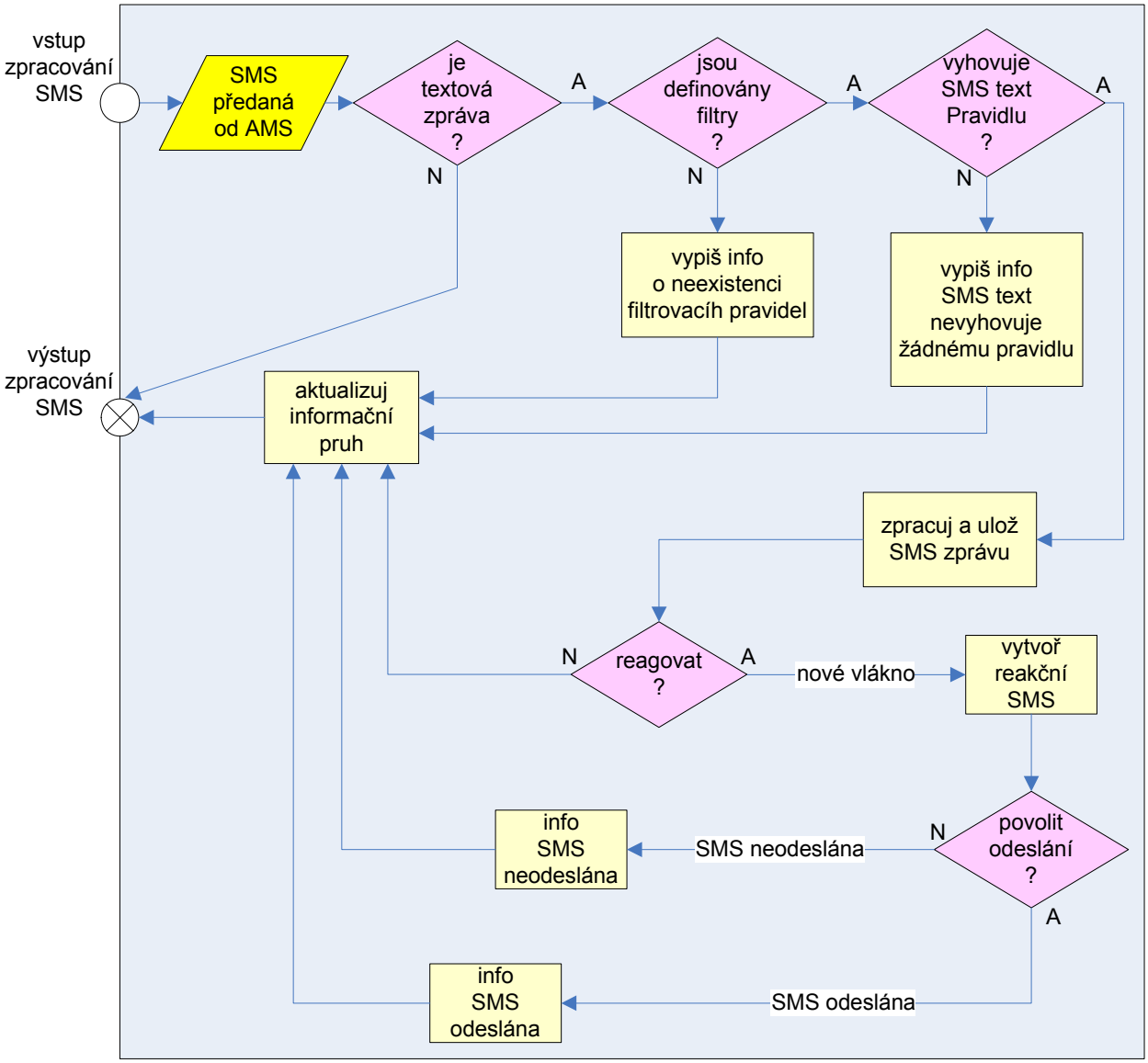
informace o filtrovacích pravidlech (počet filtrovacích pravidel, dostupnou a využitou kapacitu paměti uloženými filtrovacími pravidly). Příkaz *SMS info* zobrazí informace o uložených SMS záznamech (počet záznamů, dostupnou a využitou kapacitu paměti uloženými záznamy SMS).



Obr. 16 – zjednodušený diagram průběhu MIDletu SMSka

Asi nejdůležitější funkcí MIDletu je příjem a zpracování SMS zpráv. Zjednodušený diagram průběhu této činnosti je znázorněn na Obr. 17. Je-li přijata SMS zpráva která má určeno číslo portu, pak manažer aplikací (AMS) zkontroluje, zda je toto číslo shodné s číslem portu na kterém naslouchá MIDlet SMSka. Pokud se čísla portů shodují, je přijatá SMS zpráva předána MIDletu a současně je manažerem aplikací zavolána metoda `notifyIncomingMessage`. Tato metoda během svého vykonávání spustí běh vlákna

třídy SMSReceiver. Nejdříve je zjištěno, zda se jedná o textovou nebo binární SMS zprávu. Přijaté binární SMS zprávy jsou ignorovány a MIDlet je nijak nezpracovává. V případě přijetí textové SMS zprávy, je nejdříve zjišťováno zda jsou definována filtrovací pravidla, podle kterých jsou SMS zprávy zpracovávány. Pokud tato pravidla nejsou definována, je aktualizován informační pruh a v hlavní obrazovce je vypsán text, který nabádá k synchronizaci filtrovacích pravidel s aplikací SMSkaServer.



Obr. 17 – zjednodušený diagram průběhu příjmu a zpracování SMS

V případě, že jsou definována filtrovací pravidla, je zjišťováno, zda text příchozí SMS zprávy vyhovuje některému z těchto pravidel. Když nevyhovuje, je aktualizován informační pruh a v hlavní obrazovce je vypsána informace, že text přijaté SMS zprávy nevyhovuje žádnému pravidlu.

Vyhovuje-li text přijaté SMS zprávy některému z filtrovacích pravidel, pak je tato SMS zpráva zpracována a uložena. Určuje-li filtrovací pravidlo, že na SMS zprávu má být reagováno, je v rámci zpracování vytvořená nová instance třídy SMSSending, která běží ve vlastním vlákně. Třída SMSsending využívá metodu `send`, která spadá mezi kontrolované rozhraní balíčku WMA. Práva tohoto rozhraní musí být konzultovány s uživatelem. Proto je položen uživateli dotaz v souvislosti s odesláním SMS zprávy (pravá obrazovka z Obr. 13). Tento dotaz je zobrazen na displeji až do doby, než na něj uživatel reaguje. Povolení nebo případné zakázání odeslání SMS zprávy si MIDlet nepamatuje (typ jednorázové povolení, viz kapitola 3.6.9). Proto je při každém pokusu MIDletu o odeslání SMS zprávy položen tento dotaz znovu. Jednotlivé dotazy jsou řazeny za sebe do fronty a čekají na rozhodnutí uživatele. Zákaz odeslání způsobí, že MIDlet neodešle SMS zprávu. Následně je na hlavní obrazovce vypsána informace o neodeslání SMS zprávy a je aktualizován informační pruh. V případě, že uživatel udělí povolení odeslání SMS zprávy, MIDlet odešle SMS zprávu a na hlavní obrazovce je vypsána informace o odeslání SMS zprávy. Současně je aktualizován informační pruh.

Díky tomu, že jsou funkce starající se o zpracování přijatých SMS zpráv umístěny v samostatném vlákně, nemá trvající zobrazení dotazu na displeji, v souvislosti s odesláním SMS zprávy, žádný vliv na jejich běh. I přesto zobrazení dotazu na displeji, MIDlet dále přijímá a zpracovává SMS zprávy.

5.1.4. Realizace MIDletu Synchronizace

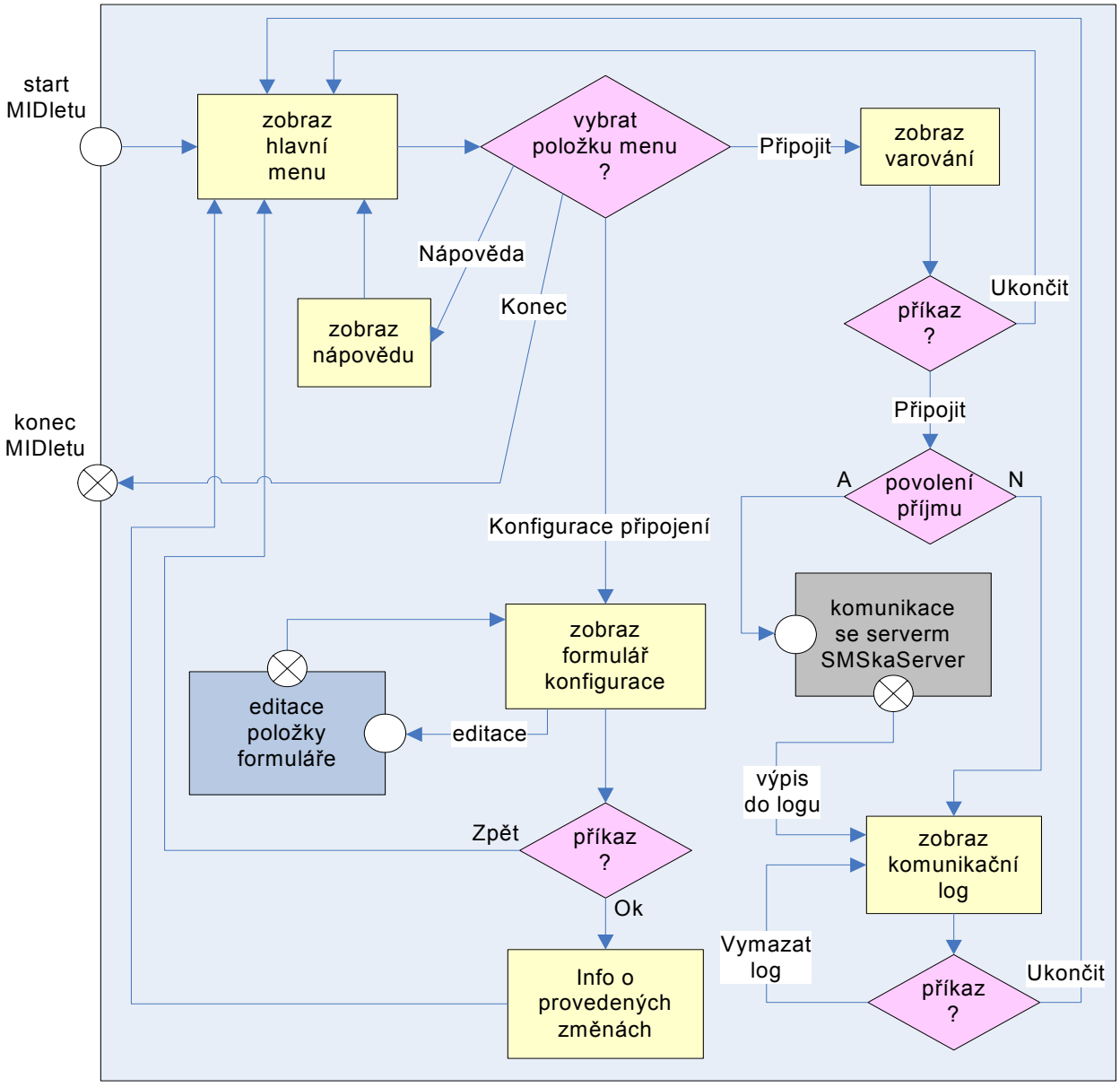
MIDlet Synchronizace je součástí sady MIDletů SMSka. MIDlet *Synchronizace* slouží ke komunikaci s aplikací *SMSkaServer*. Pomocí této komunikace jsou přenášeny informace a data uložených SMS záznamů směrem k aplikaci SMSkaServer a směrem k MIDletu jsou z této aplikace přenášeny filtrovací záznamy. Tento MIDlet se skládá ze čtyř tříd. Je to třída *Synchronization*, která obstarává zobrazení a interakci s uživatelem, třída *SocketClient*, která zajišťuje spojení s aplikací SMSkaServer skrze Intrernet, třída *SocketSender*, která má na starost odeslání dat směrem k aplikaci SMSkaServer a třída *Storee*, jenž poskytuje ostatním třídám metody pro práci s datovými úložišti.

Roli systémového vlákna v tomto MIDletu má třída Synchronization. Třídy SocketClient a SocketSender jsou implementovány tak, že každá instance těchto tříd je spouštěna jako další vlákno.

Diagram průběhu MIDletu Synchronizace je naznačen na Obr. 18. Po startu MIDletu je zobrazeno hlavní menu MIDletu. Toto menu obsahuje tři položky (možnosti), ze kterých lze zvolit. Je to položka *Připojit k serveru*, položka *Konfigurace připojení* a položka *Nápověda*. Hlavní menu MIDletu má také přiřazeno příkaz *Konec*, který způsobí ukončení MIDletu.

Po volbě položky *Nápověda* je zobrazena nová obrazovka s jednoduchou nápovědou. Po zvolení položky *Konfigurace připojení* je zobrazen formulář obsahující položky *Port* a položku *IP adresa serveru*, jejíž hodnota je rozdělena na jednotlivé bajty. Jednotlivé položky lze editovat. Formulář má přiřazeny dva příkazy. Je to příkaz *Ok* a příkaz *Zpět*.

Příkaz *Zpět* provede návrat do hlavního menu, aniž by byly případné provedené změny jednotlivých položek uloženy. Příkaz *Ok* zobrazí informační displej. Pokud se číslo zadaného portu nachází v rozmezí 1000 až 65534 a současně čísla jednotlivých bajtů IP adresy leží v rozsahu 0 až 255, je na displeji zobrazeno potvrzení uložení změn. V opačném případě se na displeji zobrazí varování, že změněná položka, která nevyhovuje rozsahu, nebyla uložena.



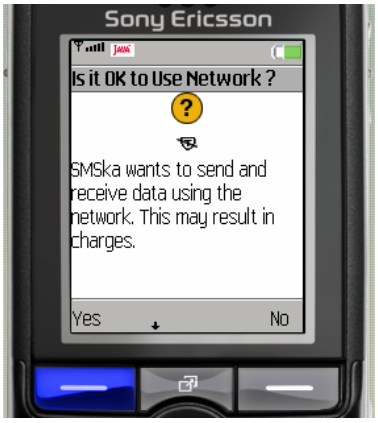
Obr. 18 – zjednodušený diagram průběhu MIDletu Synchronizace

Po zvolení položky *Připojit k serveru* je zobrazena obrazovka s varováním, která má přiřazeny příkazy *Připojit* a *Ukončit*. Příkaz *Ukončit* provede návrat do hlavního menu MIDletu. Příkaz *Připojit* způsobí, že je vytvořena nová instance třídy `SocketClient`.

Této instanci jsou předány nastavené parametry z formuláře *Konfigurace připojení* a je spuštěn běh této instance třídy ve vlákně. Instance třídy začne navazovat spojení se severem SMSkaServer a to pomocí příkazu:

```
sc=(SocketConnection)Connector.open
                              ("socket://"+destinationIP+":"+port);
```

Tento příkaz slouží k vytvoření spojení se serverem. Rozhraní `SocketConnection` využívá metody `open` třídy `Connector`, kterému je předán parametr, jenž má podobu `socket://"+destinationIP+":"+port`. Kde `destinationIP` určuje IP adresu aplikace SMSkaServer a `port` číslo portu na kterém tato aplikace naslouchá. Rozhraní `SocketConnection` patří mezi kontrolovaná rozhraní, jehož práva musí být konzultována s uživatelem. V důsledku toho je při vykonávání metody `open` zobrazen dotaz na povolení práv (viz Obr. 19), tedy přístupu na Internet.



Obr. 19 – dotaz na použití kontrolovaného rozhraní SocketConnection při otevření

Povolení nebo případné zakázání přístupu na Internet, si MIDlet pamatuje po dobu spuštění MIDletu (typ povolení pro sezení, viz kapitola 3.6.9). Proto je tento dotaz uživateli pokládán pouze při prvním pokusu o přístup na Internet po spuštění MIDletu. Zákaz odeslání dat způsobí, že průběh vykonávání MIDletu přejde do části, ve které je ošetřena nastanuvší bezpečnostní výjimka (`SecurityException`), a že se MIDlet nepřipojí na Internet. Následně je zobrazen *komunikační log* s informacemi o nepovolení práv pro přístup na Internet a postupu jak dále pokračovat. V případě, že uživatel udělí povolení k přístupu na Internet, MIDlet se pokusí o spojení s aplikací SMSkaServer. Je zobrazen komunikační log a výsledky komunikace mezi MIDletem a SMSkaServerem jsou vypisovány do tohoto komunikačního logu. Komunikační log má přiřazeny příkazy *Vymazat log*, který vymaže informace zobrazené v logu a příkaz *Ukončit*, který provede návrat do hlavního menu MIDletu a ukončí spojení se serverem, pokud je navázáno.

5.1.5. Aplikační protokol komunikace

Komunikace MIDletu Synchronizace a aplikace SMSkaServer je vedena podle schématu dotaz–odpověď a je řízena navrženým aplikačním protokolem. Aplikace SMSkaServer (dále v kapitole jen server) se dotazuje MIDletu Synchronizace (dále v kapitole jen klient). Klient přijaté dotazy zpracuje a serveru zašle odpověď.

Server zasílá klientovi tyto dotazy (příkazy):

- SMSKA_SRV – identifikace pro klienta,
- WRITE_FILTER – zápis jednoho filtrovacího záznamu do klienta,
- GET_NUM_QUAD – zjištění počtu uložených SMS zpráv v klientovi,
- FILTER_SIZE – zjištění počtu uložených filtrovacího pravidel v klientovi,
- READ_QUAD – stažení všech v klientovi uložených SMS zpráv,
- RENEW_DUO – smazání všech položek v úložišti filtrovacích pravidel,
- RENEW_QUAD – smazání všech položek v úložišti uložených SMS zpráv.

Klient zasílá serveru tyto odpovědi (příkazy):

- SMSKA – identifikace pro server,
- FILTER_NOTIFY – výsledek zápisu filtrů,
- NUM_QUAD – počet uložených SMS zpráv v klientovi,
- ONE_QUAD – odeslání jednoho uloženého SMS záznamu,
- DUO_ACK – potvrzení smazání všech položek úložiště s filtrovacími pravidly,
- QUAD_ACK – potvrzení smazání všech položek úložiště s uloženými SMS zprávami.

Protokol je složen ze dvou částí. *Identifikační* části, kdy se klient a server vzájemně autentizují a *příkazové* části, ve které si server a klient vzájemně vyměňují dotazy a odpovědi (probíhá výměna dat).

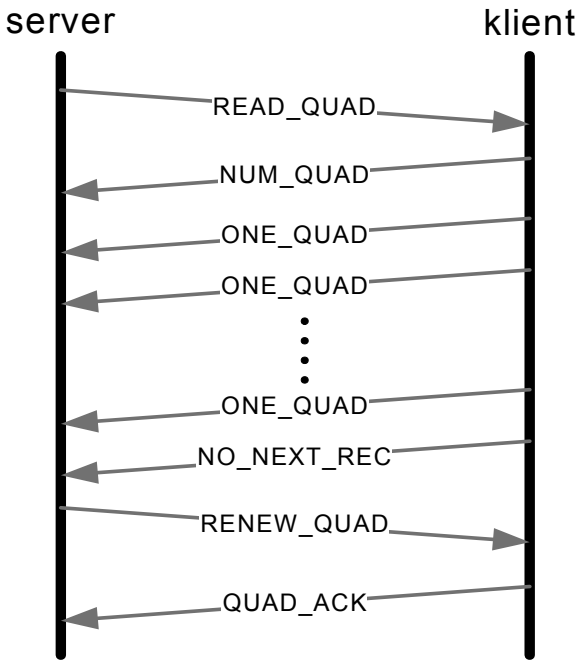
Server otevře spojení a čeká na určeném portu. Klient se připojí k serveru. Server očekává identifikaci klienta v podobě zprávy SMSKA. Pokud přijde jakákoli jiná zpráva, server ukončí spojení (je pravděpodobné, že se nepřipojil klient, ale jiná aplikace). V případě správné identifikace klienta odešle server klientovi zprávu SMSKA_SRV. V případě, že by klientovi došla jakákoli jiná zpráva než SMSKA_SRV, klient by ukončil spojení se serverem. Tímto je identifikační část protokolu ukončena a nastává příkazová část.

V příkazové části již není pravděpodobné, že by se mezi zasílanými zprávami objevila jakákoli jiná zpráva, než zprávy uvedené výše v seznamech. Příkazová část protokolu je vytvořena tak, že případná ztráta nějaké zprávy během přenosu (ač by toto nemělo nastat díky TCP/IP protokolu) či případné přerušení spojení, nedovolí vykonat operaci, která by vedla ke ztrátě dat uložených v mobilním zařízení. Přenesená data jsou po přenosu navíc kontrolována a případné zjištěné nekonzistence jsou ohlášeny uživateli v podobě zprávy s doporučeními jak dále pokračovat, aby byla tato nekonzistence odstraněna.

Protokol zajišťuje následující činnosti. Umožňuje serveru zjistit počet v klientovi uložených SMS zpráv, stažení všech uložených SMS zpráv z klienta na server, nahrání filtrů ze serveru do klienta.

Zjištění počtu v klientovi uložených SMS zpráv se děje za pomocí odeslání dotazu GET_NUM_QUAD. Klient po obdržení tohoto dotazu zjistí aktuální počet uložených SMS zpráv a odešle odpověď NUM_QUAD, která obsahuje tento zjištěný počet. Server zobrazí uživateli informaci o počtu v klientovi uložených zpráv.

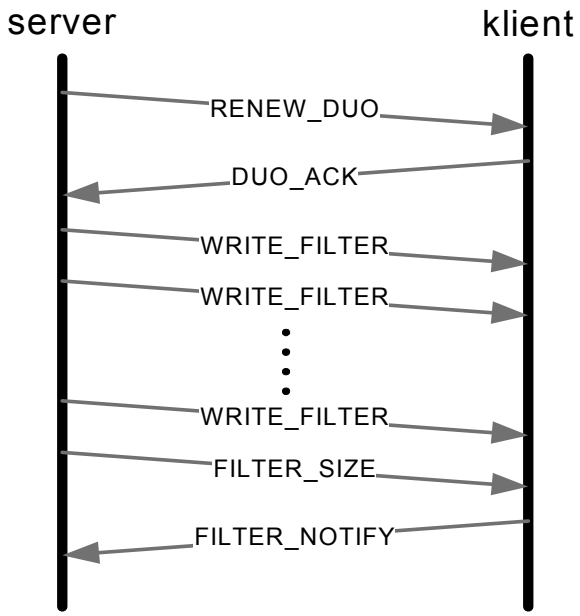
Průběh komunikace, která je nutná pro stažení všech uložených SMS zpráv z klienta na server, je poněkud složitější a zobrazuje ji Obr. 20. Nejprve je serverem odeslán dotaz READ_QUAD. Klient na základě tohoto dotazu zašle odpověď NUM_QUAD, která obsahuje počet uložených SMS zpráv v klientovi. Tento počet je později použit jako kontrola správnosti přenosu. Následně klient pokračuje zasíláním odpovědí ONE_QUAD, které obsahují data z úložiště uložených SMS zpráv. Klient zasílá tyto odpovědi tak dlouho, dokud neodešle všechny uložené SMS zprávy. Po odeslání všech uložených SMS zpráv klient zasílá serveru odpověď NO_NEXT_REC. Server provede kontrolu, zda odpovídá počet přenesených záznamů. Pokud neodpovídá, je uživatel vyzván k opakování přenosu a komunikace je ukončena. Pokud počet odpovídá, odešle server klientovi dotaz RENEW_QUAD, který způsobí v klientovi smazání všech položek v úložišti uložených SMS zpráv. Klient po smazání všech položek v úložišti odešle serveru odpověď QUAD_ACK. Tímto je stažení všech uložených SMS zpráv z klienta ukončeno.



Obr. 20 – komunikace nutná pro stažení všech uložených SMS zpráv z klienta

Průběh komunikace nutné k nahrání filtrovacích záznamů ze serveru do klienta zobrazuje Obr. 21. Serverem je nejdříve odeslán dotaz RENEW_DUO. Klient tento dotaz příjme a na jeho základě smaže všechny položky v úložišti filtrovacích pravidel. Klient po smazání všech položek v úložišti odešle serveru odpověď DUO_ACK. Server následně začne zasílat dotazy WRITE_FILTER, které obsahují data filtrovacích záznamů, které jsou

uloženy na serveru. Klient tyto dotazy zpracuje a data v nich obsažená uloží do úložiště filtrovacích pravidel. Server zasílá tyto dotazy tak dlouho, dokud neodešle všechny uložené filtrovací záznamy. Po odeslání všech filtrovacích záznamů server zasílá klientovi dotaz FILTER_SIZE, který obsahuje počet odeslaných filtrovacích záznamů. Klient na základě tohoto dotazu provede porovnání počtu uložených filtrovacích pravidel v úložišti a počtu uvedeném v dotazu. Na základě výsledku tohoto porovnání odesílá odpověď FILTER_NOTIFY, která obsahuje buďto potvrzení úspěšného přenosu filtrovacích záznamů, anebo negativní potvrzení. Server toto potvrzení zpracuje a uživateli zobrazí buďto zprávu o úspěšném přenosu filtrovacích záznamů, anebo o zprávu o nekonzistenci přenesených záznamů.



Obr. 21 – komunikace nutná k nahrání filtrovacích záznamů ze serveru do klienta

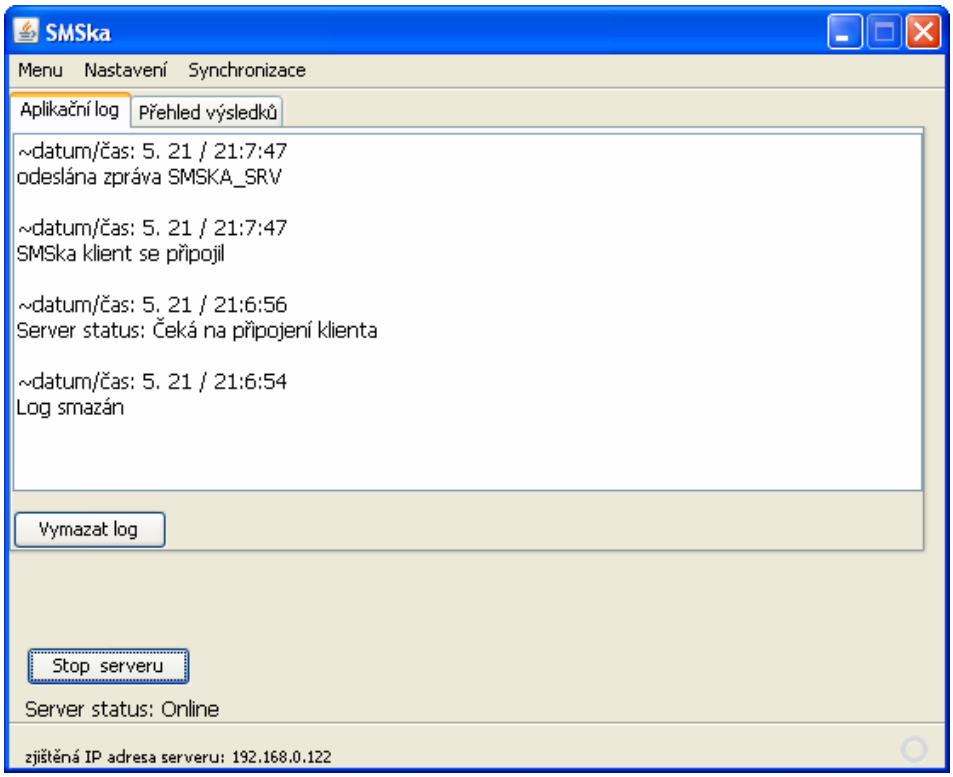
5.1.6. Realizace aplikace SMSkaServer

Aplikace *SMSkaServer* slouží ke komunikaci a nastavování vlastností sady MIDletů SMSka. Vlastní komunikace je uskutečňována mezi aplikací a MIDletem Synchronizace podle navrženého aplikačního protokolu. Spojení s tímto MIDletem je provedeno skrze Internet a to za pomoci socketů. Pro vytvoření aplikace SMSkaServer bylo použito skeletu desktopové aplikace založené na Swing aplikačním rámci. Tento skelet poskytlo při vytváření projektu vývojové prostředí NetBeans IDE 6.1. Mimo tříd tvořících tento skelet se aplikace SMSkaServer skládá z následujících tříd: *CopySMSData*, *DataOperations*, *EnumerateResults*, *FilterViewDialog*, *HelpDialog*, *PopUpDialog*, *PortSettingsDialog*, *SendFilters*, *Server*, *SMSkaServerApp*, *SMSkaServerView*.

Hlavní třídou obsahující metodu `main` je třída `SMSkaServerApp`. Tato třída vykonává v metodě `main` pouze dvě činnosti. Vytváří v umístění kde je aplikace spuštěna

pracovní adresář *data* a volá třídu `SMSkaServerView`, která zobrazí hlavní okno aplikace. Do adresáře data si aplikace ukládá pracovní soubory nutné pro správnou činnost aplikace. Aplikace je navržena tak, že při případném smazání tohoto pracovního adresáře nebo souborů v adresáři obsažených, nedojde k neočekávanému stavu v aplikaci, který by vedl k jejímu pádu.

Vzhled okna vytvořeného třídou `SMSkaServerView` je zobrazen na Obr. 22. Třída je zodpovědná za zobrazení okna *SMSka*, které je hlavním oknem aplikace. Okno obsahuje pruh nabídky menu (*Menu*, *Nastavení*, *Synchronizace*), dvě záložky (*Aplikační log* a *Přehled výsledků*), tlačítko pro spuštění/zastavení serveru a textové informace o serveru (*status* a *IP adresu*).



Obr. 22 – okno zobrazené třídou SMSkaServerView

Třídy `HelpDialog` a `PopUpDialog` obstarávají pouze jedinou činnost. Tou je zobrazování oken s informacemi a nápovědou.

Třída `PortSettingsDialog` zobrazuje okno, ve kterém je možno změnit číslo portu, na němž naslouchá server příchozímu spojení. Nastavené číslo portu si aplikace nikam neukládá a pamatuje si jej pouze po dobu běhu aplikace. Po startu aplikace je toto číslo nastaveno na implicitní hodnotu 2000. Metody třídy `PortSettingsDialog` se také starají o kontrolu správného formátu čísla nastavovaného portu. Zadané číslo musí být v rozsahu 1000 až 65534, aby jeho změna proběhla úspěšně.

Třída `Server` zajišťuje vlastní komunikaci aplikace (server) s MIDletem (klient). Instance třídy Server je implementována tak, že běží ve vlastním vlákně. Běh vlákna je

spuštěn po zmáčknutí tlačítka *Start serveru* v okně *SMSka*. Činnost vlákna lze ukončit zmáčknutím tlačítka *Stop serveru*. Vlákno je vytvářeno příkazem

```
srvSocket = new ServerSocket(port);
```

Serverový socket čeká na určeném portu (neomezeně dlouhou dobu) na připojení klienta. Připojení klienta je zjištěno pomocí příkazu

```
clientSocket = srvSocket.accept();
```

Třída `ServerSocket` poskytuje metodu `accept`, která vrací odkaz na třídu `Socket`. Tato metoda zablokuje běh vlákna, dokud nepřijde požadavek od klienta o navázání spojení. Přijde-li požadavek o navázání spojení, jsou otevřeny potřebné vstupní a výstupní streamy. Poté proběhne vzájemná komunikace serveru a klienta podle *identifikační* části navrženého aplikačního protokolu komunikace (viz kapitola 5.1.5). V této části se server a klient vzájemně autentizují. Dále se komunikace odehrává podle pravidel *příkazové* části aplikačního protokolu komunikace. Třída `Server` také zajišťuje korektní ukončení spojení (uzavření všech streamů, nastavení hodnot příznaků atd.).

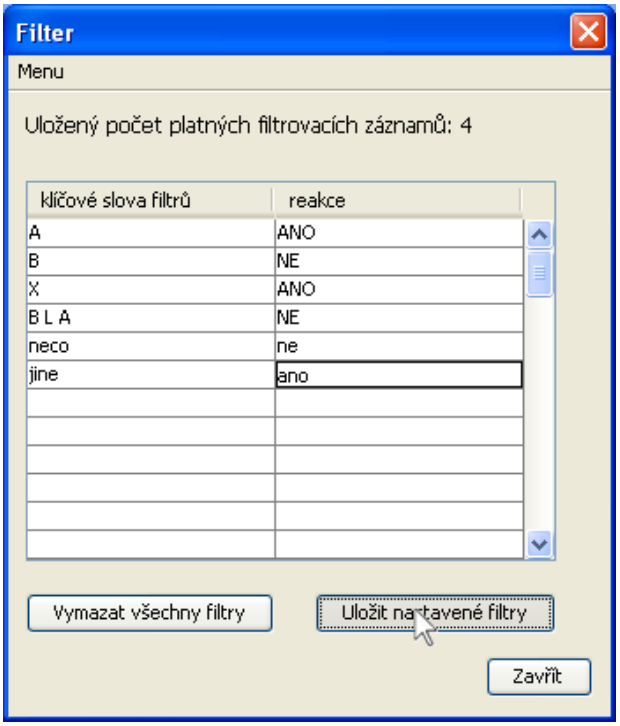
Třída `DataOperations` poskytuje ostatním třídám metody pro práci se soubory. Jak již bylo zmíněno výše, veškerá data se kterými aplikace SMSkaServer pracuje jsou ukládány v pracovním adresáři *data*. Třída `DataOperations` se především stará o uložení nastavených filtrovacích záznamů do souboru `filter.dat`. Dále pak o zápis přijímaných SMS záznamů do dočasného souboru `tempdata.dat`.

Třída *CopySMSData* je spouštěna ve vlastním vlákně a provádí jedinou činnost. Tou je překopírování stažených a verifikovaných dat z dočasného souboru `tempdata.dat` do souboru `smsdata.dat`. V tomto souboru jsou na serveru uloženy všechny stažené SMS záznamy, které kdy byly z klienta na server staženy od posledního vymazání souboru (pozn. Ze záznamů uložených v souboru `smsdata.dat` se vypočítává statistika. Jako rozlišovací pravidlo četnosti je použito hlasovacího textu. Tento soubor lze smazat z menu okna *SMSka* pomocí příkazu *Vymazat na serveru uložené SMS záznamy* nebo pomocí klávesová zkratky *Ctrl* + *Del* v tomto okně).

Třída `EnumerateResults` má na starosti výpočet a zobrazení dat do přehledu výsledků. Třída je implementována tak, že tuto činnost provádí ve vlastním vlákně. Data, ze kterých třída vypočítává výsledky, jsou načítány ze souboru `smsdata.dat`. Vlákno je spuštěno po zmáčknutí tlačítka *Zobraz výsledky*, které je umístěno na záložce *Přehled výsledků* okna *SMSka*. Třída nejprve spočítá četnost záznamů jednotlivých hlasovacích textů. Poté tyto záznamy setřídí sestupně tak, že na první pozici je hlasovací text s nejvyšší četností. Takto setříděné záznamy jsou vypsány do okna záložky *Přehled výsledků*.

Třída `FilterViewDialog` je zodpovědná za vytvoření okna *Filter*. Vzhled okna je zobrazen na Obr. 23. Okno obsahuje pruh nabídky *Menu*, ve kterém se skrývá jediná položka. A to položka *Nápověda*. Tato nápověda složí k vysvětlení práce s filtrovacími záznamy. Okno *Filter* dále obsahuje tabulku, do které lze zadávat jednotlivé filtrovací záznamy. Podle těchto filtrovacích záznamů jsou rozpoznávány příchozí SMS zprávy. Ale to až po té, co jsou tyto záznamy nahrány do klienta.

Nastavené filtrovací záznamy lze ukládat pomocí tlačítka *Uložit nastavené filtry*. Již nastavené filtry lze kompletně vymazat pomocí tlačítka *Vymazat všechny filtry*. Filtrovací záznamy jsou ukládány do souboru `filter.dat`. Filtry lze zapisovat pomocí malých nebo pomocí velkých písmen, anebo jejich kombinací. Toto nemá vliv na to, zda se filtr uloží či bude ignorován.



Obr. 23 – okno zobrazené třídou FilterViewDialog

Jeden záznam filtrovacího pravidla se skládá ze dvou částí. A to klíčového slova filtru a reakce. Klíčové slovo může být jakékoli slovo nebo více slov, obsahující alfanumerické znaky včetně mezer (kromě znaku |). Klíčové slovo nesmí začínat mezerou. Klíčové slova začínající mezerou nebo obsahující znak | budou ignorovány a neuloží se. Reakce může obsahovat pouze slova "ano", anebo "ne". Cokoli jiného ve slově reakce způsobí ignorování daného filtrovacího záznamu a jeho neuložení. Je jedno v jakém pořadí jsou filtry uloženy a zda jsou mezi jednotlivými záznamy filtrů mezery (tzn. nevyplněné řádky). Hodnota nastavená v buňce reakce určuje, zda na text zprávy vyhovující klíčovému slovu filtru bude reagováno (tzn. odeslána SMS zpráva).

Při ukládání jsou filtry kontrolovány na duplicitní záznamy. Filtry jsou ukládány od vrchu směrem dolů. Duplicity jsou rozlišovány podle klíčových slov filtrů. Pokud je při ukládání nalezeno duplicitní klíčové slovo, pak je celý filtrovací záznam ignorován.

Uložení nastavených filtrů proběhne po stisku tlačítka *Uložit nastavené filtry*. V jednotlivých buňkách je při ukládání zjišťováno, zda splňují pravidla pro uložení. Pokud splňují, jsou jednotlivé znaky v buňce převedeny na kapitálky a uloženy do souboru. Po uložení jsou v tabulce filtrovacích záznamů zobrazeny všechny uložené filtry.

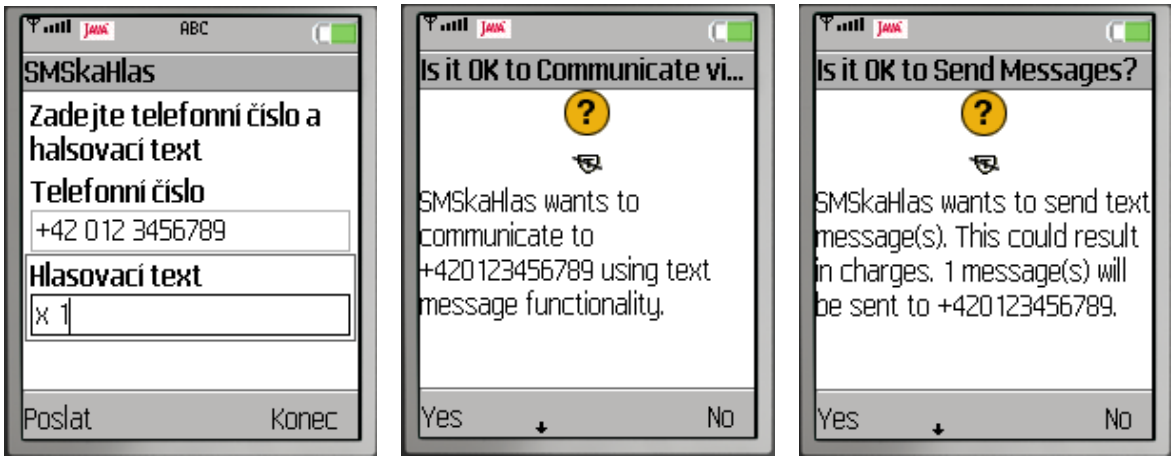
Smazání všech uložených filtrovacích záznamů lze provést pomocí tlačítka *Vymazat všechny filtry*. Pokud je požadováno vymazaní pouze jednoho nebo několika málo filtrovacích záznamů, stačí jednotlivé záznamy jakkoli zneplatnit (např. vymazáním obsahu buňky, zkomolením obsahu buňky reakce, přidáním mezery na začátek buňky) a následně provést uložení filtrů. Filtry nesplňující požadavky kontrolních mechanizmů (zneplatněné filtry) budou ignorovány a z uložených záznamů odstraněny.

5.2. Popis ovládání jednotlivých aplikací

V této kapitole bude uveden popis ovládání jednotlivých MIDletů a aplikace z uživatelského hlediska.

5.2.1. Popis ovládání MIDletu SMSkaHlas

MIDlet SMSkaHlas je jednoduchý na ovládání. Po startu MIDletu je zobrazena informační obrazovka. Tato obsahuje varování a doporučení, jak s MIDletem zacházet. Po jejím potvrzení je zobrazen hlavní formulář MIDletu (viz levá obrazovka Obr. 24). Zde uživatel vyplní číslo a hlasovací text. Pro správné započítání hlasu je nutné, aby byl přesně dodržen formát hlasovacího textu. Vyplněné údaje uživatel odešle pomocí tlačítka *Poslat*. Při prvním pokusu o odeslání SMS zprávy, je uživatel vyzván, aby umožnil aplikaci přijímat SMS zprávy (viz prostřední obrazovka Obr. 24). Pro správnou funkci MIDletu je nutné příjem povolit. Po povolení je uživatel dotázán, zda chce povolit odeslání SMS zprávy (viz pravá obrazovka Obr. 24). Aby byl hlas odeslán, je nutno i tento úkon povolit. Následně je zobrazena obrazovka informující uživatele o odeslání SMS zprávy. Po odeslání SMS zprávy je uživateli oznámeno, že SMS zpráva byla odeslána. Po potvrzení této obrazovky se zobrazí hlavní formulář MIDletu. Uživatel má možnost editovat údaje formuláře, zaslat další hlas nebo ukončit MIDlet.

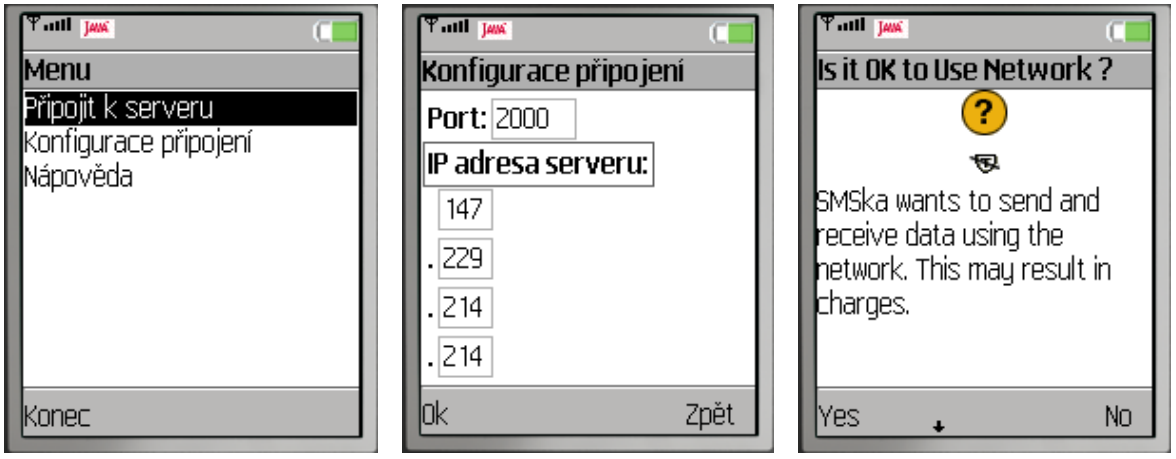


Obr. 24 – hlavní formulář MIDletu SMSkaHlas, žádost o povolení příjmu SMS zpráv, žádost o povolení odeslání SMS zprávy

5.2.2. Popis ovládání sady MIDletů SMSka

Sada MIDletů SMSka obsahuje MIDlet s názvem Synchronizace a MIDlet s názvem SMSka. Záleží na implementaci správce aplikací mobilního zařízení, zda bude ve správci aplikací sadu MIDletů zobrazena jako jedna položka (Sony Ericsson K750i), anebo zda budou MIDlety ze sady MIDletů vystupovat samostatně jakoby nesvázány (Nokia N73).

MIDlet Synchronizace: pro správnou funkci MIDletu je nutno mít správně nastaveny parametry datových přenosů pro daného mobilního operátora. Po startu MIDletu je zobrazeno hlavní menu (viz levá obrazovka Obr. 25). Uživatel by měl nejdříve přejít na položku *Konfigurace připojení* (viz prostřední obrazovka Obr. 25). Zde by měl vyplnit údaje podle nastavení aplikace SMSkaServer (IP adresu a port). Poté by měl vyplněné údaje potvrdit tlačítkem *Ok*. Bude zobrazena obrazovka potvrzující uložení údajů a po jejím potvrzení bude zobrazeno hlavní menu. Nyní lze zvolit položku *Připojit k serveru*. Zobrazí se obrazovka s varováním. Uživatel by měl zvolit *Připojit*. Při prvním pokusu o připojení na Internet bude dotázán, zda aplikaci povolit odesílání dat do sítě Internet (viz pravá obrazovka Obr. 25). Toto povolení by měl uživatel udělit. Po udělení povolení je zobrazen *komunikační log* a MIDlet se pokusí o spojení s aplikací SMSkaServer. V případě úspěšného spojení může uživatel provádět ovládání MIDletu pomocí aplikace SMSkaServer. Do komunikačního logu jsou vypisovány zprávy o průběhu komunikace (obdržené dotazy, MIDletem provedené akce). Uživatel má možnost vymazat log nebo ukončit spojení se serverem (pozn. před navázáním spojení, by měl uživatel zkontrolovat, zda je status aplikace SMSkaServer ve stavu *Waiting*, tzn. zda je spuštěn server a čeká na příchozí spojení).



Obr. 25 – hlavní menu MIDletu Synchronizace, formulář konfigurace připojení, žádost o povolení odeslání dat do sítě Internet

MIDlet SMSka: před prvním spuštěním MIDletu SMSka by měly být nejdříve do zařízení nahrány filtrovací záznamy pomocí MIDletu Synchronizace a aplikace

SMSkaServer. Pokud tomu tak nebude, MIDlet SMSka bude sice přijímat SMS zprávy, ale tyto nebudou zpracovávány, ani na ně nebude nijak reagováno.

Při prvním spuštění MIDletu je uživatel vyzván, aby umožnil aplikaci přijímat SMS zprávy (viz levá obrazovka Obr. 26). Pro správnou funkci MIDletu je nutné příjem SMS zpráv povolit. Od okamžiku povolení je MIDlet schopen přijímat SMS zprávy (příjem a zpracování je spuštěn ihned po startu MIDletu na pozadí a běží nezávisle na tom, co je zobrazeno na displeji). Následně je zobrazena obrazovka s výsledkem inicializace filtrů. Po potvrzení této obrazovky je zobrazena hlavní obrazovka MIDletu. Ta má podobu logu, ve kterém je nahoře zobrazen *informační pruh*. V informačním pruhu jsou zobrazeny informace o počtu SMS zpráv, které byly přijaty, zpracovány a odeslány, od startu MIDletu. Na hlavní obrazovku jsou do logu vypisovány informační zprávy. Hlavní obrazovce je přiřazeno pět příkazů (viz prostřední obrazovka Obr. 26). Příkaz *Konec* ukončí MIDlet. Příkaz *Nápověda* zobrazí novou obrazovku s jednoduchou nápovědou. Příkaz *Vymazat log* vymaže všechny záznamy z hlavní obrazovky. Příkaz *Filtry info* zobrazí informace o filtrovacích pravidlech (počet filtrovacích pravidel, dostupnou a využitou kapacitu paměti uloženými filtrovacími pravidly). Příkaz *SMS info* zobrazí informace o uložených SMS záznamech (počet záznamů, dostupnou a využitou kapacitu paměti uloženými záznamy SMS). Přijaté SMS zprávy jsou zpracovávány podle uložených filtrovacích pravidel. MIDlet SMSka je schopen běžet bezobslužně. Pokud je ovšem filtrovacím pravidlem určeno, že na příchozí SMS zprávu má být reagováno odesláním SMS zprávy, MIDlet se pokusí o odeslání SMS zprávy. V rámci tohoto pokusu o odeslání je uživateli položen dotaz (viz pravá obrazovka Obr. 26). Tento dotaz je zobrazen na displeji až do doby, než na něj uživatel reaguje. Při každém pokusu MIDletu o odeslání SMS zprávy je položen takovýto dotaz. Jednotlivé dotazy jsou řazeny za sebe do fronty a čekají dokud je uživatel neobslouží.

Obr. 26 – žádost o povolení příjmu SMS zpráv, hlavní obrazovka MIDletu s příkazy, žádost o povolení odeslání SMS zprávy

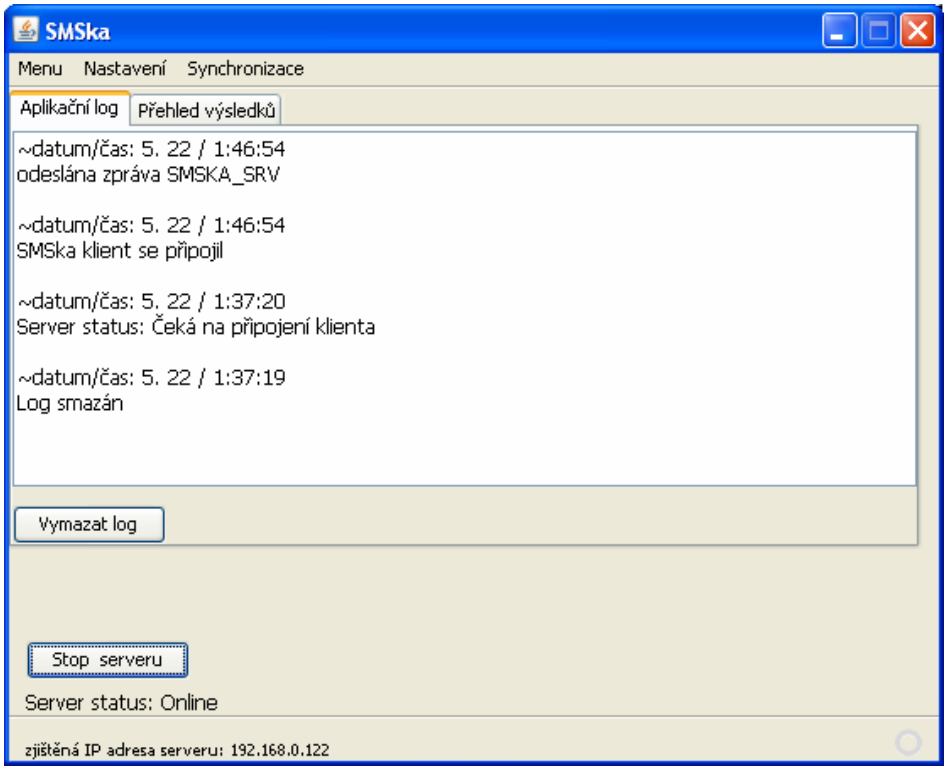
5.2.3. Popis ovládání aplikace SMSkaServer

Aplikace SMSkaServer slouží jako socketový server. Aplikace komunikuje s klientským MIDletem SMSka. Hlavními funkcemi aplikace jsou správa filtrovacích

záznamů pro klienta, nahrání filtrovacích záznamů do klienta, stahování dat uložených v klientovi, prezentaci těchto stažených dat. Pro správnou funkci aplikace je nutné, aby měl program v adresáři kde je spuštěn práva pro vytváření adresářů, souborů a práva pro jejich mazání. Dále je nutné, aby počítač, na němž je program spuštěn, měl veřejnou IP adresu a byla povolena komunikace na zvoleném portu (výchozí port je nastaven na hodnotu 2000).

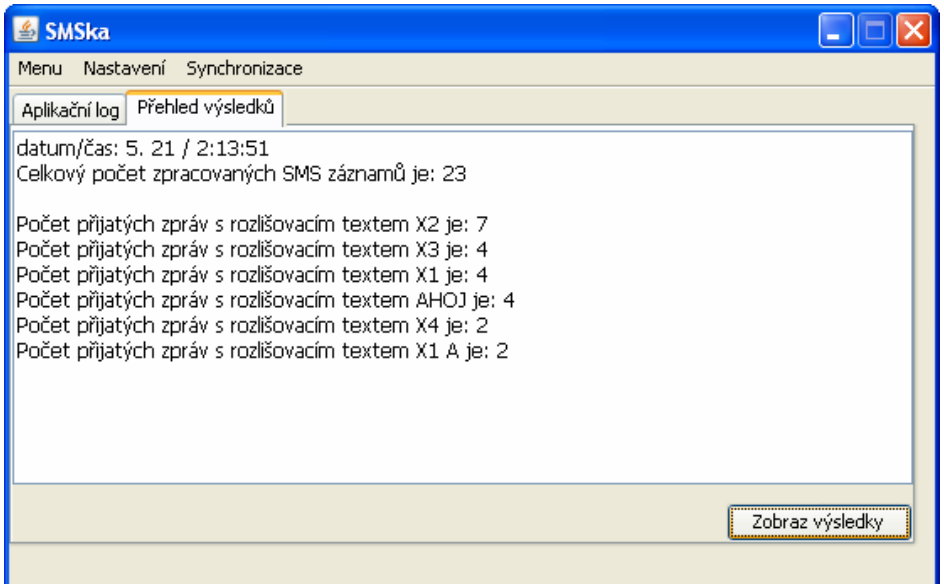
Po spuštění aplikace SMSkaServer je zobrazeno hlavní okno aplikace (nazvané *SMSka*, viz Obr. 27). Okno obsahuje pruh nabídky s položkami *Menu*, *Nastavení* a *Synchronizace* (nabídka Synchronizace je dostupná pouze tehdy, je-li připojen klient). Dále záložku *Aplikační log* a záložku *Přehled výsledků*. Součástí okna SMSka je tlačítko pro *Start* / *Stop serveru* a textová informace *Server status*. Tlačítko Star / Stop serveru spouští nebo zastavuje činnost serveru. Ve stavovém řádku okna je zobrazena *zjištěná IP adresa serveru* (jde o aplikací zjištěnou IP adresu počítače, na němž je aplikace spuštěna).

V záložce *Aplikační log* se zobrazují v textové podobě události týkající se běhu serveru. Především je zde zobrazen průběh komunikace s klientem. Obsah aplikačního logu lze vymazat pomocí tlačítka *Vymazat log*.



Obr. 27 – hlavní okno aplikace SMSkaServer

Záložka *Přehled výsledků* je určena k zobrazení statistik výsledků. Tyto statistiky jsou vytvářeny ze záznamů uložených SMS zpráv, které jsou k dispozici na serveru. Výsledky jsou prezentovány v textové podobě (viz Obr. 28). Ke generování statistik slouží tlačítko *Zobraz výsledky*.



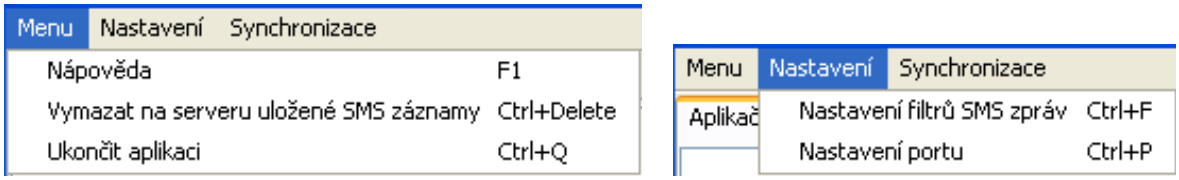
Obr. 28 – vygenerovaná statistika z výsledů SMS záznamů uložených na serveru

Každá položka *Menu* z pruhu nabídky má přiřazenu klávesovou zkratku. Tuto klávesovou zkratku lze použít pro rychlejší vyvolání dané akce, pokud je zobrazeno okno SMSka.

Položka *Menu* (viz levá část Obr. 29) obsahuje tři volby. Volba *Nápověda* zobrazí okno s nápovědou. Tato nápověda obsahuje popis aplikace.

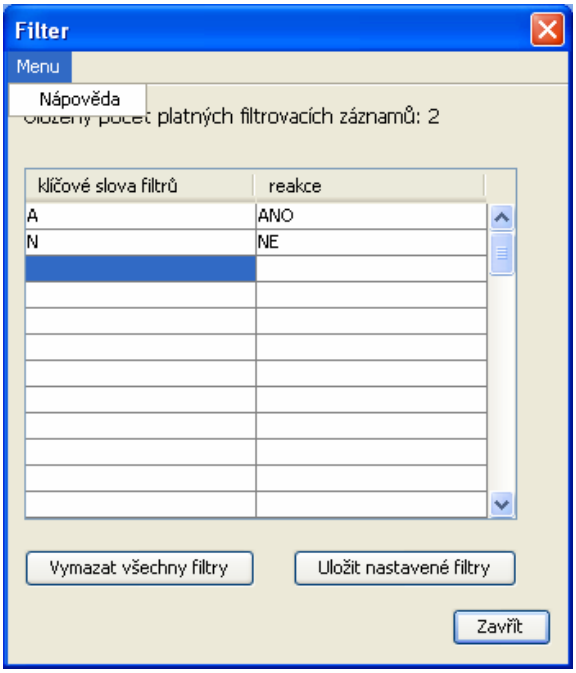
Volba *Vymazat na serveru uložené SMS záznamy* způsobí vymazání všech doposud na serveru uložených SMS záznamů (pozn. statistiky jsou tvořeny z uložených SMS záznamů na serveru).

Volba *Ukončit aplikaci* ukončí a zavře aplikaci SMSkaServer.



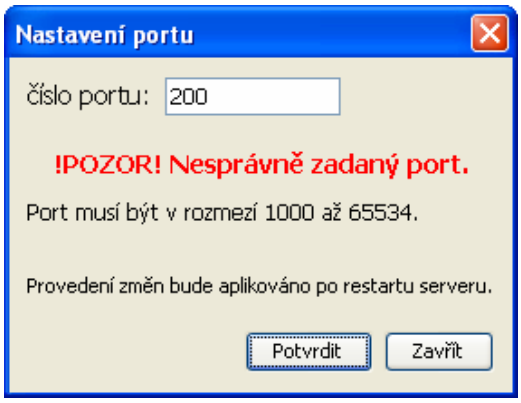
Obr. 29 – možné volby položky Menu a položky Nastavení z pruhu nabídky

Položka *Nastavení* (viz pravá část Obr. 29) obsahuje volbu *Nastavení filtrů SMS zpráv*. Tato volba zobrazí okno *Filter* (viz Obr. 30), ve kterém je možnost nastavení filtrovacích záznamů.



Obr. 30 – okno Filter sloužící k nastavování filtrovacích záznamů

Okno Filter obsahuje pruh nabídky *Menu*, ve kterém se skrývá jediná položka. A to položka *Nápověda*. V této nápovědě je vysvětlena práce s filtrovacími záznamy a možnosti tohoto okna. Okno *Filter* dále obsahuje tabulku, do které lze zadávat jednotlivá filtrovací pravidla. Jeden záznam filtrovacího pravidla se skládá ze dvou částí. A to *klíčového slova filtru* a *reakce*. Podle těchto filtrovacích záznamů budou rozpoznávány příchozí SMS zprávy (ale to až po nahrání těchto filtrovacích záznamů do klienta). Nastavené filtry lze ukládat pomocí tlačítka *Uložit nastavené filtry*. Již nastavené filtry lze kompletně vymazat pomocí tlačítka *Vymazat všechny filtry*.



Obr. 31 – okno nastavení portu

Položka *Nastavení* (viz pravá část Obr. 29) obsahuje také volbu *Nastavení portu*. Tato volba zobrazí okno *Nastavení portu* (viz Obr. 31). V tomto okně lze změnit hodnotu portu, na kterém aplikace naslouchá příchozímu spojení (pozn. při zapnutí aplikace je nastavena

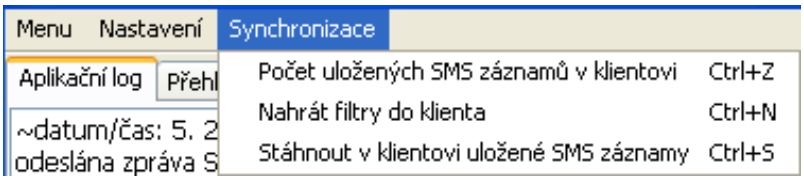
implicitní hodnota portu, které má hodnotu 2000). Po potvrzení změny tlačítkem *Potvrdit* je kontrolováno, zda jsou vyplněné údaje vyhovující a lze je uložit. V opačném případě je uživatel informován a nabádán k nápravě.

Položka *Synchronizace* (viz Obr. 32) je dostupná pouze v případě, když je připojen klient. Obsahuje volby, které umožňují výměnu dat aplikace a klienta.

Po volbě *Počet uložených SMS záznamů v klientovi* začne aplikace komunikovat s klientem. Po získání dat od klienta, dojde k zobrazení informačního okna s počtem uložených SMS záznamů v klientovi.

Volba *Nahrát filtry do klienta* vyvolá komunikaci mezi aplikací a klientem, v jejímž důsledku jsou do klienta nahrány filtrovací záznamy uložené na serveru (pokud jsou nějaké definovány).

Volba *Stáhnout v klientovi uložené SMS záznamy* spouští akci, při níž se přenesou všechny v klientovi uložené SMS záznamy na server. Stažené SMS záznamy jsou přidány k těm, které jsou již na serveru uloženy.



Obr. 32 – možné volby položky Synchronizace z pruhu nabídky

5.2.4. Předpokládaný způsob použití aplikací

Nyní uvedu posloupnost kroků, které by měly zajistit bezproblémové a smysluplné použití aplikací.

1. Instalace sady MIDletů SMSka.
2. Vytvoření a uložení filtrovacích záznamů (čímž je de facto vytvořena anketa).
3. Nahrání filtrovacích záznamů do klienta (čímž jsou nastavena filtrovací pravidla).
4. Distribuce MIDletů SMSkaHlas.
5. Příjem hlasů automatizovaným hlasovacím SMS serverem (MIDlet SMSka).
6. Stažení uložených hlasů ze sady MIDletů SMSka do aplikace SMSkaServer.
7. Zobrazení přehledu výsledků.

Pozn. Asi nejjednodušší distribucí MIDletu je jeho vystavení na webových stánkách, odkud si je mohou potencionální uživatelé stáhnout a nainstalovat pomocí O-T-A.

5.3. *Praktické testy*

V této kapitole uvedu své zkušenosti z praktického testování vytvořených aplikací.

5.3.1. Předpoklady vs. praxe

Vytvoření 3 MIDletů a jedné aplikace představuje největší projekt, jaký jsem doposud za pomocí jazyka Java a jeho platforem realizoval. Před tímto projektem jsem měl zkušenosti pouze s vývojem jen velice jednoduchých aplikací.

Má představa postupu vývoje byla takováto. Vytvořím MIDlety. Tyto s největší pečlivostí odsimuluji v emulátorech mobilních telefonů (viz Obr. 33). Po dosažení plné funkčnosti v emulátorech, MIDlety přenesu na mobilní telefon a tyto bezproblémově poběží.

Obr. 33 – emulátory mobilních telefonů (Default color phone, Sony Ericsson K750 emu, Sony Ericsson K310 emu)

Jelikož vlastním dva mobilní telefony značky Sony Ericsson (model K750i a model K310i), rozhodl jsem se, že vývoj povedu se záměrem výsledné MIDlety testovat a případně provozovat na těchto mobilních telefonech. Proto jsem si ze stránek výrobce stáhl vývojovou sadu Sony Ericsson SDK 2.5.0 for the Java ME Platform. Tato vývojová sada

obsahuje emulátory mobilních telefonů této značky. Tyto emulátory jsem integroval do vývojového prostředí NetBeans IDE 6.1 a používal k vývoji.

Když jsem shledal, že vytvořené MIDlety jsou v emulátorech plně funkční, přikročil jsem k testům na skutečných zařízeních. K mému překvapení MIDlety, které byly plně funkční v emulátoru, na skutečných zařízeních nepracovaly.

Naštěstí mobilní telefon Sony Ericsson K750i, ostatně jako i mnoho dalších nových modelů mobilních telefonů jiných značek, podporuje tzv. ladění na zařízení (debug on device). Toto ladění na zařízení (využívající KVM Debug Wire Protocol – KDWP) umožňuje po připojení mobilního telefonu k osobnímu počítači spustit MIDlet na mobilním telefonu takovým způsobem, že informace určené pro výpis do textové konzole MIDletu, je možno díky KDWP zobrazovat v textové konzoli na osobním počítači. Nevýhodou takovéhoto ladění je ovšem jeho velká časová náročnost.

5.3.2. Výpis informací z běhu MIDletu na displej

Jelikož reálné mobilní telefony nedisponují textovou konzolí, vytvořil jsem si metodu, která měla vypisovat ladící informace na displej mobilního telefonu. Původní představa byla, že si celkovou délku textu takto zobrazené informace, mohu zvolit libovolně. První spuštění MIDletu, který obsahoval tuto metodu pro výpis informací na displej na reálném mobilním telefonu, však dopadlo neúspěchem. A to proto, že se výpis informací choval podivně. Jakoby MIDlet překračoval hranice paměti pro danou komponentu, která se starala o zobrazení těchto informací. Chování této metody bylo tak nekorektní, že v případě pokusu o spuštění tohoto MIDletu na mobilním telefonu BenQ-Siemens S68, došlo dokonce k restartu mobilního telefonu (podle informací majitele mobilního telefonu, firmware tohoto telefonu nebyl příliš stabilní, což se projevovalo restarty mobilního telefonu i během běžného používání).

Díky funkci ladění na zařízení (debug on device) se podařilo zjistit, že maximální dálka textu, kterou je možno na mobilních telefonech značky Sony Ericsson (pravděpodobně i jiných značek) zobrazit, je 450 znaků. Na základě tohoto zjištění jsem provedl úpravu metody pro zobrazení ladících informací tak, aby zobrazovala maximálně oněch 450 znaků. Po této úpravě bylo možno MIDlet spustit bez problémů i na mobilním telefonu BenQ-Siemens S68.

5.3.3. Socketové spojení v praxi

V MIDletu Synchronizace je využito socketů pro spojení se serverem skrze Internet. V profilu MIDP 2.0 je definováno rozhraní SocketConnection z Generic Connection Framework (viz kapitola 3.8.1). Očekával jsem, že každé mobilní zařízení, které podporuje profil MIDP 2.0, bude schopno tohoto socketového spojení. Praxe byla opět jiná. Při pokusu o navázání spojení na mém mobilním telefonu K750i se aplikace jevila, jako že se pokouší o spojení, avšak žádná komunikace se neuskutečnila. Při spuštění MIDletu na mobilním telefonu K310i došlo ke spojení se serverem bez problémů. Proto jsem opět použil funkce ladění na zařízení a zjistil jsem, že spojení v mobilním telefonu K750i vyvolalo tuto chybu:

`File not found: com/sun/midp/io/j2me/socket/Protocol.class.`

Při zjišťování informací týkajících se této chyby vyšlo najevo, že se jedná o chybu ve firmware telefonu. Tato chyba je docela nepříjemná. Protože při pokusu o spuštění MIDletu na jiném mobilním telefonu K750i došlo ke spojení se serverem bez problémů. Jen uvedu, že zjištěná verze firmware (R1CA021) mého i druhého mobilního telefonu byly stejné. Jediný rozdíl byl ve vlastních nastaveních telefonu operátorem (tzv. customization).

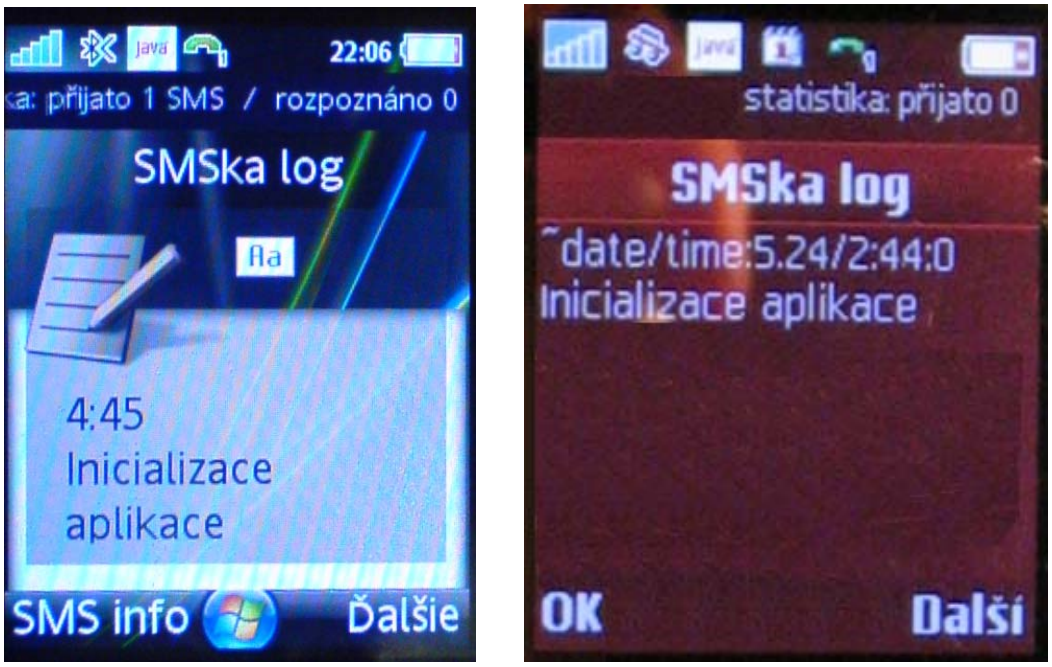
5.3.4. Velikost dostupné paměti pro běh Javy (Heap size)

Při testování aplikací na mobilních telefonech K310i, K750i, K300i jsem zjistil, že MIDlet SMSka je plně funkční a je schopen bez problémů přijímat SMS zprávy. Ale to pouze do doby, než má MIDlet odeslat SMS zprávu v rámci reakce na příchozí SMS zprávu. V tento okamžik je na displeji zobrazen dotaz pro uživatele, zda povolit odeslání SMS zprávy. A právě tato operace způsobí, že mobilnímu telefonu dojde paměť, která je určena pro běh MIDletu (tzv. Heap). Při bližším zkoumání společných vlastností těchto mobilních telefonů jsem zjistil, že podle specifikace disponují 1,5 MB dostupné paměti pro běh Javy. Proto jsem se pokusil najít mobilní telefony, které mají více dostupné paměti pro běh Javy. Při zkoumání parametrů na stránkách výrobců jsem zjistil, že dnes vyráběné mobilní telefony disponují již mnohem větší kapacitou dostupné paměti pro běh Javy než je 1,5 MB. Podařilo se mi získat na krátkodobý test mobilní telefon značky Nokia N73. Zde aplikace neměla problém s dalším příjmem SMS zpráv i po zobrazení dotazu pro povolení odeslání SMS zprávy. Podle stránek výrobce mobilní telefon Nokia N73 disponuje neomezenou velikostí paměti pro běh Javy (unlimited Heap size). V rámci testů se mi podařilo zajistit na krátkodobý test mobilní telefon Sony Ericsson K810i. Tento mobilní telefon je zajímavý především tím, že disponuje 3 MB dostupné paměti pro běh Javy. Zajímalo mne, zda tato velikost bude dostatečná pro bezproblémový běh MIDletu. Při testu mobilní telefon dále přijímal SMS zprávy i po zobrazení dotazu pro povolení odeslání SMS zprávy. Tudíž lze říci, že pro běh MIDletu SMSka je minimálně požadováno 3 MB dostupné paměti pro běh Javy. Ovšem vzhledem k tomu, že pro odeslání SMS zprávy je vytvořená instance třídy `SMSSending` a každé takovéto odeslání vyžaduje potvrzení od uživatele, je pravděpodobné, že při neobsluhování MIDletu a tím pádem velkém počtu nepotvrzených dotazů, by mobilnímu telefonu došla paměť.

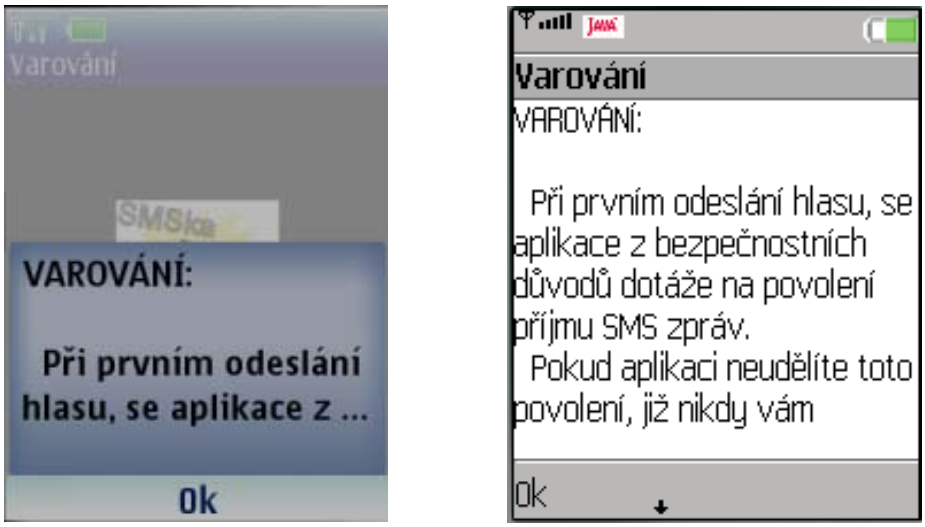
5.3.5. Rozdílnost mobilních zařízení (device fragmentation)

Jak jsem již uvedl, nejvíce jsem testoval na mobilních telefonech Sony Ericsso K310i a K750i, neboť tyto mobilní telefony vlastním. Vývoj MIDletů po grafické stránce byl tedy proveden tak, aby zobrazení na těchto mobilních telefonech bylo korektní a přehledné. Během doby vývoje se mi také podařilo provést krátké testování MIDletů na mobilních telefonech Sony Ericsson K300i, K810i, Nokia N73, 5300, E50 a BenQ-Siemens S68. Během těchto testů jsem narážel na rozdílnost jednotlivých mobilních telefonů (tzv. device fragmentation). Tato rozdílnost se projevovala především velikostí rozlišení displeje, počtem ovládacích prvků mobilního telefonu, rozdílností zobrazování jednotlivých prvků API na displeji a také rozdílností implementace platformy Java ME.

Jako příklad lze uvést rozdílné zobrazení textového pole SMSka log na mobilním telefonu Sony Ericsson K750i a K810i (viz Obr. 34). Podle specifikací výrobce spadá model K750i do JP-5 (Java Platform) a model K810i do JP-7. Výrobce Sony Ericsson používá Java Platform dělení zařízení, aby vývojářům usnadnil orientaci mezi modely mobilních telefonů této značky. Zařízení zařazená do stejné skupiny Java Platform vykazují podobné vlastnosti implementace Java ME platformy. Více viz literatura [32].



Obr. 34 – rozdílné zobrazení textového pole v závislosti na verzi Java Platform u Sony Ericsson (vpravo JP-7 K810i, vlevo JP-5 K750i)



Obr. 35 – rozdílné zobrazení výstrahy na displeji mobilního telefonu značky Nokia a Sony Ericsson

Dalším příkladem rozdílnosti implementace Java ME platformy na mobilních telefonech může být zobrazení výstrahy. U značky Sony Ericsson je výstraha zobrazena jako nová obrazovka. Kdežto u značky Nokia je výstraha zobrazována jakoby v okně, které překryje obrazovku na pozadí (viz Obr. 35). Tato implementace u Nokií neumožňuje zobrazení dvou obrazovek výstrahy za sebou. MIDlet je po spuštění na Nokii Series 40 ukončen s chybovým hlášením a u Series 60 se MIDlet zablokuje a je nutno jej ukončit pomocí správce aplikací.

Z výše uvedeného vyplývá, že vytvořit MIDlet, který by byl univerzální a korektně se zobrazil na všech mobilních telefonech bez nutnosti úprav a bez možnosti reálného testování na daném zařízení, je velice obtížné. V praxi se tyto případy řeší tak, že jsou MIDlety vytvářeny cíleně pro určitou skupinu zařízení (Sony Ericsson a jejich Java Platform, Nokia a jejich Series 40/60 s rozšířením future pack apod.), anebo pro jedno konkrétní zařízení.

5.4. Výsledek

Během praktického testování byly realizované aplikace mírně upraveny tak, aby bylo možné jejich nasazení na co největším počtu mobilních zařízení. Byly určeny minimální požadavky, při jejichž splnění je pravděpodobné, že aplikace půjde spustit a bude plnit svou funkci.

Minimální požadavky pro nasazení sady MIDletů SMSka na mobilním zařízení jsou: podpora platformy Java ME, konfigurace CLDC 1.1, profil MIDP 2.0, 3 MB dostupné paměti pro běh Javy, barevný grafický displej s rozlišením 128 * 128 obrazových bodů, správně nakonfigurované připojení k Internetu.

Minimální požadavky pro spuštění MIDletu SMSkaHlas na mobilním zařízení jsou: podpora platformy Java ME, konfigurace CLDC 1.1, profil MIDP 2.0, 1 MB dostupné paměti pro běh Javy, barevný grafický displej s rozlišením 128 * 128 obrazových bodů.

Minimální požadavky pro spuštění aplikace SMSkaServer jsou: běhové prostředí Java Standard Edition Runtime Environment version 5 nebo vyšší, práva aplikace v místě spuštění vytvářet adresáře a soubory, povolení komunikace pomocí Internetu na daném portu (implicitní port má hodnotu 2000), veřejná IP adresa, operační systém MS Windows 2000 nebo vyšší (pozn. ačkoli Java SE platforma umožňuje vytvořit platformě nezávislé aplikace, aplikace SMSkaServer nebude funkční na jiných operačních systémech než MS Windovs, kvůli použití oddělovače cest k souborům, který používají pouze operační systémy MS Windows).

6. Závěr

Cílem diplomové práce bylo prostudovat možnosti příjmu a odesílání SMS zpráv na platformě Java ME. Poté na základě teoretických poznatků navrhnout a vytvořit aplikaci na této platformě, která bude využitelná pro rozesílání i příjem SMS zpráv. Následně realizovat spojení aplikace s ovládacím rozhraním na PC, vytvořeném na platformě Java SE.

Před samotným řešením praktických úkolů zadání jsem v teoretické části práce obecně rozebral Java platformu a poté se zaměřil na vlastnosti platformy Java ME. Zde byla podrobněji rozebrána architektura platformy Java ME, a to především s ohledem na zařízení s omezenými zdroji. Byly rozebrány pojmy důležité pro pozdější realizaci aplikací, jako jsou správa průběhu aplikací a manažer aplikací (AMS), bezpečnostní model MIDP, volitelné balíčky a jejich použití, vlastnosti volitelného balíčku Wireless Messaging API (WMA). Dále jsou v práci uvedeny zjištěné skutečnosti ve vztahu podpory balíčku WMA jednotlivými výrobci mobilních telefonů.

Na základě zadání a poznatků získaných z teorie jsem provedl návrh řešení. Jedná se o řešení automatizovaného hlasovacího SMS serveru, který je určen pro provoz na mobilních zařízeních podporující platformu Java ME, a který je ovládán z aplikace určené pro platformu Java SE. Návrh řešení se sestává ze 3 MIDletů a ovládací aplikace. MIDlety využívají volitelného balíčku WMA pro odesílání a přijímání SMS zpráv. Pro komunikaci mezi MIDletem a ovládací aplikací je využito spojení pomocí Internetu. Takto navržené řešení jsem po poté realizoval.

Výsledkem realizace je MIDlet SMSkaHlas. Tento MIDlet slouží pro zasílání hlasů od jednotlivých hlasujících účastníků. MIDlet je určen k distribuci mezi uživatele. Tato distribuce může být realizována buďto nahrání MIDletu do mobilního zařízení pomocí kabelu, infraportu bluetooth, paměťové kary, anebo stažením MIDletu z webové stánky pomocí Internetu (O-T-A). MIDlet má na mobilní zařízení, na které je instalován, tyto minimální požadavky: podpora platformy Java ME, konfigurace CLDC 1.1, profil MIDP 2.0, 1 MB dostupné paměti pro běh Javy, barevný grafický displej s rozlišením 128 * 128 obrazových bodů. MIDlet je vytvořen tak, aby mohl pracovat na co největším počtu mobilních zařízení. V praxi byl ozkoušen pouze na mobilních telefonech značky Sony Ericsson (K300i, K310i, K750i, K810i) a Nokia (5300, E50), kde pracoval korektně.

Dalším realizovaným výsledkem je sada MIDletů nazvaná SMSka. Tato sada MIDletů v sobě obsahuje MIDlet Synchronizace a MIDlet SMSka. MIDlet Synchronizace se stará o spojení mobilního zařízení s ovládací aplikací skrze Internet. V rámci tohoto spojení je možno přenést data uložená v mobilním zařízení do ovládací aplikace a filtrovací záznamy uložené v ovládací aplikaci do sady MIDletů. MIDlet SMSka je jádrem automatizovaného hlasovacího SMS serveru a má na starost příjem, zpracování a případnou reakci na příchozí SMS zprávy. Zpracování a reakce na příchozí SMS zprávu závisí na nastavení filtrovacích pravidel, které byly do MIDletu uloženy z ovládací aplikace. Příjem a zpracování příchozích SMS zpráv je plně automatický a není nutno žádného zásahu uživatele. Bohužel při reakci v podobě odeslání SMS zprávy s reakčním textem je potřeba zásahu uživatele. Tato nutnost zásahu uživatele je dána bezpečnostní politikou MIDP profilu verze 2.0. Tato

bezpečnostní politika neumožňuje odeslat SMS zprávu bez souhlasu uživatele. To v důsledku znemožňuje provoz plně automatizovaného hlasovacího SMS serveru tak, aby na přijatou zprávu reagoval odesláním SMS zprávy. Možným řešením tohoto stavu by bylo digitální podepsání aplikace certifikátem, tak aby se aplikace stala důvěryhodnou, a současně provoz MIDletu na mobilním zařízení podporující profil MIDP 2.1. Tyto kroky by měly umožnit nastavit bezpečnostní politiku profilu MIDP 2.1 pro MIDlet tak, aby při odesílání SMS zprávy nebyl nutný souhlas uživatele, ale vše probíhalo automaticky (viz. [33]). Toto řešení bylo zkoumáno pouze za pomocí emulátorů a není ověřeno praktickým testem, neboť pořízení certifikátu pro digitální podpis aplikací (nejen Java, ale obecně všech aplikací) je velice nákladné (cca 140 $). MIDlet má na mobilní zařízení, na které je instalován, tyto minimální požadavky: podpora platformy Java ME, konfigurace CLDC 1.1, profil MIDP 2.0, 3 MB dostupné paměti pro běh Javy, barevný grafický displej s rozlišením 128 * 128 obrazových bodů. Sada MIDletů byla nejvíce testována na mobilních zařízeních značky Sony Ericsson (K310i a K750i). Bohužel na těchto zařízeních není běh možný v plném rozsahu v důsledku nedostatku paměti pro běh Javy (viz kapitola 5.3.4). Podařilo se ale zajistit krátké testování aplikace na mobilních telefonech Sony Ericsson K810i a Nokia N73, kde se sada MIDletů během tohoto testování jevila jako plně funkční.

Posledním realizovaným výsledkem práce je ovládací aplikace. Tato aplikace je realizována na platformě Java SE a má název SMSkaServer. Aplikace slouží k ovládání chování sady MIDletů SMSka. Pomocí této aplikace lze vytvořit a uložit filtrovací záznamy, které je možno poté přenést do sady MIDletů. Sada MIDletů pomocí těchto uložených filtrovacích záznamů řídí zpracování příchozích SMS zpráv. Aplikace SMSkaServer také umožňuje stažení uložených SMS zpráv ze sady MIDletů a jejich uložení v aplikaci. Z takto uložených zpráv je možno vytvořit statistiky četnosti přijatých SMS zpráv. Minimální požadavky pro spuštění aplikace SMSkaServer jsou: běhové prostředí Java Standard Edition Runtime Environment version 5 nebo vyšší, práva aplikace v místě spuštění vytvářet adresáře a soubory, povolení komunikace pomocí Internetu na daném portu (implicitní port má hodnotu 2000), veřejná IP adresa, operační systém MS Windows 2000 nebo vyšší.

Jako možné rozvinutí této práce bych viděl lepší prezentaci četnosti záznamů uložených SMS zpráv. Určitě by nebylo na škodu nějaké grafické znázornění pomocí grafů. Další rozvinutí by bylo možné za předpokladu, že by aplikace byla digitálně podepsána certifikátem a provozována na zařízení s podporou MIDP 2.1. Pak by bylo možno po menších programových úpravách aplikací tyto použít např. pro reklamní účely, kdy by zájemce o reklamu použil MIDlet SMSkaHlas k odeslaní žádosti o reklamní sdělení a následně by mu byla sdělena propagační informace.

7. Použitá literatura

[1] Sun Microsystems, Inc. *JSRs: Java Specification Requests - JSR Overview* [online]. c2006. [cit. 2006-11-24]. Dostupné z URL: <http://jcp.org/en/jsr/overview>.

[2] GOSLING, J., MCGILTON, H. *The Java Language Environment: Contents* [online]. Poslední aktualizace květen 1996. [cit. 2006-11-24]. Dostupné z URL: <http://java.sun.com/docs/white/langenv/>.

[3] Byous, J. *JAVA TECHNOLOGY: THE EARLY YEARS* [online]. Poslední aktualizace duben 2003. [cit. 2006-11-24]. Dostupné z URL: <http://java.sun.com/features/1998/05/birthday.html>.

[4] MAHMOUD, Qusay, H. *Naučte se JAVA 2 Micro Edition.* Praha: Grada Publishing a.s., 2002. 246 s. ISBN 80-247-0444-7.

[5] *Programovací jazyk Java* [online]. Poslední aktualizace 1999-11-29. [cit. 2006-11-24]. Dostupné z URL: <http://www.e-mental.com/dvorka/logr/doc/logrbook/node3.html>.

[6] *Java* [online]. Poslední aktualizace 2006-11-20. [cit. 2006-11-24]. Dostupné z URL: <http://cs.wikipedia.org/wiki/Java>.

[7] PŠENČÍKOVÁ, J. *VisualAge for Java 2.0 - Příručka programátora.* Brno: Computer Press, 1999. 372 s. ISBN 80-7226-178-9.

[8] PIROUMIAN, V. *Wireless J2ME Platform Programming.* USA: Prentice Hall PTR, 2002. 400 s. ISBN: 0-13-044914-8.

[9] Sun Microsystems, Inc. *JSR-000030 J2ME(TM) Connected, Limited Device Configuration Specification 1.0a Final Release* [online]. Poslední aktualizace 2000-05-19. [cit. 2006-12-02]. Dostupné z URL: <http://jcp.org/aboutJava/communityprocess/final/jsr030/index.html>.

[10] Sun Microsystems, Inc. *JSR-000030 J2ME(TM) Connected, Limited Device Configuration Specification 1.1 Final Release* [online]. Poslední aktualizace březen 2003. [cit. 2006-12-02]. Dostupné z URL: <http://jcp.org/aboutJava/communityprocess/final/jsr139/index.html>.

[11] Sun Microsystems, Inc. *JSR-000036 Connected Device Configuration 1.0b Maintenance Release* [online]. Poslední aktualizace 2005-12-20. [cit. 2006-12-02]. Dostupné z URL: <http://jcp.org/aboutJava/communityprocess/mrel/jsr036/index.html>.

[12] Sun Microsystems, Inc. *JSR-000218 Connected Device Configuration 1.1.2 Maintenance Release* [online]. Poslední aktualizace 2006-07-18. [cit. 2006-12-03]. Dostupné z URL: <http://jcp.org/aboutJava/communityprocess/mrel/jsr218/index.html>.

[13] Sun Microsystems, Inc. *CDC FAQ* [oline]. c2006. [cit. 2006-12-03]. Dostupné z URL: <http://java.sun.com/products/cdc/faq.html>.

[14] Sun Microsystems, Inc. *J2ME Building Blocks for Mobile Device – White Paper on KVM and the Connected, Limited Device Configuration (CLDC)*. California: Palo Alto, 2000. 36 s. Dostupné z URL: <http://java.sun.com/products/cldc/wp/KVMwp.pdf>.

[15] Sun Microsystems, Inc. *The Project Monty Virtual Machine*. California: Palo Alto, 2003. 16 s. Dostupné z URL: <http://www.lbs360.net/LBSArticles/Sun.ProjectMontyWhitePaper.pdf>.

[16] Sun Microsystems, Inc. *JSR-000037 Mobile Information Device Profile (MIDP) Specification 1.0a Final Release* [online]. Poslední aktualizace 2000-12-15. [cit. 2006-12-03]. Dostupné z URL: <http://jcp.org/aboutJava/communityprocess/final/jsr037/index.html>.

[17] Sun Microsystems, Inc. *JSR-000118 Mobile Information Device Profile Specification 2.0 Final Release* [online]. Poslední aktualizace 2002-11-05. [cit. 2006-12-03]. Dostupné z URL: <http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html>.

[18] Sun Microsystems, Inc. *JSR 271: Mobile Information Device Profile 3* [online]. c2006. [cit. 2006-12-03]. Dostupné z URL: <http://jcp.org/en/jsr/detail?id=271>.

[19] Sun Microsystems, Inc. *Porting Guide, Sun Java Wireless Client Software 2.0, Java Platform, Micro Edition* [online]. Poslední aktualizace květen 2007. [cit. 2008-04-14]. Dostupné z URL: <http://java.sun.com/javame/reference/docs/sjwc-2.0-web/docs/PortingGuide-html/index.html>.

[20] Sun Microsystems, Inc. *JSR-000120 Java Wireless Messaging API Specification 1.0 Final Release* [online]. c2002. [cit. 2006-12-12]. Dostupné z URL: <http://jcp.org/aboutJava/communityprocess/final/jsr120>.

[21] Sun Microsystems, Inc. *Wireless Messaging API 2.0 Specification 2.0 Final Release* [online]. Poslední aktualizace 2004-05-17. [cit. 2006-12-12]. Dostupné z URL: <http://jcp.org/aboutJava/communityprocess/final/jsr205/index.html>.

[22] GIGUERE, E. *J2ME Optional Packages* [online]. Poslední aktualizace prosinec 2006. [cit. 2006-12-09]. Dostupné z URL: <http://developers.sun.com/techtopics/mobility/midp/articles/optional/>.

[23] ORTIZ, E., C. *The Generic Connection Framework* [online]. Poslední aktualizace srpen 2003. [cit. 2006-12-09]. Dostupné z URL: <http://developers.sun.com/techtopics/mobility/midp/articles/genericframework/>.

[24] ORTIZ, E., C. *The Wireless Messaging API 2.0* [online]. Poslední aktualizace říjen 2005. [cit. 2006-12-17]. Dostupné z URL: <http://developers.sun.com/techtopics/mobility/midp/articles/wma2/index.html>.

[25] STEUER, M. *Pokročilé JAVA aplikace pro mobilní telefony.* Brno, 2005. 31 s., 2. příl. Bakalářská práce na fakultě Elektrotechniky a komunikačních technologií Vysokého učení technického v Brně na ústavu telekomunikací. Vedoucí bakalářské práce Ing. Ivo Lattenberg, Ph.D.

[26] FUČÍK, M. *Problémy J2ME z hlediska platformové nezávislosti.* Liberec, 2006. 44 s. Bakalářská práce na fakultě Mechatroniky a mezioborových inženýrských studií Technické univerzity v Liberci na katedře aplikované informatiky. Vedoucí bakalářské práce Ing. Miroslav Holubec.

[27] KOVÁŘ, P. *Specifika vývoje aplikací pro mobilní terminály v jazyce Java* [online]. Poslední aktualizace 2006-05-02. [cit. 2006-11-24]. Dostupné z URL: <http://access.feld.cvut.cz/view.php?nazevclanku=&cisloclanku=2006050201>.

[28] KNUDSEN, J. *Understanding MIDP 2.0's Security Architecture* [online]. Poslední aktualizace únor 2003. [cit. 2008-04-25]. Dostupné z URL: <http://developers.sun.com/mobility/midp/articles/permissions/>

[29] *Java Security Domains* [online]. Poslední aktualizace 2008-04-07. [cit. 2008-05-01]. Dostupné z URL: < http://wiki.forum.nokia.com/index.php/Java_Security_Domains>

[30] HODEK, V. *Programujeme v J2ME (8.): RMS.* [online]. Poslední aktualizace 2004-03-22. [cit. 2008-04-14]. Dostupné z URL: <http://www.pcsvet.cz/art/article.php?id=4769>.

[31] GIGUERE, E. *Databases and MIDP, Part 1: Understanding the Record Management System.* [online]. Poslední aktualizace únor 2004. [cit. 2006-04-14]. Dostupné z URL: <http://developers.sun.com/mobility/midp/articles/databaserms/>.

[32] Sony Ericsson Mobile Communications AB, Java ME Platform CLDC MIDP 2 Developers' Guidelines [online]. Poslední aktualizace duben 2008. [cit. 2008-05-10]. Dostupné z URL: <http://developer.sonyericsson.com/getDocument.do?docId=65067>.

[33] *MIDP 2.1 API access rights* [online]. Poslední aktualizace 2007-06-16. [cit. 2008-05-10]. Dostupné z URL: <http://wiki.forum.nokia.com/index.php/MIDP_2.1_API_access_rights>.