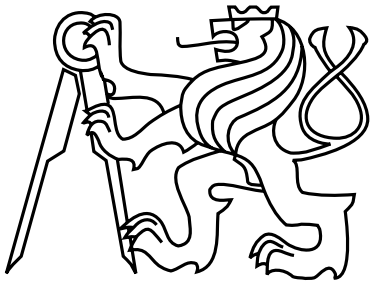
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Aplikace pro zkoušení slovní zásoby pro mobilní telefony

Rudolf Brožek

Vedoucí práce: Ing. Pavel Kubalík, Ph.D.

Studijní program: Elektrotechnika a informatika, strukturovaný, Bakalářský

Obor: Výpočetní technika

7. července 2009

Poděkování

Na tomto místě bych rád poděkoval mému vedoucímu práce, panu Ing. Pavlovi Kubalíkovi, Ph.D., za vedení a cenné rady a připomínky v průběhu vzniku mé práce. Dále bych chtěl poděkovat rodině a všem blízkým za trpělivost a podporu během celého období studia bakalářského programu na ČVUT.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze 2. 7. 2009 .....

Abstract

The aim of this work is to develop an application for testing and learning the English vocabulary. This application should build on existing solution Vocab Pro, which is targeted at personal computers running Windows. Main benefit is the change of platform from PC to mobile phones. The practice has shown the need to have an application with the words constantly along with users, and thus to be able to start learning whenever and wherever. This requirement is better to be met with new application written for mobile phone, than the original one installed on the portable computer.

Abstrakt

Cílem této práce je navržení aplikace pro učení a zkoušení anglické slovní zásoby. Tato aplikace by měla navázat na již existující řešení Vocab Pro, které je cílené na osobní počítače se systémem Windows. Hlavním rozdílem by ale měla být změny platformy na mobilní telefony. V praxi se totiž ukázala potřeba mít aplikaci se slovíčky neustále při sobě, a tak mít možnost spustit výuku kdekoliv a kdykoliv. Tento požadavek lépe splní nová aplikace psaná pro mobilní telefon, než původní nainstalovaná v přenosném počítači.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

V současné době dochází k neustálému navyšování nároků na moderního člověka a jedním z takových je schopnost ovládat cizí řeči. Na našem kontinentě je pravděpodobně nejrozšířenější výukou druhého jazyka angličtina. Není se čemu divit, angličtina zasahuje do mnoha oborů a dnes si již nelze představit, že by se bez ní člověk obešel například v oblasti komunikací nebo výpočetní techniky. České ekvivalenty bývají často nesrozumitelné, matoucí nebo vůbec neexistují a proto je znalost anglického jazyka například jedním z nezbytných předpokladů pro úspěšné splnění některých předmětů na vysokých školách.

Osvojení cizího jazyka ale nemusí být pro každého lehká záležitost a zvládnutím gramatiky tento proces rozhodně nekončí. Existuje mnoho učebnic, které dokáží gramatiku názorně vysvětlit a naučit, ale druhá část znalosti cizího jazyka je naučení slovíček které obsahuje. Tuto část lze s učebnicí zvládnout jen částečně, protože obsahuje pouze omezené množství slovní zásoby, kterou autoři zvolili jako nejvíce potřebnou pro porozumění. Slovíčka odborných textů z různých oborů se v nich hledají jen ztěžka.

Zde přichází na řadu neustálé učení neznámých slovíček, se kterými během života přijdete do styku. Určitě znáte papírkovou metodu, kdy se na jednu stranu napíše anglický význam a na druhou český překlad. Je to velice stará metoda, při které se procházejí lístečky z jednoho balíku a naučená slovíčka přesouvají do druhého balíku. Nenaučená slovíčka zůstávají v původním balíku do té doby, než se je naučíte. Je to velice jednoduchá a přitom efektivní cesta, jak se naučit dodatečnou slovní zásobu. Nešlo by ale využít dostupných moderních technologií a tuto osvědčenou metodu posunout trochu dál? Zde přichází na řadu má bakalářská práce, jejíž cílem je vytvoření aplikace, těžící právě z této metody. V současné době existuje podobná aplikace pro osobní počítače se systémem Windows, s vlastním formátem slovníků. Tato aplikace je výsledkem bakalářské práce pana Bc. Cinibulka z roku 2007 [2] a již pro ni mnoho slovníků existuje. Obrovskou výhodou papírkové metody je ale právě její „přenositelnost“ a proto bude má aplikace cílená právě na mobilní přístroje. Díky tomu bude mít uživatel možnost učení například při cestě do zaměstnání, při trávení dovolené nebo v jakémkoli jiném vhodném čase.

Kapitola 2

Popis problému, specifikace cíle

Tento text je členěn do několika kapitol. Kapitola 1 obsahuje úvod do problematiky. V kapitole 2 Vás seznámím s řešeným problémem, cíli mé práce a požadavky na implementovaný systém. V kapitole 3 naleznete rešerši již existujících aplikací, které jsem na internetu našel. U každé aplikace naleznete krátký rozbor jejích výhod a nevýhod a pro představu i pár obrázků z uživatelského rozhraní. Poté následuje kapitola 4 věnující se analýze a návrhu implementace. Zde se pokusím specifikovat požadavky na výslednou aplikaci a rozebrat potřebné úkony pro vznik aplikace. V kapitole 5 se snažím popsat výslednou realizaci aplikace. Kapitola 6 je věnovaná výsledkům testování aplikace a různým parametrům zjištěných z běhu. Naleznete zde také její porovnání s aplikacemi z rešerše. V 7. kapitole zhodnotím splnění vytyčených cílů.

V přílohách naleznete materiály, ke kterým se často v tomto textu odkazuji. Doporučuji přečíst kapitolu C ve které naleznete instalační a uživatelskou příručku. Z ní si můžete udělat celkový přehled o chování a vzhledu aplikace.

2.1 Specifikace

Cílem práce je prostudovat existující anglické slovníky pro mobilní telefony a na základě jejich analýzy navrhnout vlastní aplikaci zaměřenou na zkoušení a učení anglické slovní zásoby. Výsledné požadavky na aplikaci jsou následující:

- Aplikace bude kompatibilní s mobilními telefony podporující jazyk Java ME.
- Aplikace povede statistiku úspěšného zodpovězení jednotlivých slovíček.
- Aplikace umožní zobrazení podrobné statistiky pro jednotlivá slovíčka.
- Aplikace umožní učení a zkoušení slovíček s využitím statistiky úspěšnosti.
- Zkoušení bude probíhat formou testu.
- Aplikace umožní přidávat a editovat slovíčka.
- Aplikace umožní import slovíček z textového souboru.

- Aplikace bude kompatibilní s Vocab Pro.

Vzhledem k omezeným možnostem dnešních mobilních telefonů by aplikace neměla být příliš složitá a náročná na výpočetní výkon. Grafické rozhraní aplikace by také mělo být jednoduché a co možná nejvíc intuitivní. Ovládání aplikace je navíc omezeno pouze na pár tlačítek a funkčních prvků, které nemusí být dostupné na všech zařízení.

Kapitola 3

Rešerše stávajících řešení

Na internetu je v současné době k dostání minimálně pět aplikací, které lze použít za účelem učení a zkoušení slovní zásoby. Bohužel polovina z nich je placená a druhá polovina nefunkční. Testování proto proběhlo na následujících aplikacích:

- **Live Teacher** firmy Utisoft, komerční aplikace dostupná v TRIAL verzi.
- **Verbs** od autora Lukáše Kosíka za 20Kč.

Další dostupné aplikace

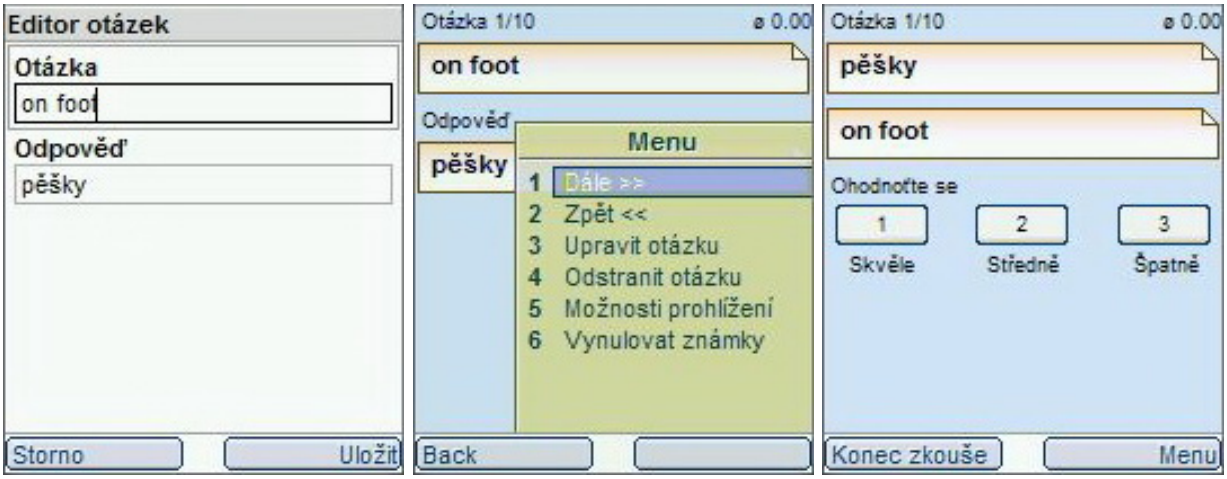
- **EnglishDrill**, komerční aplikace firmy Yoctosoft.
- **Vocabulary Trainer**, freeware od autora Christophera Lofflera.
- **MobVoc**, opuštěný open-source projekt.

Testování proběhlo na emulátoru MidpX pro Windows XP firmy Kwyshell a emulátoru MIDlet Manager pro Windows Mobile. Emulátor MidpX byl použit pro zachycení obrazovek uživatelského rozhraní jednotlivých programů.

3.0.1 Live Teacher

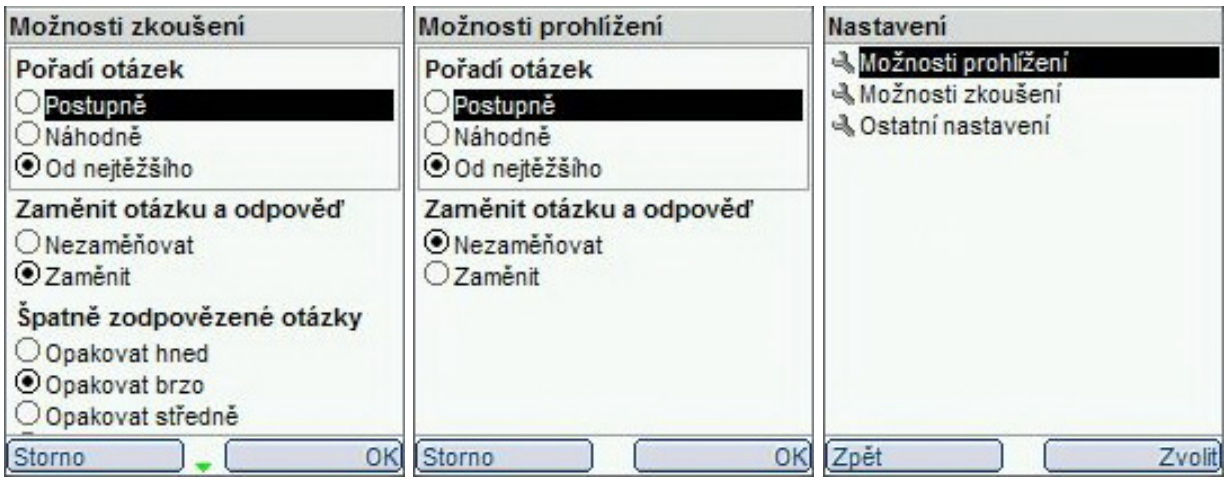
Aplikace **Live Teacher** se mi líbila z vybrané dvojice asi nejvíce. Umožňuje rozdělení slovíček na jednotlivé kategorie a tím šetří pamětí zařízení. Zkoušení poté obsahuje slovíčka pouze ze zvolené kategorie. Zkoušení probíhá jednoduchým způsobem - nejprve se uživateli ukáže slovíčko a po kliknutí na tlačítko OK se zobrazí správný překlad. Uživatel poté musí aplikaci zadat s jak velkou přesností slovíčko věděl. Na výběr jsou možnosti výborně, průměrně a špatně. Aplikace pak dokáže pro jednotlivá slovíčka vést statistiku úspěšnosti zodpovězení. V nastavení zkoušení se tato statistika dá využít pro pořadí zkoušených slovíček. Špatně zodpovězená slovíčka se mohou zobrazovat jako první a četnost jejich opakování se dá odstupňovat po 5 stupních. Bohužel jiný způsob zkoušení aplikace neumožňuje. Uživatel tak může podvědomě "podvádět"a tvrdit aplikaci, že slovíčko věděl, nebo věděl alespoň přibližně. Jako alternativu bych viděl zkoušení způsobem testu. Uživateli se zobrazí slovíčko spolu s

Obrázek 3.1: Live Teacher - Manažer témat



Obrázek 3.2: Live Teacher - Manažer slovíček

několika náhodnými variantami ze slovníku. Pro ztížení by se mohla jako poslední varianta zobrazit možnost "jiná odpověď". Další způsob zkoušení je ukázat slovíčko a přinutit uživatele napsat jeho překlad. Tato možnost je ale často nespolehlivá a při sebemenším rozdílu překladu se odpověď vyhodnotí jako špatná. Aplikace také nabízí jednoduchou možnost, jak vytvořit novou kategorii, přidání nového slovíčka do otevřené kategorie, či editaci již existujícího slovíčka. Popřípadě je možné stáhnout již hotové kategorie z internetu. U zkoušení je možné nastavit, zda se má ukazovat nejdříve anglické, nebo české slovíčko. Bohužel databáze aplikace neumí uložit ke slovíčkům více významů a chybí vypsání podrobnějších informací o úspěšnosti jednotlivých slovíček. Uživateli je nabídnuto více jazykových rozhraní, přesněji angličtina, němčina a čeština.



Obrázek 3.3: Live Teacher - Nastavení aplikace

Výhody / nevýhody:

+ Příjemné ovládání
+ Rozdělení do kategorií
+ Editor kategorií a slovíček
+ Podrobné nastavení zkoušení
+ Statistika úspěšnosti
+ Vícejazyčnost
- Aplikace je placená
- Pouze jedna metoda zkoušení
- Statistiku úspěšnosti nelze zobrazit
- Pouze jeden význam na slovíčko
- Bez nápovědy

3.0.2 Verbs

Aplikaci **Verbs** jsem do mého testu zařadil i přesto, že v základu umožňuje pouze učení a zkoušení sloves. Princip zkoušení je totiž velice podobný, je potřeba vyplnit pouze více políček. Aplikace sice umí víc než je pouhé zkoušení sloves, bohužel pro ostatní testy je zapotřebí internetové připojení přímo v mobilním přístroji. Po ukončení aplikace se stažené testy neuloží, a to mi přijde jako veliká škoda. Testy jsou totiž velice pěkně udělané a značně se podobají cvičením z učebnic. Dále je možné připojit se ke vzdálenému slovníku a gramatice, opět ale vše jen přes internet. Zkoušení sloves z lokální databáze probíhá tak, že se ukáže infinitiv slovesa a uživatel je poté vyzván k vyplnění minulého času a minulého příčestí. Po vyplnění a potvrzení slovíčka se ukáže i český překlad. Po konci zkoušení je uživateli zobrazena statistika. Bohužel statistika odpovědí se neukládá a není ani využita při plánování zkoušení. Jako další nevýhodu aplikace vidím nemožnost vytvoření vlastního slovníku, ani přidání a úpravu slovíček.

Výhody / nevýhody:

+ Obsahuje slovník
+ Velice pěkně udělané testy gramatiky
+ Statistika
- Všechny extra funkce vyžadují připojení k internetu
- Statistika se nevztahuje k jednotlivým slovíčkům
- Statistika se neukládá
- Nelze vytvářet vlastní slovníky
- Bez nápovědy

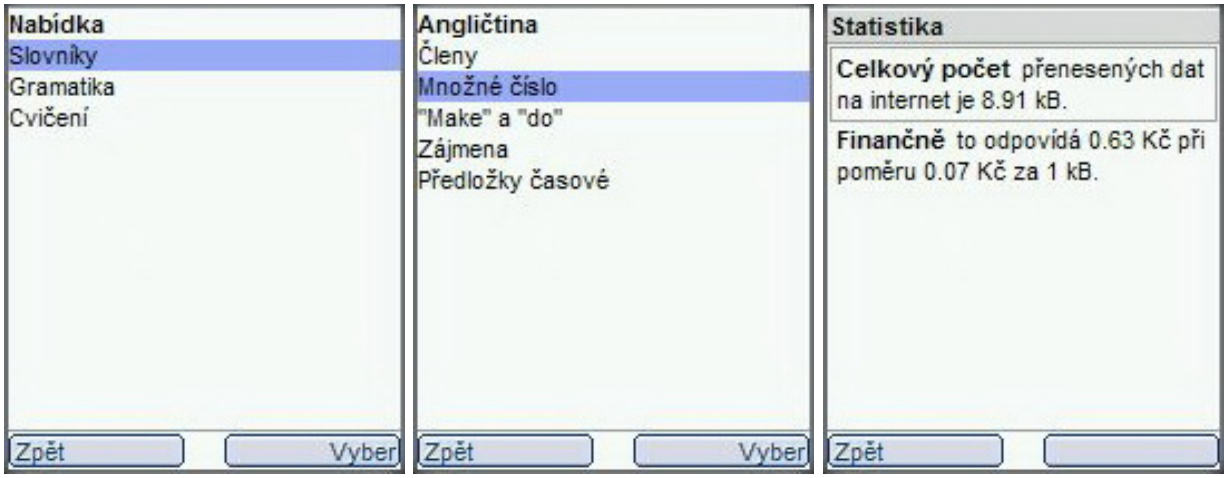
3.0.3 Vocabulary Trainer

Aplikace **Vocabulary Trainer** je psaná pouze pro přístroje Siemens SE45 a tak na jiných zařízeních nejde v nejlepším případě nahrát slovník. Slovník nejde v aplikaci ani vytvořit, musí se dodat zvlášť v textovém souboru do složky uživatele. Podle struktury tohoto souboru usuzuji, že nejde použít více významových slovíček. Zkoušení může probíhat v náhodném pořadí a dá se zvolit, jaký překlad slovíčka se má ukázat jako první, zda anglický, nebo český.

Výhody / nevýhody:

+ Aplikace je zdarma
- Kompatibilita
- Bez editoru slovníku
- Pouze jeden význam na slovíčko

Obrázek 3.4: Verbs - Menu aplikace



Obrázek 3.5: Verbs - Rozšíření aplikace z internetu

3.0.4 MobVoc

U aplikace **MobVoc** je pravděpodobně pozastavený vývoj, jelikož poslední verze je označena jako 1. Alfa verze a byla vydána v lednu 2008. Tuto aplikaci se mi nepovedlo spustit.

Kapitola 4

Analýza a návrh implementace

Pokud chceme navrhnout mobilní aplikaci, musíme se nejprve rozhodnout pod jakou platformou budeme chtít aby běžela. Proto bych na začátku této kapitoly rád napsal takový jednoduchý úvod do platforem dostupných v současné době. Zaměřím se především na platformu JavaME, která byla jasným favoritem díky masivnímu rozšíření a která má možnost běžet i na některých dalších platformách.

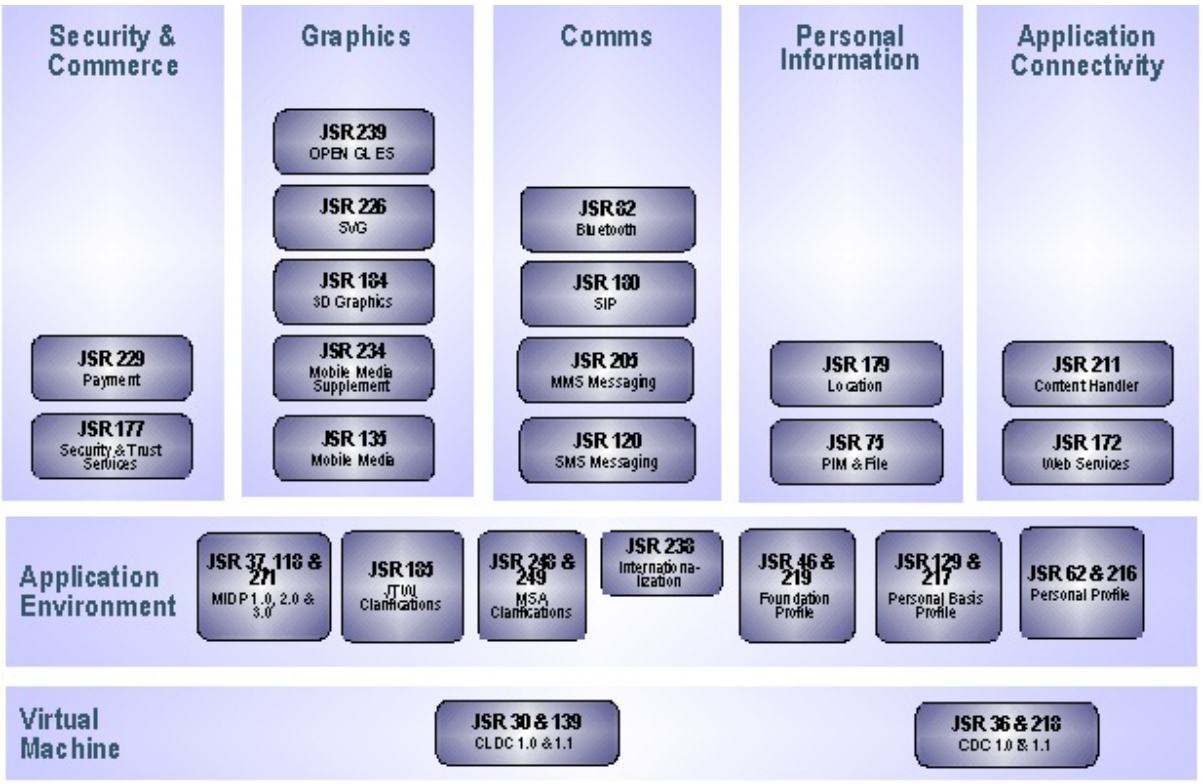
4.1 Platforma Java ME

Rychle se rozvíjející platforma[1] Java Micro Edition [7] je dílem společnosti SUN Microsystems a vychází z programovacího jazyka Oak[2]. Java ME poskytuje funkcionalitu JDK 1.1 , ale je optimalizována pro malé přístroje s omezeným výpočetním výkonem. Bohužel, na rozdíl od oblasti stolních PC kde dominuje JavaSE, je oblast malých zařízení velice rozmanitá (viz sekce fragmentace zařízení), a proto JavaME není jediná entita, ale spíše se dá chápat jako soubor specifikací, které vždy definují pouze určitou část platformy. Programovací prostředí JavaME je tak dáno jako množina profilů pro jednotlivé zařízení. V současné době se JavaME dělí na dvě konfigurace, konfiguraci CDC a CLDC, které se dají dodatečně rozšířit o další specifikace JSR. Tyto specifikace přidávají například další funkcionality jako podpora bluetooth nebo GPS.

Pro můj účel jsem vybral konfiguraci CLDC, která je určena pro nízkoúrovňovou oblast spotřební elektroniky, typicky mobilní telefony či organizéry PDA. Skládá se z virtuálního stroje a standardní kolekce tříd, které jsou založeny na knihovnách platformy Java 2. Nad touto konfigurací se dále vyskytují profily, které určují základ aplikace. Jako nejznámější profil CLDC lze považovat profil MIDP , který doplňuje tuto konfiguraci přidáním dalších tříd, jako jsou třídy poskytující funkcionalitu pro uživatelské rozhraní nebo pro místní úložný prostor. Profil MIDP je zaměřen zejména na zařízení s malými displeji a omezené úložné prostředky a vyskytuje se zejména na mobilních telefonech.

[1] V případě JavaME mluvíme o softwarové platformě, která je teoreticky schopna běžet na více hardwarových platformách

[2] Programovací jazyk pro spotřební elektroniku jako je přenosný IR ovladač a zařízení s dotykovou LCD obrazovkou. Jazyk vychází z C++, ale zabraňuje klasickým programátorským chybám jako je chybná alokace paměti nebo nesprávné zacházení s ukazateli



Obrázek 4.1: Členění platformy Java ME dle jednotlivých specifikací [7]

4.1.1 Požadavky CLDC a MIDP

Vzhledem zaměření CLDC je téměř nemožné, aby tato konfigurace poskytovala úplný virtuální stroj schopný interpretovat kompletní sadu tříd standardní knihovny JavaSE. Obyčejný "Hello-World"pro svůj běh na platformě JSE potřebuje alokovat přes 10MB paměti , a to už by byla pro mobilní zařízení příliš veliká zátěž. Proto tato konfigurace poskytuje omezený virtuální stroj KVM[3], který pro svůj běh potřebuje pouze 128kB paměti ROM a 32kB paměti RAM. Profil MIDP tyto požadavky dále zvyšuje o 160kB paměti RAM pro haldu a vlastní implementaci MIDP. Dále jsou kladeny požadavky na displej, který musí být alespoň 96 pixelů široký a 54 pixelů vysoký. Pixely mají být přibližně čtvercové a displej musí podporovat alespoň dvě barvy. MIDP specifikace dále vyžaduje, aby zařízení obsahovalo klávesnici umožňující psát minimálně číslice 0-9, případně ekvivalenty kláves.

4.1.2 Fragmentace zařízení

Fragmentace zařízení je typická pro mobilní trh. Nejen že každý výrobce používá různě veliké displeje, paměti či různě rychlé procesory, rozdílné jsou také i virtuální stroje, které výrobce na svých zařízeních použije. K tomu ještě výrobci poskytují různé doplňky pro platformu JavaME, které mohou například poskytovat komunikaci mezi zabudovaným GPS modulem, blikáním diod nebo ovládáním vibrací. Aby byl kód co možná nejpřenosnější, poskytuje kompilátor javy jednoduchý preprocesor, který umožňuje vytvořit segmenty kódů, které jsou specifické pouze pro tu, či onu platformu. Tímto je zajištěna možnost kompilovat stejný zdrojový kód bez jediného zásahu pro různá zařízení různých výrobců.

4.1.3 MIDP aplikace

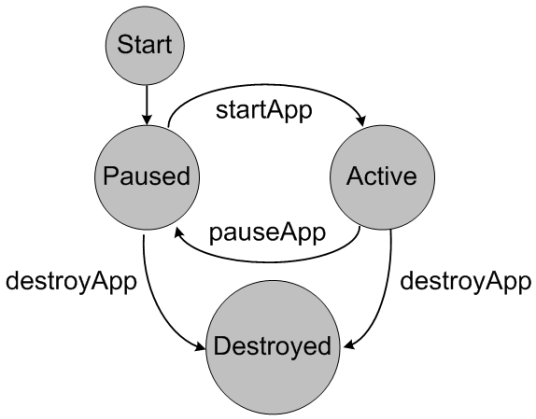
Aplikace MIDP je tvořena JAR souborem, který je podobný souboru zip. Obsahuje soubor manifest, který obsahuje dodatečné informace použitelné v aplikaci. Takovou informací může být zpráva o midletech[4], nebo vyžadovaná konfigurace a profily pro běh aplikace. Dále se v tomto souboru nacházejí třídy Javy a jiné multimediální soubory potřebné pro běh aplikace. Dalším souborem MIDP aplikace je JAD soubor, který je kopií manifestu a obsahuje pouze informace o aplikaci. Stavy MIDlet aplikace zachycuje obrázek 4.2.

4.1.4 Uživatelské rozhraní

Standardní API pro uživatelské rozhraní poskytuje balíček LCDUI, který je obsažen v konfiguraci MIDP. Toto API má velice prostý přístup k vykreslování grafiky a poskytuje pro uživatelské rozhraní pouze ty nejzákladnější objekty, jako jsou formuláře, textová pole, listy či tlačítka. Pod touto platformou existuje toolkit LWUIT [3] dostupný pod licencí GNU-GPL 2.0, který poskytuje API pro uživatelské rozhraní podobné rozhraní Swing obsažené v JavaSE. Předností LWUIT oproti SWING jsou jeho daleko menší nároky na zařízení. Díky tomu poskytuje programátorovi více možností, než základní JavaME, ale stále si zachovává

[3] Původní význam zkratky znamenal Kilobyte Virtual Machine, nedávno došlo ale ke změně na K Virtual Machine. Více o problematice KVM lze nalézt zde [15]

[4] MIDlet - spustitelná aplikace



Obrázek 4.2: Stavový diagram běhu MIDP aplikace [6]

nenáročnost na výpočetní prostředky. V základní množině dostupných funkcí jsou například velice známí správci rozvržení, jako je border layout a box layout, jednoduchý systém zpracování výjimek, či animované přechody mezi okny. Tento toolkit je schopen běžet na zařízeních podporujících CLDC1.1, MIDP2.0, CDC, PBP/SE. Pro řízení událostí a vykreslování grafiky uživatelského rozhraní se využívá modelu EDT . Všechny události a volání po vykreslení grafiky jsou řízeny pomocí jednoho vlákna. Tím je zajištěna serializace a po virtuálním stroji nejsou vyžadovány náročné mechanismy pro řízení běhu vláken. Další výhodou je snazší přenositelnost toolkitu mezi profily, které mohou mít drobné nekonzistence při práci s vlákny.

4.1.5 Trvalé úložiště

Vzhledem k tomu, že téměř všechny MIDlety vyžadují uložení informací tak, aby se mezi relacemi zachovaly, specifikace MIDP vyžaduje po všech implementací, aby poskytly trvalé úložiště. Skutečný mechanismus pro uložení informací se liší zařízení od zařízení, programové rozhraní musí být ale naprosto stejné[5]. Takovým mechanismem je úložiště záznamů implementované v MIDP. Každé takové úložiště obsahuje záznamy s unikátním identifikátorem, které jsou přístupné přes iterátor. Bohužel tato implementace trpí velikou restrikcí práv MIDletů, a tak není možné přistupovat k jiným souborům. Tento nedostatek byl odstraněn ve specifikaci MIDP 2.0, která přidává bezpečnostní model pro řízení práv MIDletů jako je tomu například u Java apletů. MIDletu se může povolit právo přístupu k souborovému systému zařízení a tím získá přístup k ostatním souborům, například na paměťové kartě. Další cestou, kterou jde načíst data do aplikace, je skrze využití JSR 75, která poskytuje API pro přístup k souborovému systému samotného zařízení. Vzhledem ke snaze zamezit rizikům zneužití přístupu k souborům, každý MIDlet musí upozornit uživatele na možnost přístupu k datům a vyžádat si od něj povolení. Implementace je bohužel dobrovolná a tak nelze zaručit, že jí bude podporovat většina mobilních přístrojů. Krajním řešením je možnost přístupu k souborům přes internet. Toto řešení by spoléhalo na to, že se veškerá potřebná data stáhnou až při běhu aplikace a v případě zájmu uloží do trvalého úložiště implementované specifikací

[5] Jinak by nebyla zajištěna přenositelnost aplikací

MIDP.

4.2 Ostatní platformy

4.2.1 Platforma Android

Tato platforma [10] je projektem nedávno vzniklé aliance OHA[6] a obsahuje softwarový balík pro mobilní zařízení, který obsahuje operační software, middleware a další důležité aplikace potřebné pro správný chod mobilního telefonu. Pro tento balík je dostupný zatím jen zkušební SDK, který poskytuje nástroje a API potřebné pro vývoj základních aplikací pro tuto platformu. Platforma běží na linuxovém jádře a poskytuje sadu knihoven odvozených z C/C++.

Dle speficikace výrobce [11] každá aplikace pro tuto platformu běží jako samostatný proces s vlastní instancí virtuálního stroje Dalvik. Tento virtuální stroj je optimalizován tak, aby jich mohlo na každém zařízení běžet více a efektivně. Virtuální stroj je zaměřen na registry a pracuje s třídami zkompilovanými java kompilerem. Virtuální stroj se spoléhá na upravené monolitické linuxové jádro, které poskytuje takové funkcionality jako jsou vlákna nebo nízko-úrovňová správa paměti. Podrobnější představu o architektuře si můžete udělat na obrázku 4.3 ze stránek výrobce [11].

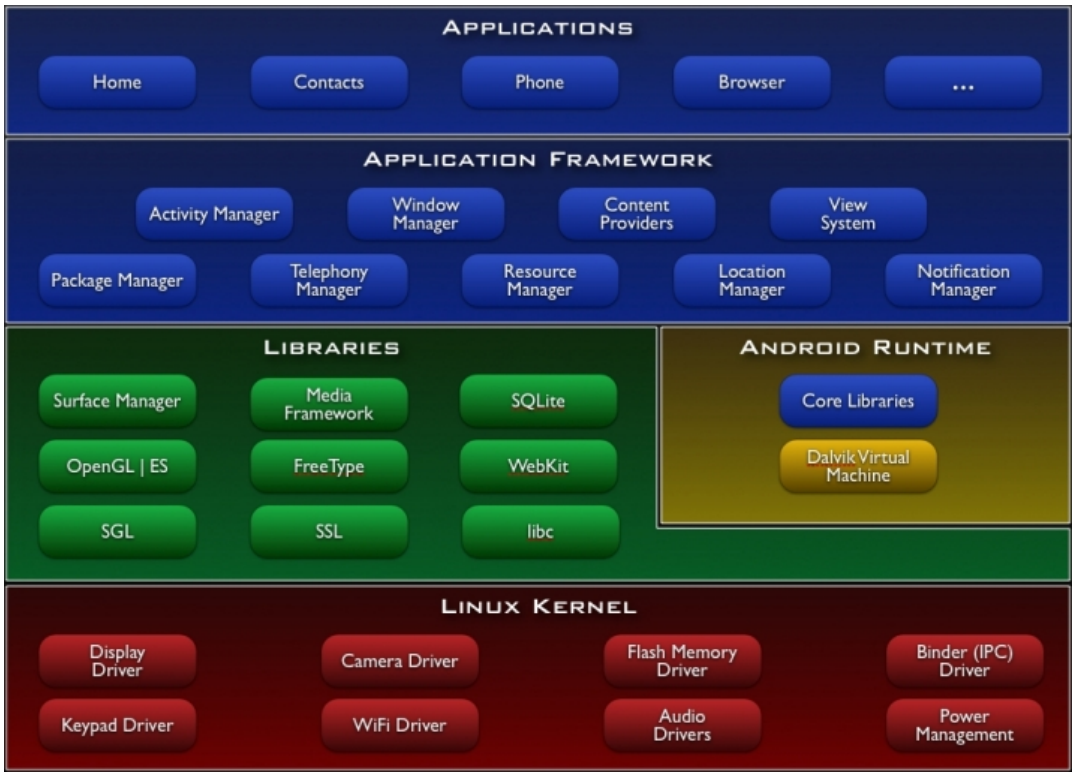
4.2.2 Platforma Symbian

Historie této platformy sahá až do roku 1998, kdy vzniklo partnerství mezi firmami Nokia, Motorola, Ericsson a Psion [8]. Výsledkem tohoto spojení vznikla nová platforma Symbian OS [9], která měla pokrýt trh mezi jednoduchými mobilními telefony a složitějšími PDA. Symbian OS částečně vychází ze staré platformy EPOC firmy Psion [8], podporuje navíc preemptivní[7] multitasking a ochranu paměti. Nativním jazykem je jazyk C++, díky kterému je dosažena veliká optimalizace aplikací, současně s tím jsou ale kladeny větší nároky na programátora, především v oblasti správy paměti a dalších problematických oblastí typických pro tento jazyk. Systém je schopný běžet pouze na zařízeních s ARM[8] architekturou. V současné době vlastní většinový podíl firma Nokia, která se rozhodla radikálním způsobem pohnout s vývojem dál a tak konkurovat Windows Mobile zařízením. Jedním z takových kroků je například postupné zveřejnění systému jako open-source, ze kterého si Nokia slibuje silnější programátorskou komunitu. Nutno ale podotknout, že Symbian OS má v současné době největší (45%) zastoupení na trhu chytrých mobilních zařízení. Dalším v pořadí je až firma Apple s 15% pokrytím.

[6] Aliance Open Handset Alliance, která obsahující firmy jako jsou Google, HTC, Motorola, Qualcomm, and T-Mobile

[7] Preemptivní plánování procesu (neboli plánování s předbíháním) je metoda, kterou využívá operační systém pro přidělování svého výpočetního času jednotlivým spuštěným procesům. Více o multitaskingu na symbian zde [12]

[8] Architektura ARM je 32 bitová mikroprocesorová RISC architektura vyvinutá firmou ARM Limited, která je široce využívaná v mnoha embedded systémech



Obrázek 4.3: Architektura platformy Android

	Symbian	Java ME	Android
GUI	2D, 3D grafika (především novější telefony), Mnoho doplňků, Visuální GUI editor	2D, 3D grafika, Mnoho doplňků, Visuální GUI editor	2D, 3D (OpenGL ES) grafika, některé doplňky, GUI editor třetích stran
Funkcionalita	Bez omezení	Liší se podle různých zařízení. Závisí na dostupnosti rozšiřujících profilů JSR. Omezení přístup k souborům.	Částečná skrze API
Přístup k datům přístroje	Plný přístup	Liší se podle různých zařízení. Závisí na dostupnosti rozšiřujících profilů – především JSR 75, nepovinného profilu PDA	Plný přístup
Rychlost běhu aplikací	Nejlepší	Průměrný (Vzhledem k běhu na VM)	Průměrný (Vzhledem k běhu na VM)
Přenositelnost	Kompilace vždy pro určité zařízení	Průměrná (VM obsahují různé chyby)	Přenositelnost vždy jen v rámci platformy

Obrázek 4.4: Shrnutí platforem

4.3 Schopnosti platforem

Pro lepší orientaci v platformách přikládám obrázek 4.4, na kterém se snažím shrnout veškeré důležité vlastnosti probraných platforem.

4.4 Vývojové prostředí

Každá platforma má svůj specifický jazyk a odlišné vývojové prostředí, ve kterém se aplikace programují. Nejrychlejší způsob, jak začít s vývojem aplikace, je opatření SDK cílené platformy od výrobce [7]. Výhodou těchto balíků je veliké množství dodatečných nástrojů podporujících vývoj a distribuci aplikace. Užitečným se ukázal emulátor mobilního zařízení, kde se dá vyzkoušet běh aplikace aniž by se musela nahrávat do mobilního telefonu. U plat-

formy JavaME je díky fragmentaci zařízení složitější, jelikož mimo oficiálního nástroje Sun Java ME Software Development Kit, které vzniklo spojením starších nástrojů Sun Java Wireless Toolkit pro CLDC a CDC, existuje také mnoho dalších SDK jednotlivých výrobců umožňující snažší vývoj aplikace v rámci jejich vlastních produktů. Jmenovitě mohu uvést například firmu Sony Ericsson, která je známá velikou podporou platformy JavaME a produktem Sony Ericsson SDK 2.5.0.3 for the Java™ ME Platform to jen potvrzuje. Jako hlavní výhodu tohoto SDK vidím těsné provázání s mobilními přístroji výrobce a tak výsledné aplikace mohou používat i jiné služby, které by jinde dostupné nebyly. Tvůrci IDE netbeans také poskytují jednoduché rozšíření, které využívá původní SDK od Sunu a rozšiřuje ho o takové prvky jako verzování, nástroj pro grafické uživatelské rozhraní, dodatečné třídy z balíku mobility a mnoho dalších prvků typických pro klasické IDE Netbeans [9]. Symbian v současné době zaštiťuje firma Nokia a poskytuje SDK vždy pro určitou řadu svých mobilních telefonů.

4.5 Požadavky na použitou platformu

Nejdůležitějším požadavkem na použitou platformu je snadná přenositelnost aplikací mezi jednotlivými zařízeními. Snazší přenositelností dosáhneme většího pokrytí mobilních telefonů schopných spuštění aplikace. Aplikace by měla být schopna běžet na běžně dostupných mobilních telefonech, které jsou mezi lidmi nejvíce rozšířené.

Dalším požadavkem je svižný běh aplikace na zvolené platformě. Aplikace si vystačí pouze s jednoduchou 2D grafikou a tak není potřeba velikého výpočetního výkonu pro uživatelské rozhraní. Limitujícím faktorem je ale maximální velikost instance programu, kterou je dané zařízení schopné zavést do paměti. Platforma by měla být schopna poskytnout aplikaci prostor pro nahrání dostatečného množství slovíček do vnitřní paměti přístroje z externího zdroje, například SD karty. Slovník o kapacitě 30 slovíček uložený v prostém textovém souboru zabere necelý 1kB, xml soubor se blíží ke 2kB V případě slovníků obsahujících přes 2000 slovíček se dostáváme k velikostem přes 40kB pro textové soubory a 50kB pro xml. Vzhledem k pomalé odezvě uživatele při zkoušení slovíček není potřeba všechna data nahrávat ihned do operační paměti. Pro náš účel postačí nahrát vždy pouze několik slovíček a postupně dohrávat další.

4.6 Úzká místa návrhu

Návrh aplikace počítá s možností přístupu k lokálním souborům daného zařízení za účelem otevření, či importu slovníků. Tuto možnost poskytuje většina platforem pro chytré telefony, bohužel platformy pro jednoduchá mobilní zařízení tuto funkcionalitu často nemají implementovanou. Toto je limitující faktor, který omezuje okruh zařízení, na kterých aplikace poběží. Jako jediná možná cesta, jak obejít tento nedostatek, je přístup k datům slovníku přes internet, který jsem viděl u některých již existujících aplikací.

[9] Velice názorná přednáška o Netbeans s rozšířením Mobility zachycuje video [14]

4.7 Časový plán a odhad množství kódu

Aplikace se bude dělit na několik balíků, balík obstarávající uživatelské rozhraní, balík pro práci se slovníkem a slovíčky a balík pro aplikační logiku. V první fázi je potřeba navrhnou třídy obsažené v balíku pro aplikační logiku a práci se slovníkem. Současně je třeba připravit jednoduché třídy i pro uživatelské rozhraní, které by mohl programátor použít pro interakci s aplikací. Uživatelské rozhraní bude v první fázi díky propojení s IDE Netbeans založené na knihovně LCDUI. Ve druhé fázi se naprogramuje výsledný balík pro uživatelské rozhraní založený na knihovně LWUIT, tedy až nakonec, po dokončení předchozích pracích.

Odhad množství kódu je dle zkušeností s podobným projektem 3000-5000 řádků. Vycházím z předpokladu, že bude potřeba přibližně 30 tříd pro uživatelské rozhraní, které vychází dle jednoduché hello world aplikace na 50 řádků. Funkci na přepínání oken tak odhaduji na 30 * 30 řádků, tj 900-1000 řádků. Aplikační logiku pro načtení a uložení slovníku odhaduji také na 1000 řádků. Dalších 1000 řádků bude obsahovat dodatečná logika aplikace. Balík model musí obsahovat třídy zastupující slovník a slovíčko se všemi getry a setry, které by mohli vycházet celkem na 500 řádků.

4.8 Výběr způsobů implementace

Pro dodržení podmínek maximální možné kompatibility bude výsledná aplikace naprogramována v programovacím jazyce JavaME, který v současné době podporuje většina nejvíce rozšířených mobilních telefonů. Jako vývojové prostředí bude, díky dobrých zkušenostem z vývoje JavaSE aplikací, použito IDE Netbeans s rozšířením mobility a SDK Sun Java Wireless 2.5.2. Starší SDK je použito vzhledem k lepší podpoře JSR 75 než v novější verzi, které je dostupné zatím jako zkušební verze. Uživatelské rozhraní aplikace bude zajišťovat knihovna LWUIT, která je poskytována pod licencí GNU-GPL 2.0 a poskytuje více možností než standardní knihovna Java Micro LCDUI.

4.9 Návrh řešení

Návrh řešení vychází ze studie již existujících aplikací, které se mi povedlo zprovoznit. Při studii jsem se zaměřil na funkce, které bych v mé aplikaci také chtěl vidět a na funkce, kterým bych se radši vyhnul. Jako největší nectnost vidím nekompatibilitu aplikací, plynoucí ze špatně zvoleného způsobu uložení dat. Taková aplikace je poté naprosto nepoužitelná a veškeré snahy o uživatelsky přívětivé prostředí jsou zbytečné. Proto se budu držet striktně specifikace MIDP 2.0, která zajistí jednoduchý mechanismus pro načtení slovníků ze souborů na paměťových kartách. Slovníky budou při prvním použití aplikací importovány a dále uloženy pomocí mechanismů specifikovaných profilem MIDP 2.0. Tímto bude zajištěna maximální možná optimalizace přístupu k uloženým informacím a přenositelnost aplikace.

Po importu slovíček bude možné vést ke každému statistiku úspěšného zodpovězení. Tato statistika bude použita při plánování zkoušení. Uživatel si tak bude moci zvolit, s jak velkou frekvencí se mají špatně zodpovězená slovíčka opakovat v průběhu zkoušení. Tento velice efektivní přístup k učení slovíček byl použit pouze u aplikace Live Teacher. Statistika půjde

zobrazit samostatně, aby uživateli poskytla přehled o tom, jak byl při zkoušení úspěšný. V případě potřeby bude mít uživatel možnost statistiku smazat a pokračovat od začátku.

Pro správu slovíček bude poskytnut jednoduchý manager, který umožní seskupovat slovíčka do různých slovníků, které budou reprezentované samostatným souborem . Dále manager umožní přidání, smazání a editování slovíček v již uložených slovnících. Kapacita takto vytvořených slovníků bude omezena pouze velikostí paměti daného zařízení.

Zkoušení slovíček bude probíhat dvěma způsoby. Prvním způsobem je forma testu, kdy aplikace ke zkoušenému slovíčku načte z právě otevřeného slovníků další významy jiných slovíček, a nabídne je uživateli spolu se správnou variantou. Druhou možností je zobrazení zkoušeného slovíčka a po určité prodlevě, nebo reakci uživatele, se promítne jeho překlad. Zajímavou možností, jak uživatele zkoušet v průběhu dne, je běh aplikace na pozadí a po určité prodlevě na sebe upozornit například vibrací, nebo zvukovým signálem a vyzkoušet uživatele z dalšího slovíčka.

Uživatelské rozhraní musí být jednoduché a vzhledem k malým displejům cílených zařízení se musí počítat pouze s jedním oknem na obrazovku. Tato okna se pak budou měnit v závislosti na postupu uživatele aplikací. Tento přístup byl ostatně použit ve všech testovaných aplikací. Předností bude eliminace možných scroll barů, které sice umožňují větší okna na obrazovku, ale znesnadňují uživateli jeho práci s aplikací. Jako nejlepší možné API pro uživatelské rozhraní jsem zvolil LWUIT, které zprostředkuje nejen pěknou grafiku aplikace, ale také jednoduché ovládání typické pro dnešní mobilní aplikace.

Jako velikou výhodu použití tohoto toolkitu pro mou aplikaci vidím v minimalizaci problémů vykreslování uživatelského rozhraní, způsobených již zmíněnou fragmentací mobilních zařízení. Pokud je záměrem co možná největší přenositelnost aplikace, musím počítat jak s displeji 100x100 pixelů, tak i s displeji překračující rozlišení QVGA či VGA. API je navíc velice podobné SWING z platformy JavaSE. Jako další přednost vidím poskytnutý jednoduchý mechanismus pro záznam a výpis logu z běhu aplikace. Uživateli je navíc poskytnuta velmi jednoduchá cesta jak si přizpůsobit vzhled aplikace svým potřebám pomocí různých schémat a možnost ovládat aplikaci pomocí dotykového displeje.

Kapitola 5

Realizace

V této kapitole bych Vás rád seznámil s výsledkem svojí práce a podrobně rozebral její zajímavější stránky, jako je například kontrola syntaxe slovíček při načítání textových slovníků nebo výhody využití knihovny LWUIT. Často se zde odkazuji na obrázky s UML diagramy které naleznete v příloze. Případné zájemce o širší rozebrání zdrojového kódu aplikace odkazuji na vygenerovanou dokumentaci javadoc, kterou naleznete na přiloženém CD. Tato dokumentace totiž vznikala přímo při jeho psaní a tak je s ním velice těsně provázána.

5.1 Členění aplikace

Aplikaci jsem se snažil navrhnout podle vzoru MVC a tak rozdělit kód na datový model, řídící logiku a nakonec uživatelské rozhraní aplikace. Zdrojové kódy tak obsahují několik samostatných balíků, které obsahují další třídy poskytující aplikaci potřebné funkce. Diagram, zachycující balíky, naleznete na obrázku B.1, v příloze B.

5.1.1 Balík model

Tento balík poskytuje třídy starající se o datový model aplikace, do kterého patří slovíčka, slovníky a jejich statistika nebo různá nastavení aplikace. Diagram na obrázku B.2 zachycuje třídy z tohoto balíku a jejich vztahy. Třída ApplicationModel poskytuje singleton selfModel, který se stará jak o správné zacházení se slovníky, slovíčky i statistikou, tak i o vlastní nastavení aplikace. Jsou zde například funkce pro čtení dodatečných dat knihovny LWUIT, nebo funkce pro práci se slovníkem. Dále funkce kontrolující stav vyzkoušených nebo naučených slovíček, funkce pro zjištění významů nebo statistiky slovíček. Programový vzor singleton zde zajišťuje snadnou dostupnost modelu z jiných částí aplikace a zaručuje jeho jedinečnost.

Další třída v tomto balíku, třída Dictionary, má na starost zejména spolupráci s vnitřním úložištěm aplikace, které spravuje RMS. Do tohoto prostoru se ukládá vlastní nastavení a popřípadě i současně otevřený slovník. Tato třída je mu nejblíž a přístup k němu by měl být prováděn vždy pouze zde. Tím se zabrání porušení dat nebo jiným konfliktům, které by mohly vzniknout. Obsahuje funkce pro otevření nebo uložení slovníku, vytváření iterátoru pro slovíčka v úložišti a funkce pro práci se slovíčky. Pro zkoušení se využívá funkcí, které vrací náhodné slovíčko nebo náhodné významy slovíček obsažených ve slovníku. Současně s

```
public CopyClone getCopyClone() {
    if (copyClone == null) {
        copyClone = new CopyClone((String)
        model.ApplicationModel.selfModel.getTranslation("f_copyClone"));
    }
    return copyClone;
}
```

Obrázek 5.1: Funkce pro vytvoření třídy UI

touto třídou je zde třída DictionaryData, která zastupuje statistiku pro slovíčka uložená ve slovníku.

Velice důležitou třídou je třída Word, která poskytuje funkce především pro konverzi pole bajtů načtených z úložiště na rozumnější formu, se kterou pak aplikace dál pracuje.

Třída Settings zastupuje uživatelské nastavení aplikace a obsahuje funkce pro jeho načtení, uložení nebo obnovení původního nastavení.

5.1.2 Balík view

Uživatelské rozhraní aplikace je schované pod balíkem view, který poskytuje třídu FormFactory, starající se o vytváření uživatelského rozhraní. Třída FormFactory je taktéž psaná dle vzoru singleton. Na obrázku B.3 vidíte, že kromě třídy FormFactory jsou zde obsaženy další balíky Forms a Dialogs. Zde jsou uloženy interface a konečné třídy, které jsou určeny k zobrazení na displeji. FormFactory využívá metody takzvané pomalé inicializace[1], typické pro programový vzor Factory. Tento vzor je všeobecně doporučovaný pro JavaME aplikace. V případě potřeby je zde metoda dispose() pro zneplatnění většiny vytvořených tříd, které jsou tak určeny ke smazání při dalším průchodu garbage kolektorem. Také je zde schovaná třída Progress dodatečně dopsaná autory knihovny LWUIT pro grafické zobrazení průběhu. Sekvenční diagram B.7 zachycuje tvorbu hlavního menu. Vidíme zde volání funkce getTranslation nad modelem, která vrací jméno okna dle nastaveného jazykového prostředí a dále již volání konstruktoru třídy MainMenu. Třídy balíku Forms dědí od třídy Form knihovny LWUIT a dají se přirovnat k JFrame u swingu. Jsou to takové kontejnery pro další grafické prvky, které jsou určeny k zobrazení na celé obrazovce. Balík Dialogs naopak obsahuje třídy dědící třídu Dialog, které plní funkci vyskakujících dialogů. Příklad pomalé inicializace zobrazuje například funkce 5.1.

5.1.3 Balík Controller

Posledním balíkem je balík controller, který obsahuje třídy starající se o vlastní chod aplikace (viz. obrázek B.4). Tento balík dle specifikace obsahuje třídu MIDlet, která je zavedena při startu aplikace a stará se o nastartování mé aplikace. Inicializuje grafickou knihovnu LWUIT a poté řídící část. Řídící část je v třídě ApplicationController, která poskytuje další singleton controllerSelf. Hlavní část třídy tvoří funkce traverse, která na základě vstupů a současně

[1] Tento postup je doporučen v kurzu [4] nebo v předmětu [1]

```
\begin{verbatim}
    public void traverse(int i_from, int i_where) {

        switch (i_from) {

            case 1:
            ...
```

Obrázek 5.2: Funkce traverse

```
   public static short readShortLittleEndian(InputStream is)
       throws IOException {
       int low = is.read();
       int high = is.read();
       return(short)( high << 8 | low );
    }
```

Obrázek 5.3: Funkce třídy LittleIndianConversion

zobrazeného okna rozhoduje o dalším okně, na které se přejde. Její hlavičku a část těla funkce je uvedena na příkladu 5.2. Podle parametru i_from, který jí předá každé okno při potřebě přesunutí na další okno, a podle parametru i_where dojde k vygenerování dalšího okna, které se následně zobrazí. V případě potřeby může dojít k dalším akcím v aplikační logice jako například otevření slovníku nebo uložení nastavení.

Zde vidíte Další zajímavou třídou z tohoto balíku je třída Parser, která se stará o přístup aplikace k paměťové kartě a obsahuje syntaktický analyzátor pro načtení slovníku v textové formě, který rozeberu podrobněji v podkapitole Syntaktický analyzátor. Přístup k paměťové kartě využívají funkce pro otevření nebo export slovníku a je založen na JSR 75.

Dále se zde nachází třída LittleIndianConversion, poskytující funkce umožňující konverzi informací uložených ve tvaru little-endian. Jsou důležité zejména při načítání slovníků uložených aplikací Vocab Pro. Důležité je zde poznamenat, že tato aplikace je psaná v jazyce C++ a formát uložených slovníků se může lišit, zejména pokud by byly uložené aplikací zkompilovanou pro jinou platformu. Vocab Pro je ale těsně provázán s operačním systémem Windows knihovnou MFC a tak problémy s jiným formátem dat ve slovnících nepředpokládám. Funkce na příkladu 5.3 slouží pro načtení 2B short hodnoty.

Dodatečně byla přidána třída ThreadManager, která slouží ke spuštění déle trvajících operací v samostatném vlákně. Při výsledcích testování aplikace s většími slovníky totiž docházelo k nepříjemným prodlevám, které se jevily jako klasické zaseknutí. Proto dochází k oznámení činnosti s jednoduchou obrazovkou indikující průběh a vykonání delších operací v dalším vlákně. Toto řešení je však na úkor jednoduchosti aplikace a přidává nároky na prostředky mobilního telefonu. V praxi se ověřilo, že načítání stejného slovníku v původní verzi aplikace trvalo kratší dobu, než ve verzi s více vlákny.

(1) S → DICTLINE
(2) DICTLINE → WORLDS [PRON] = MEANINGS DICLINE | ϵ
(3) WORLDS → WORLDS_STD
(4) PRON → WORLDS_STD | WORLDS_PRON
(5) MEANINGS → WORLDS_STD | WORLDS_MEAN
(6) WORLDS_STD → pismenoWORLDS_STD2
(7) WORLDS_STD2 → pismenoWORLDS_STD2 | ϵ
(8) WORLDS_PRON → pismenoWORLDS_PRON2 | specPismenoWORLDS_PRON2
(9) WORLDS_PRON2 → pismenoWORLDS_PRON2 | specPismenoWORLDS_PRON2 | ϵ
(10) WORLDS_MEAN → pismenoWORLDS_MEAN2 | ceskePismenoWORLDS_PRON2
(11) WORLDS_MEAN2 → pismenoWORLDS_MEAN2 | ceskePismenoWORLDS_PRON2 | ϵ

Tabulka 5.1: Gramatika syntaktického analyzátoru

ʃ ʒ θ ð ŋ ɹ ʍ ɬ ʔ ː \ ə æ

Obrázek 5.4: Podporované symboly mezinárodní fonetické abecedy

5.2 Syntaktický analyzátor

Již zmíněný syntaktický analyzátor se nachází v balíků Controller ve třídě Parser a slouží ke kontrole syntaxe slovníků uložených v textové formě. Část z gramatiky je uvedená v tabulce 5.1.

Z této tabulky můžeme vyčíst, jak syntaktický analyzátor pracuje při načítání a kontrole slovníků z textových souborů. Podle pravidla (1) nejprve zjistí, zda je ve slovníku nějaké slovíčko (řádek v textovém souboru) a pokud ano, zkouší na tento řádek naroubovat pravidlo (2). Toto pravidlo říká, že každé slovíčko musí být složené z tokenů WORLDS PRON a MEANINGS a symbolů **[**, **]**, = a **;**. Při dalším postupu v pravidlech gramatiky zjistíme, že token WORLDS zastupuje anglický význam slovíčka. Může se tak skládat pouze ze standardních symbolů abecedy. Token PRON zastupuje fonetický přepis slovíčka a tak musí být obohacen o symboly mezinárodní fonetické abecedy - IPA. Přehled podporovaných symbolů naleznete na obrázku 5.4. Poslední token MEANINGS zastupuje český význam slovíčka a proto byl obohacen o znaky české abecedy.

Tato gramatika se ale časem ukázala jako příliš přísná a tak došlo k poměrně radikálnímu zmírnění pravidel. Především byly povoleny dodatečné znaky pro cizí i český význam slovíčka tak, aby uživatel mohl zadat složitější slovíčka. Dále došlo ke zrušení omezení počtu znaků na slovíčko.

Na příkladu první funkce 5.5, která kontroluje začátek řádku, můžeme vyčíst jak analyzátor pracuje, popřípadě jak informuje o chybě v syntaxi.

5.3 Práce se slovníky

Aplikace pro přístup ke slovníkům používá částečně třídy parser a třídy dictionary. Třída parser využívá funkcí definovaných ve specifikaci JSR 75 - PIM & File, která zaručuje maximální možnou přenositelnost mezi mobilními telefony, které tuto specifikaci podporují.

```
private void grammer_line1(int startToken)
    throws SyntaxException, IOException {
    line++;
    progress = new Vector();
    actToken = startToken;
    switch (actToken) {
        case 6:
            grammer_word1();
            grammer_line2();
            break;
        default:
            sex = new SyntaxException("Syntax exception");
            sex.setProgress(progress);
            sex.setExceptionLine(line);
            sex.addExpectedToken(6);
            sex.setInvalidToken(actToken);
            throw sex;
    }
}
```

Obrázek 5.5: Funkce syntaktického analyzátoru

Bohužel jsou zde patrné veliké rozdíly v implementaci VM od různých výrobců a tak se rychlost přístupu k souborům velice liší. Dále se zde vyskytují nepříjemné dotazy VM, zda uživatel chce povolit přístup k souboru, které často nejdou běžnou cestou vypnout. Proto jsem se rozhodl, že veškerý slovník napoprvé načtu celý a ke zdrojovému souboru se přistoupí, až když bude potřeba uložit změny buď ve slovníku, nebo ve statistice. Takto by ale šlo pracovat pouze se slovníky s omezeným počtem slovíček a tak využívám správce lokálního úložiště (RMS), který je obsažen v profilu MIDP. Správce poskytuje úložný prostor někde v telefonu. Slovíčko „někde“ je zde velice důležité, protože aplikace neví kam přesně se data ukládají. Mohou být uložena někde na paměťové kartě nebo v zabudované paměti telefonu. O všem totiž rozhoduje výrobce a jeho implementace VM. Pro nás je důležité, že máme prostor kam naše data ukládat a metody, jak k nim efektivně přistupovat. Efektivní přístup zajišťuje správce poskytnutím iterátoru pro úložiště. Zde přichází na řadu třídy Dictionary a DictionaryData, které tohoto správce využívají. Třída Dictionary zajišťuje funkce pro práci se slovíčky slovníku a třída DictionaryData zase funkce pro práci se statistikou slovníku.

5.4 Správce slovníků

První implementace aplikace obsahovala pokročilejšího správce slovníků, který umožnil uživateli doslova „naimportovat“ více slovníků a vždy si vybrat, ze kterého slovníku chce zkoušet. Úmysl byl takový, že si uživatel postupně během používání aplikace naimportuje nebo vytvoří více slovníku s méně slovíčky a poté se nechá postupně zkoušet z jednotlivých slovníků, které by mohly být rozdělené například jako lekce v učebnicích. V aplikaci pak byla položka

Obrázek 5.6: Správce slovníků v grafické knihovně LCDUI

Moje slovníky ze které se museli provádět úkony jako zkoušení nebo učení, kterou můžete vidět na obrázku 5.6.

Tato funkce se ale v praxi ukázala jako problematická a vnášela do ovládání programu další prvky, které uživatele mátly. Proto došlo k radikálnímu zjednodušení správce tak, aby umožnil otevření vždy pouze jednoho slovníku, se kterým by uživatel mohl pracovat. Veškeré funkce jsou pak seskupené do pod obrazovku **Akce se slovníkem** (viz. obrázek 5.7)

Tím je ovládání maximálně zjednodušené, ale vyšel na povrch problém s obsáhlejšími slovníky. Při testování se totiž ukázalo, že správce lokálního úložiště má s přibývajícím počtem slovíček problém s generováním iterátoru pro jejich procházení. Tento problém je ještě patrnější, pokud uživatel chce slovíčka třídit a iterátor se tak generuje na základě například textového porovnání významů slovíček. Počet slovíček, kdy už správce vykazuje nepříjemně dlouhé prodlevy při práci s iterátory, se liší výrobce od výrobce, ale v praxi se ukazuje, že tento počet začíná být problematický přibližně od 300-500 slovíček na slovník.

Původní správce z obrázku 5.6 byl používán ještě před změnou grafické knihovny na LWUIT a tak zde vidíte vzhled původní aplikace využívající standardní LCDUI knihovnu.

5.5 Využití knihovny LWUIT

Aplikace ze začátku využívala grafickou knihovnu LCDUI, která sloužila zejména pro ověření funkčnosti. Výhodou byla propojenost knihovny s IDE Netbeans a tak se uživatelské prostředí dalo navrhnout velice rychle. Díky programovému vzoru MVC bylo ale velice snadné přejít na knihovna LWUIT[5], která nabízí především:

- Rozhraní a vzhled podobný swing aplikacím

Obrázek 5.7: Nové pojetí práce se slovníky

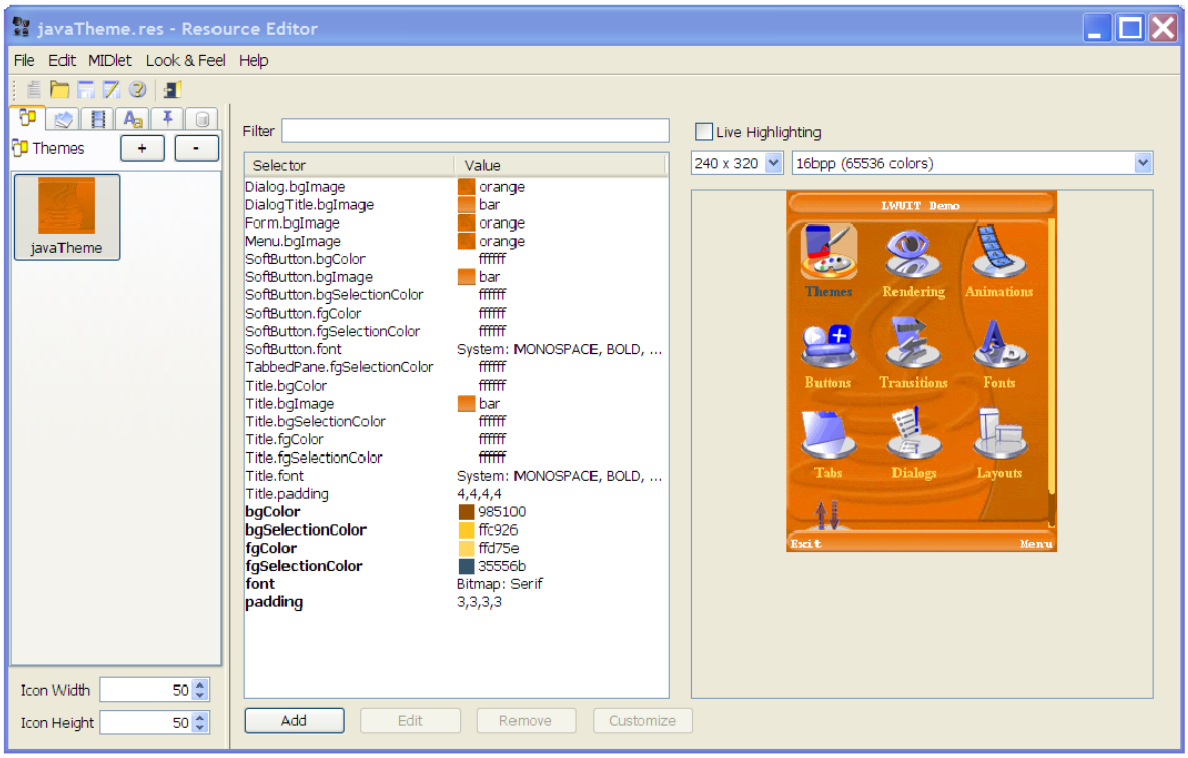
- Snadná skinovatelnost aplikace
- Rozšířená podpora písmen
- Možnost dotykového ovládání aplikace
- Systém I18N/L10N pro lokalizaci aplikace

Rozhodl jsem se skončit u verze uvolněné 22.12.2008, která obsahuje již všechny potřebné funkce pro mou aplikaci a vykazuje lepší stabilitu než předchozí knihovny.

Pro přiložení dodatečných dat jako jsou obrázky nebo lokalizace, které knihovna potřebuje, slouží speciální nástroj pojmenovaný **Resource Editor**. Tento nástroj je dodávaný s distribucí knihovny a je napsán v klasické Java SE. Výstup tvoří jediný soubor, který už umí knihovna LWUIT dál zpracovat (viz obrázek 5.8).

5.5.1 Písmo a vzhled aplikace

Hlavní výhodou využití knihovny LWUIT je možnost využití písma systémového typu, nebo odvození vlastního bitmapového typu s vlastní znakovou sadou. Tato vlastnost zjednodušila řešení problému zobrazení znaků potřebných pro výslovnost nebo české diakritiky, které by klasické LCDUI nemuselo zvládnout. Pro aplikaci jsem proto zvolil 6 velikostí písma odvozené od rodiny Arial, které obsahuje všechny potřebné znaky. Výhodou bitmapových písmem je lepší možnost zobrazení na displeji díky podpoře vyhlazování, naopak jejich hlavní nevýhodou je daleko větší náročnost při vykreslení na displeji. Později se ukázala další nevýhoda, a to absolutní velikost písma. Písmo je tak závislé na rozměrech displeje a například písmo o velikosti 13, které zabere téměř celý displej o rozlišení 80x80, je stěží viditelné na rozlišení



Obrázek 5.8: Nástroj Resource Editor

Obrázek 5.9: Porovnání písem o velikosti 8 a 13

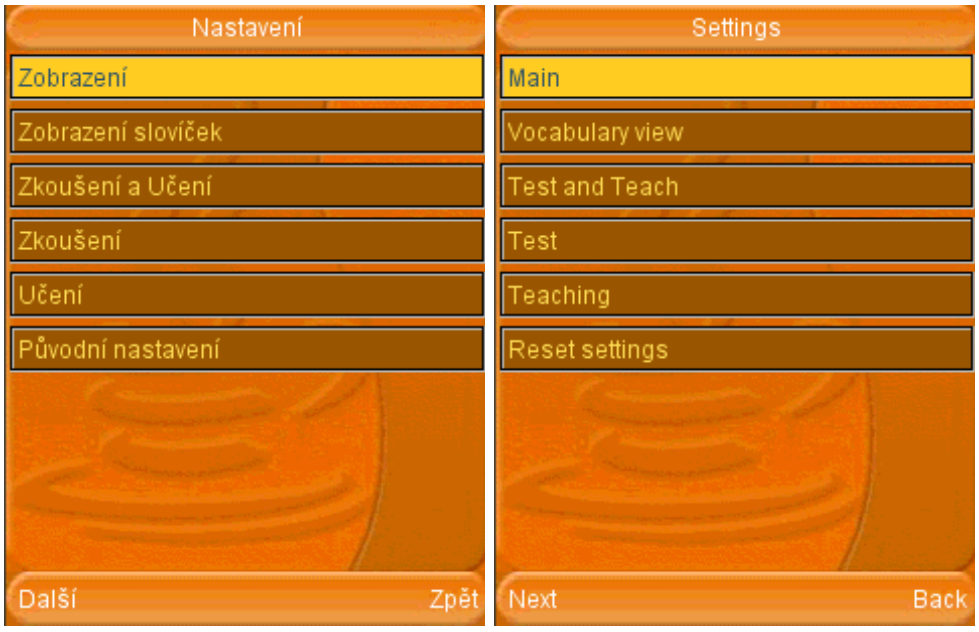
800x480. Na obrázku 5.9 je zachyceno porovnání písem velikosti 8 a 13, které by měli pokrýt potřeby nejrozšířenějších mobilních přístrojů.

S písmem souvisí vzhled aplikace, který každé komponentě přiřazuje její parametry pro zobrazení, jakými je například barva pozadí, popředí nebo také písmo textu. Definice vzhledu v knihovně LWUIT má syntaxi velice podobnou kaskádovým stylům css u webových stránek a proto není těžké na jednom místě radikálně změnit vzhled aplikace. Pro aplikaci jsem připravil dva typy vzhledů, jeden graficky bohatší pro rychlejší mobilní telefony a druhý chudší pro pomalejší přístroje. Jejich porovnání naleznete na obrázku 5.10. Ve výsledku je tak v souboru se zdroji pro knihovnu definováno 14 vzhledů, které jsou odstupňované podle velikosti písma.

5.5.2 Lokalizace

Další klíčovou vlastností LWUIT je podpora lokalizace, která je zařízená jednoduchou hashovací tabulkou. Do ní je možné ke každému klíči přiložit hodnotu, která reprezentuje hledaný text v předem nastaveném jazyce. Ve zdrojovém kódu aplikace tak používám například místo názvů tlačítek pouze jejich jedinečný identifikátor, který se použije pro vyhledání názvu v adekvátním jazyce. Pro mou aplikaci jsem se rozhodl vytvořit lokalizaci pro český a anglický jazyk. Další lokalizace by znamenala přidání další znakové sady do písem a tím k prohloubení náročnosti vykreslení uživatelského rozhraní, které dle výsledků profilace již teď zabírá mnoho času. Porovnání anglické a české lokalizace zachycuje obrázek 5.11.

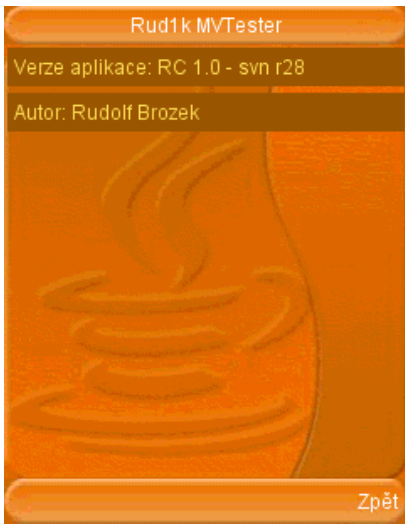
Obrázek 5.10: Porovnání vzhledu aplikace



Obrázek 5.11: Ukázka lokalizace do českého a anglického jazyka

Počet zdrojových souborů	50
Počet řádek	8057
Počet příkazů	5407
Procento skoků	20,3 %
Počet volání	3861
Procento komentářů	8.0 %
Počet tříd	91
Počet metod na třídu	4.95
Počet příkazů na metodu	8.64

Tabulka 5.2: Vlastnosti zdrojového kódu



Obrázek 5.12: Aktuální verze aplikace

5.6 Výsledný počet řádků

Pro změření množství kódu jsem použil aplikaci SourceMonitor, která umí ze zdrojových kódů zjistit kromě čistého počtu řádků i další zajímavé parametry. Shrnutí naleznete v tabulce 5.2, ze kterého můžeme zjistit, že původní odhad 5000 řádků byl o 3000 podhodnocen.

5.7 Verzování

Zpočátku byla aplikace verzována lokálně na pevném disku počítače, ale po zkušenostech se systémem subversion jsem se rozhodl založit projekt na stránkách Google Code, které poskytují základní nástroje pro podporu vývoje a distribuce aplikace. Tím jsem získal možnost snadné správy verzí na základě subversion a zálohování na serverech společnosti Google. Zobrazením aktuální verze v okně **O aplikaci** došlo k lepším výsledkům při hlášení chyb v procesu testování aplikace. (viz obrázek 5.12). Stránky projektu naleznete na adrese http://code.google.com/p/rvtester/.

Kapitola 6

Testování

Velice důležitou částí procesu vývoje aplikace je její testování. Vhodné je použít vhodné postupy i nástroje tak, aby došlo k nalezení co možná nejvíce chyb nebo špatně optimalizovaných míst, které během vývoje určitě musí vzniknout. Testování Java ME aplikací však bývá vzhledem k omezeným možnostem cíleného systému velice náročné. Teoreticky lze provést pouze v emulátorech, protože cílené mobilní zařízení nemusí mít prostředky na to, aby testování bez problému zvládlo. Tento přístup ovšem není ideální, jelikož se aplikace může chovat jinak v emulátoru, než v opravdovém zařízení. A opravdu jsem na tento jev také několikrát narazil. Během této práce jsem si vyzkoušel především funkcionální testování, které se zaměřuje na kontrolu správně implementovaných funkcí systému dle zadání.

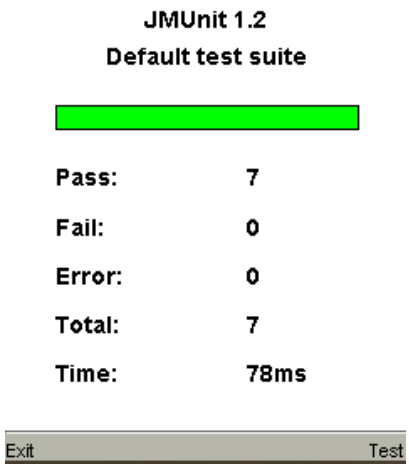
6.1 Unit testing

Částečně jsme zabrousil do oblasti jednotkových testů (unit testing), který je však v JavaME trochu problematický. Klasický JUnit framework totiž spoléhá na reflexi, která zde chybí a tak klasické nástroje nepomáhají. IDE Netbeans přídavkem mobility má od verze 6.5 podobný framework, který se JUnit snaží přiblížit, ale výsledek není dokonalý. Firma Sony Ericsson poskytuje svůj framework nazvaný **Sony Ericsson Mobile JUnit**, který se původnímu JUnit přibližuje asi nejvíce a také lze snadno zakomponovat do IDE Netbeans. Po zkušenostech jsem se ho ale rozhodl použít jen na kritické části aplikace a vyzkoušet třídu pro konverze hodnot z little-endian formy zápisu. Vygeneroval jsem pro ní proto speciální třídu přímo pro unit test, která obsahuje funkce potřebné k otestování všech jejích funkcí.

Zde uvádím příklad jedné z funkcí, kterou potřebujeme otestovat.

```
public static short readShortLittleEndian(InputStream is)
throws IOException {
    int low = is.read();
    int high = is.read();
    return(short)( high << 8 | low );
}
```

Pro ní se vygenerovala tato funkce, kterou jsem doplnil o potřebnou logiku.



Obrázek 6.1: Obrazovka úspěšného testu JMUnit

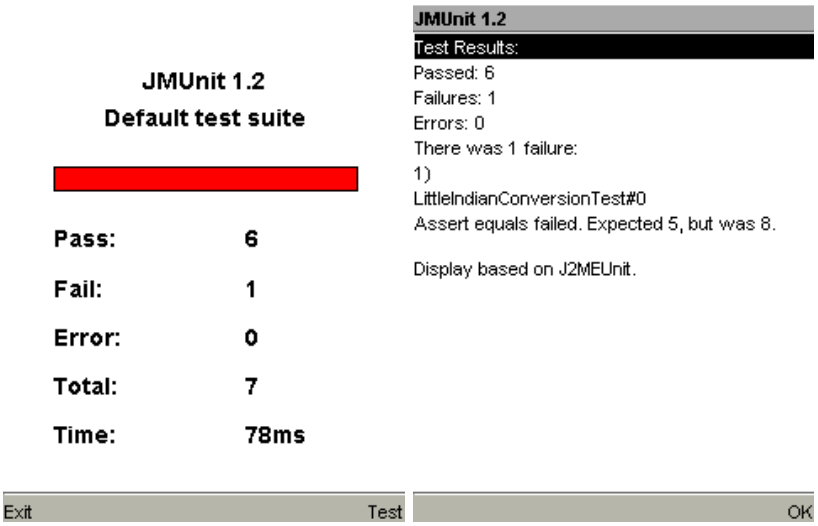
```
public void testReadShortLittleEndian()
throws AssertionFailedException, Exception {
    System.out.println("readShortLittleEndian");
    byte[] b = {5,0};
    InputStream is_1 = new ByteArrayInputStream(b);
    short expResult_1 = 5;
    short result_1 = LittleIndianConversion.readShortLittleEndian(is_1);
    assertEquals(expResult_1, result_1);
}
```

Výsledkem tohoto testu je zobrazení informace, zda se všechny funkce chovají tak, jak mají. V našem případě tomu tak opravdu je, viz. obrázek 6.1. Pokud by ale ovšem některá z funkcí byla špatně napsaná, zobrazilo by se nám okno na obrázku 6.2 vlevo, v případě potřeby zobrazení podrobností test ukáže okno na obrázku 6.2 vpravo.

6.2 Usability test

Šikovnou cestou jak otestovat aplikaci je provést test použitelnosti na vzorku předpokládaných uživatelů. Tento test má šanci odhalit chyby, které by se určitě objevily během používání aplikace, ale především slouží k otestování jak moc špatně je navržené uživatelské rozhraní. Pro test jsem vytvořil jednoduchý scénář, který jsem v několika bězích vyzkoušel na vybraných jedincích a zaměřil se na následující oblasti: (Scénář naleznete v příloze D).

- Efektivita - jak dlouho trvá, než uživatel načte slovník nebo spustí zkoušení.
- Přesnost - počet chyb, které uživatel provedl při provádění úkolů.
- Emotivnost - pocity z dokončení úloh.



Obrázek 6.2: Obrazovka neúspěšného testu JMUnit

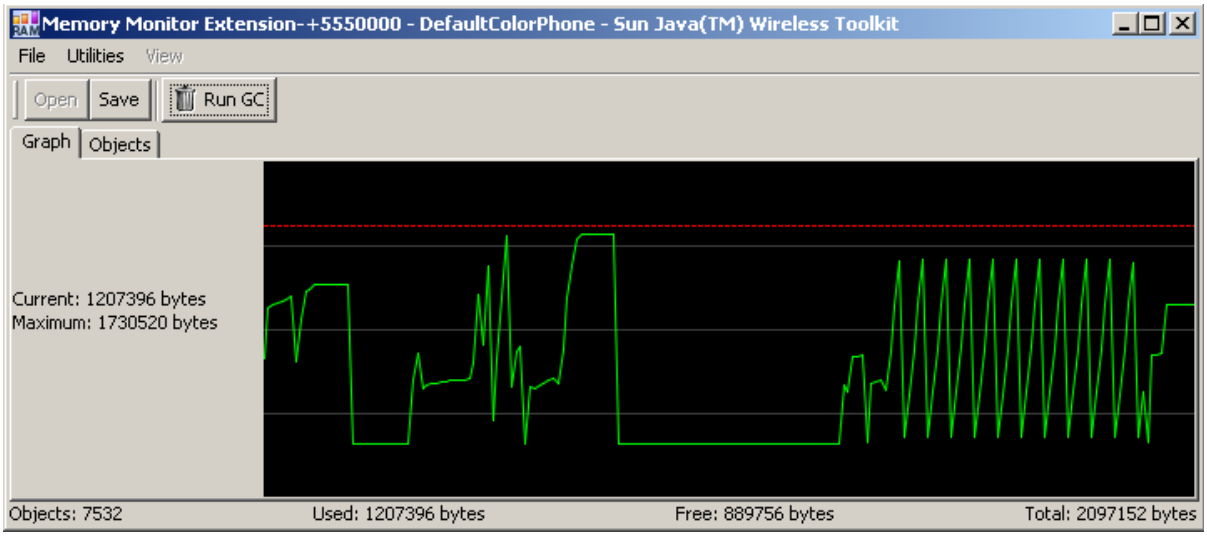
Výsledkem prvního běhu testu bylo zjištění neporozumění termínů a chaotické procházení aplikací u některých jedinců. Ve výsledku se všem lidem povedlo úkoly splnit, ale doba se velice lišila a dle mého názoru byla v průměru příliš dlouhá. Jako nejvíce problematický bod se ukázalo přidávání slovíček, které se v některé případech lišilo oproti ideálnímu splnění téměř o minutu. Proto jsem se rozhodl k napsání jednoduchého manuálu pro ovládání aplikace a upravení některých pojmů, které uživatele nejvíce mátly. Druhý běh, tentokrát i s manuálem, proběhl bez větších problémů a poskytl pouze pár postřehů pro upravení manuálu. Třetí běh testování už neproběhl.

Jako velice cenné se ukázali konzultace s vedoucím práce, panem Ing. Pavlem Kubalíkem, Ph.D, který odhalil řadu nedostatků v uživatelském rozhraní ještě před testováním. Díky těmto konzultacím došlo například k již zmiňovanému odstranění správce slovníků, který se ukázal jako velice problematický.

6.3 Podpůrné nástroje SDK

SDK Sun Java Wireless 2.5.2 poskytuje řadu šikovných nástrojů, které v mnoha případech pomohou s laděním nebo dokáží odhalit špatně optimalizovaná místa v programu. Jedním z takových nástrojů je Memory Monitor, který umí monitorovat alokaci paměti a v případě potřeby zde může programátor pustit dokonce garbage kolektor. Na obrázku 6.3 je zobrazen průběh načítání slovníku s 1000 slovíčky. První třetina odpovídá běžnému používání aplikace, poté jsem spustil GC, který uvolnil nepotřebnou paměť a spustil načítání. Všimněte si, že i před začátkem načítání má aplikace alokovanou určitou část paměti. Po spuštění můžeme sledovat prudký nárůst alokované paměti až po kritickou mez a následné uvolnění garbage kolektorem. Tento koloběh se několikrát opakuje až po kompletní načtení slovníků do vnitřního úložiště.

Dalším takovým nástrojem je Profiler, který pomocí dynamické analýzy kódu za běhu umí určit dobu provádění jednotlivých částí programu. To se hodí pro optimalizaci, jelikož



Obrázek 6.3: Nástroj Memory Monitor

přístroj	první spuštění	druhé spuštění
Emulátor	3s	-
Nokia 6085	5s	3s
HTC Diamond2	1s	1s

Tabulka 6.1: Doba spuštění aplikace

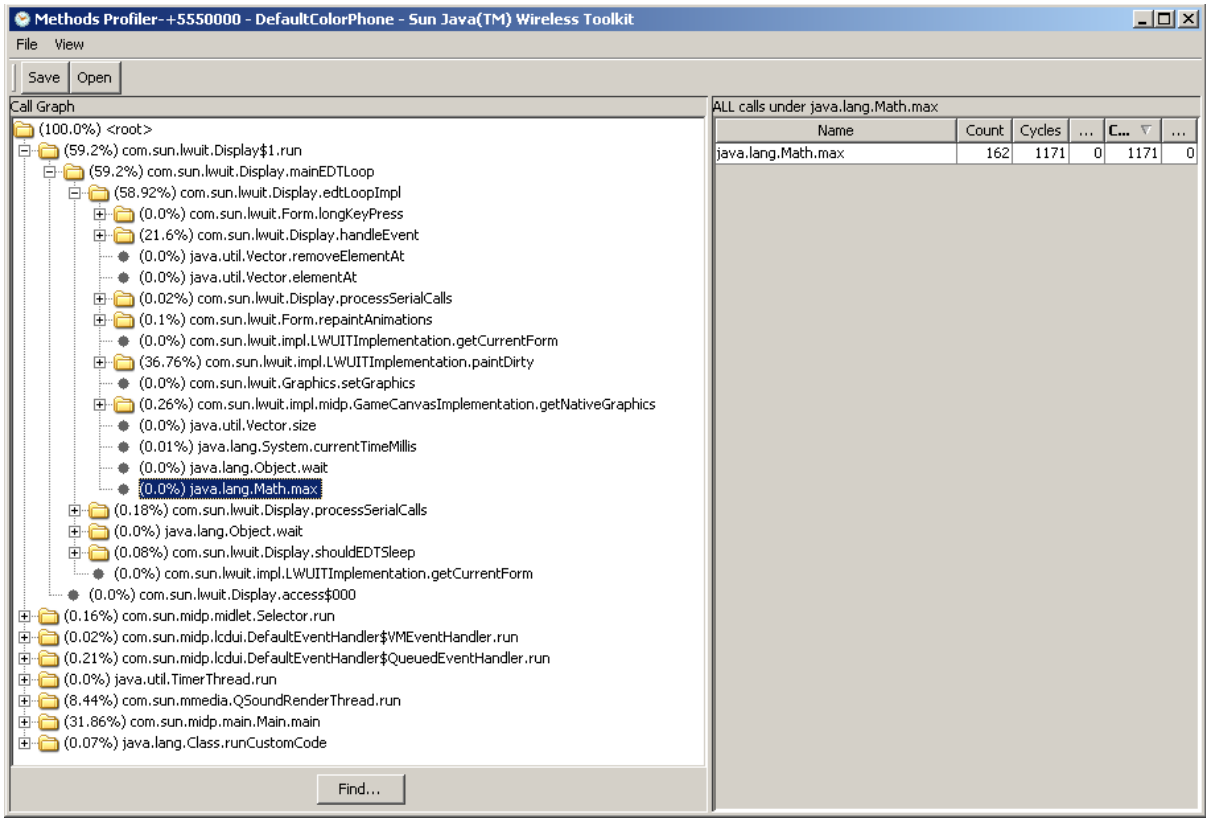
můžeme určit funkce, které se provádějí nejčastěji nebo nejdéle a které je tak vhodné optimalizovat. Příklad naleznete na obrázku 6.4, na kterém je profilován průběh zkoušení slovíček ze slovníku o kapacitě 500 slovíček. Je zde patrné, že knihovna LWUIT s téměř 60% podílem je velice náročná a z mé strany těžko optimalizovatelná.

6.4 Parametry běhu programu

Díky již zmíněným omezeným možnostem cíleného systému je zjištění přesných parametrů běhu aplikace téměř nemožné a díky veliké fragmentaci se zjištěné parametry velice liší. Zde zveřejněné údaje jsou proto získané částečně z emulátoru obsaženém v SDK Sun Java Wireless 2.5.2 a částečně z mobilních telefonů Nokia 6085 a HTC Diamond2. Emulátor byl provozován na počítači se systémem Windows XP SP3 a procesorem Intel Core2Duo E8300.

Čas potřebný pro první a druhé spuštění je uveden v tabulce 6.1. Čas druhého spuštění jsem uvedl kvůli výsledkům na přístroji Nokia 6085, který jako jediný vykazoval lepší výsledek než při prvním spuštění. Emulátor bohužel nedokázal aplikaci napodruhé spustit a musel být ukončen. Tato chyba se vyskytla až po převedení uživatelského prostředí aplikace pod knihovnu LWUIT a tak problém vidím někde na straně mezi emulátorem a knihovnou.

V tabulkách 6.2 a 6.3 je zobrazena závislost délky načtení slovníků na počtu obsažených slovíček. Všimněte si, že načtení textového slovníků (tabulka 6.2) probíhá téměř stejně



Obrázek 6.4: Nástroj Profiler

	Počet slovíček							
přístroj	**10**	**30**	**50**	**100**	**200**	**300**	**500**	**1000**
Emulátor	<1s	<1s	<1s	1s	1.5s	2.4s	3.2s	6s
Nokia 6085	<1s	<1s	<1s	1.4s	3s	4.7s	14.9s	16s
HTC Diamond2	<1s	<1s	<1s	<1s	1.4s	3s	4s	7.3s

Tabulka 6.2: Doba načtení txt slovníku

	Počet slovíček							
přístroj	**10**	**30**	**50**	**100**	**200**	**300**	**500**	**1000**
Emulátor	<1s	<1s	1s	1s	1.8s	2.5s	3.0s	7s
Nokia 6085	<1s	<1s	1.4s	3s	6.6s	10.3s	18s	37s
HTC Diamond2	<1s	<1s	<1s	<1s	1.6s	3s	8s	15s

Tabulka 6.3: Doba načtení Vocab Pro slovníku

rychle jako načtení slovníků Vocab Pro (tabulka 6.3). Tento jev mě překvapil, jelikož načtení textového slovníku je složitější o rekurzivní volání funkcí syntaktického analyzátoru, kdežto Vocab Pro slovník se čte přímo.

V tabulce 6.4 jsou zachyceny paměťové nároky slovníků na vnitřní úložiště v závislosti na počtu slovíček.

6.5 Porovnání s konkurencí

Vraťme se ke kapitole rešerše, kde jsem porovnával existující řešení na trhu. Při návrhu a vývoji aplikace jsem se snažil přiblížit k aplikaci Live Teacher, která se mi líbila nejvíce. Pokud se podíváme na všechny její plusy, myslím, že se mi až na komplikovaného správce slovníků povedly všechny zakomponovat a přidat některé funkce navíc. Aplikaci jsem vyčítal nemožnost zobrazit statistiku, která v mé verzi jde a absenci nápovědy. S odstupem času musím připustit, že nápověda na tak malém displeji nemá smysl a lze jí nahradit tištěným manuálem. Také jsem aplikaci kritizoval za poskytnutí pouze jedné metody zkoušení, která se ale ukázala dostatečná. Implementoval jsem navíc pouze možnost žádné správné odpovědi v nabízených možnostech.

Možnost importovat textové slovníky umožňovala pouze aplikace Vocabulary Trainer, která ale pravděpodobně funguje pouze na přístroji Siemens SE45. Navíc jsem přidal možnost importu a exportu slovníků do aplikace Vocab Pro.

Výhodu aplikace Verbs vidím v podpoře výuky anglické gramatiky, kterou jsem neimplementoval.

Počet slovíček	**10**	**30**	**50**	**100**	**200**	**300**	**500**	**1000**
Kapacita	3kB	5kB	7kB	12kB	23kB	34kB	55kB	109kB

Tabulka 6.4: Zabraná kapacita vnitřního úložiště v závislosti na počtu slovíček

Kapitola 7

Závěr

Tato kapitola zhodnocuje splnění cílů bakalářské práce, které byly položeny na začátků a shrnuty v podkapitole specifikace. Pokusím se zde vystihnout přínos, který má práce přináší mezi současná řešení shrnutých v rešerši a navrhnout možná rozšíření při pokračování ve vývoji aplikace.

7.1 Výsledek a přínos

Platforma JavaME se ukázala jako vhodná a i přes počáteční problémy s jejím osvojením a po delším seznamování s knihovnou LWUIT se podařilo vytvořit aplikaci splňující všechny body položené ve specifikaci i s některými rozšířeními. Ve výsledku je tak uživateli poskytnuta jednoduchá cesta, jak si narychlo zaznamenat a naučit neznámé slovíčko. Jako hlavní přínos v této oblasti vidím uvolnění mé práce, která je již v této fázi konkurenceschopná a vhodná k šíření mezi širší část uživatelů, jako open-source. Oproti hlavnímu konkurentovi, aplikaci Live Teacher, přináší uživateli možnost vytvoření vlastních textových slovníků například na stolním počítači a jako jediná umí spolupracovat s programem Vocab Pro, který běží pod systémem Windows. V této spolupráci vidím veliký potenciál a výhodu, jelikož uživatel může upravovat již zaběhlé slovníky mimo mobilní přístroj. Dalším příjemným vylepšením je podpora různých vzhledů díky knihovně LWUIT, která by šla ještě zdokonalit o podporu uživatelem vytvořených vzhledů. Knihovna v mnohém pomohla, ale k její dokonalosti stále ještě mnoho chybí. Navíc na platformě JavaME je vidět, že díky omezeným schopnostem telefonních přístrojů občas ztrácí dech a v případě složitějších aplikací je lepší se poohlédnout po něčem jiném.

7.2 Možné pokračování

Jako další možné rozšíření aplikace vidím například přidání podpory pro výuku anglické gramatiky, jako je tomu v aplikaci Verbs. Takové rozšíření by nebylo vůbec těžké, stačilo by přidat obrazovku pro uživatelské rozhraní, rozšířit metodu traverse a upravit třídu ApplicationModel o podporu dalšího datového typu v lokálním úložišti, kam by se gramatika uložila. Aplikace by se dala také o hodně rozšířit využitím podpory pro bluetooth nebo web. Zde vidím využití hlavně pro sdílení slovníků nebo stažení z internetu. JavaME také poskytuje

tzv. „Push registry“, které slouží k naplánování spuštění nějakého MIDletu na určitou dobu. Zde se přímo nabízí funkce pro zkoušení slovíček po různých intervalech, mezi kterými by byla aplikace neaktivní. Za zmínku stojí také zakomponování systému rodičovské kontroly pro dohled nad studijními výsledky uživatele. [**?**]

Literatura

[1] V. Š. Báňská. Stránky předmětu TAMZ na VšB-TU. `http://wiki.cs.vsb.cz/index.php/Edu:TAMZ/cs`.

[2] M. Cinibulk. *Aplikace pro zkoušení slovní zásoby*. 2007. Bakalářská práce.

[3] J. Knudsen. LWUIT Tutorial. `https://lwuit.dev.java.net/tutorial/index.html`.

[4] S. Microsystems. Effective JavaME Programming.

[5] S. Microsystems. LWUIT Developers Guide.

[6] S. Microsystems. Managing the midlet life-cycle with a finite state machine.

[7] S. Microsystems. Stránky platformy JavaME. `http://java.sun.com/javame/index.jsp`.

[8] B. Mobilmania. Odkud se vzal symbian? `http://unleaded.blog.mobilmania.cz/`.

[9] Nokia. Stránky platformy SymbianOS. `http://developer.symbian.com/main/index.jsp`.

[10] OHA. Stránky platformy Android. `http://www.android.com/`.

[11] OHA. What is android? `http://developer.android.com/guide/index.html`.

[12] J. Pagonis. Overview of symbian os hardware interrupt handling.

[13] S. Press. JavaME on SymbianOS.

[14] P. Suchomel. Java ME + Mobility Pack. `http://www.avc-cvut.cz/avc.php?id=7173`.

[15] K. Topley. *J2ME V KOSTCE - Pohotovostní referenční příručka*, volume 1. Grada Publishing, Grada Publishing a.s.; U Pruhonu 22, Praha 7, 1th edition, 2002. In Czech.

[16] Wikipedia. Exploit. `http://en.wikipedia.org/wiki/Exploit_(computer_security)`.

Příloha A

Seznam použitých zkratek

API Application Programming Interface

CD Compact Disc

CDC Connected Device Configuration

CLDC Connected Limited Device Configuration

ČVUT České vysoké učení technické

EDT Event Dispatch Thread

GC garbage collector

GPL General Public License

GPS Global Positioning System

GUI Graphical user interface

IDE Integrated Drive Electronics

IPA International Phonetic Alphabet

JAD Java Application Descriptor

JAR Java Archive

JDK Java Development Kit

JavaME Java Micro Edition

JSR Java Specification Requests

JavaSE Java Standard Edition

KVM K Virtual Machine

LWUIT Lightweight User Interface Toolkit

LCDUI Liquid Crystal Display User Interface

MFC Microsoft Foundation Class

MIDP Mobile Information Device Profile

MVC Model-view-controller

OHA Open Handset Alliance

OS Operating system

PBP Personal Basis Profile

PC Personal Computer

PDA Personal Digital Assistant

PIM Personal information management

QVGA Quarter Video Graphics Array

RAM random-access memory

ROM Read-Only Memory

RMS record management store

SD Secure Digital

SP3 Service Pack 3

SDK Software development kit

SVN Subversion

UTF UCS Transformation Format

UML Unified Modeling Language

UCS Universal Character Set

UI User Interface

VGA Video Graphics Array

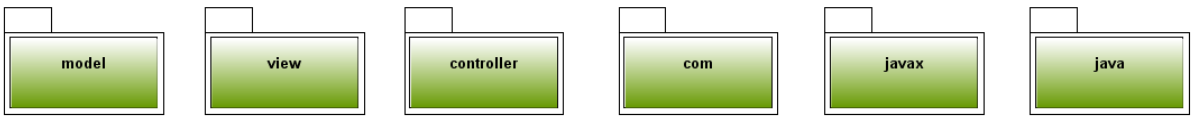
VM Virtual machine

2D Two-Dimensional

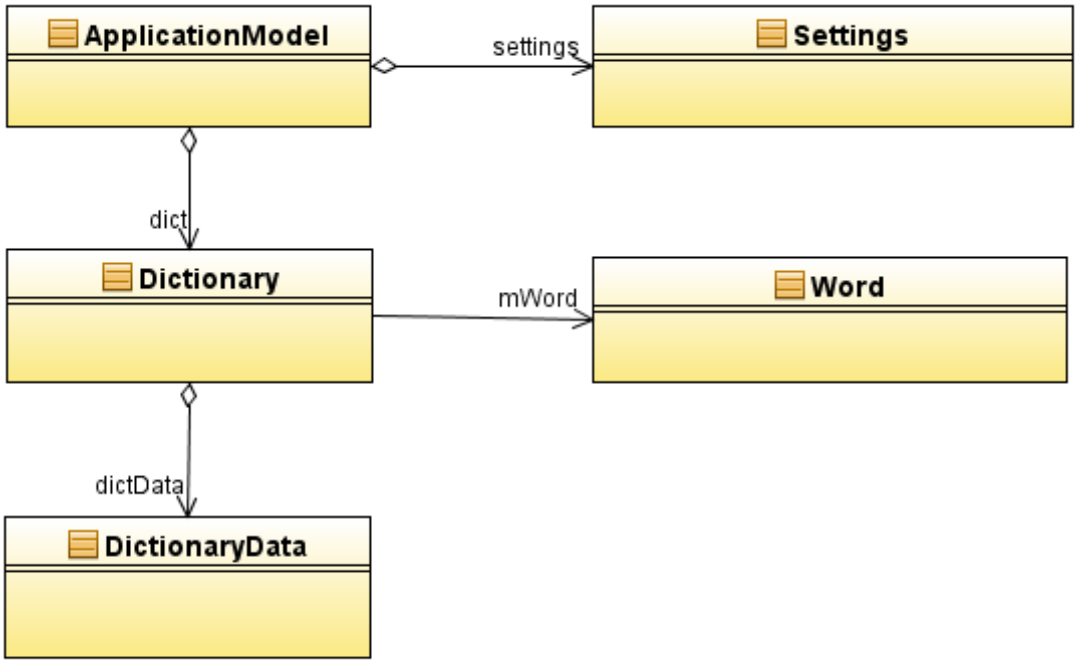
3D Three-Dimensional

Příloha B

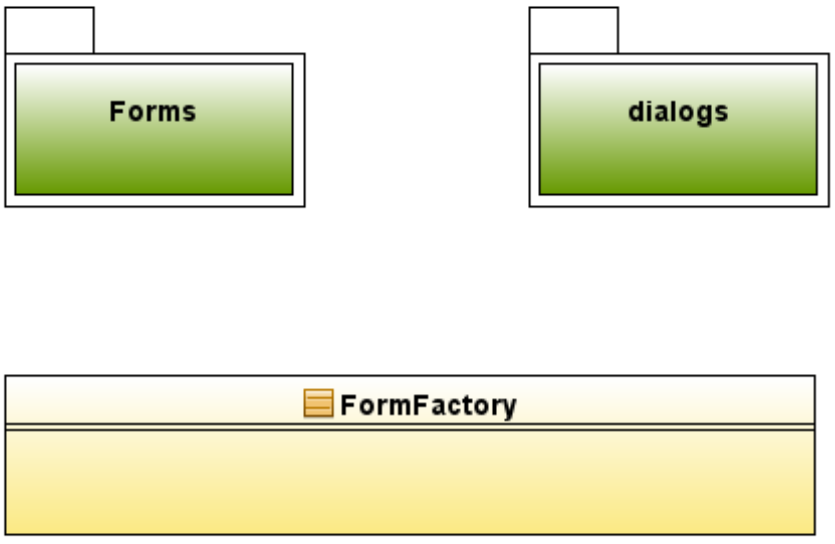
UML diagramy



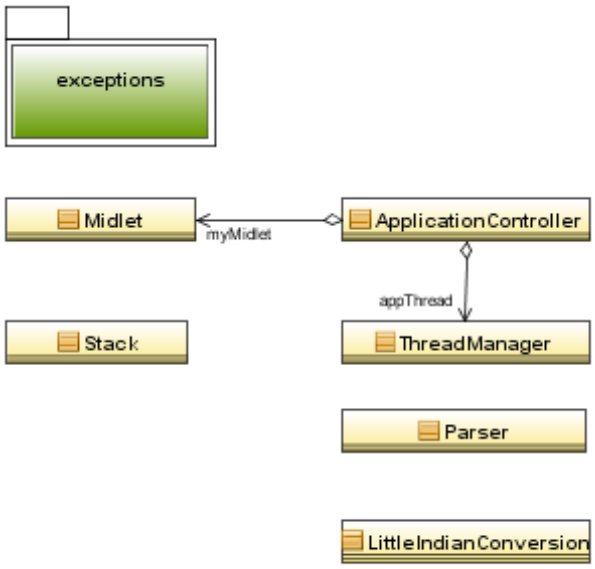
Obrázek B.1: Rozvržení balíků aplikace



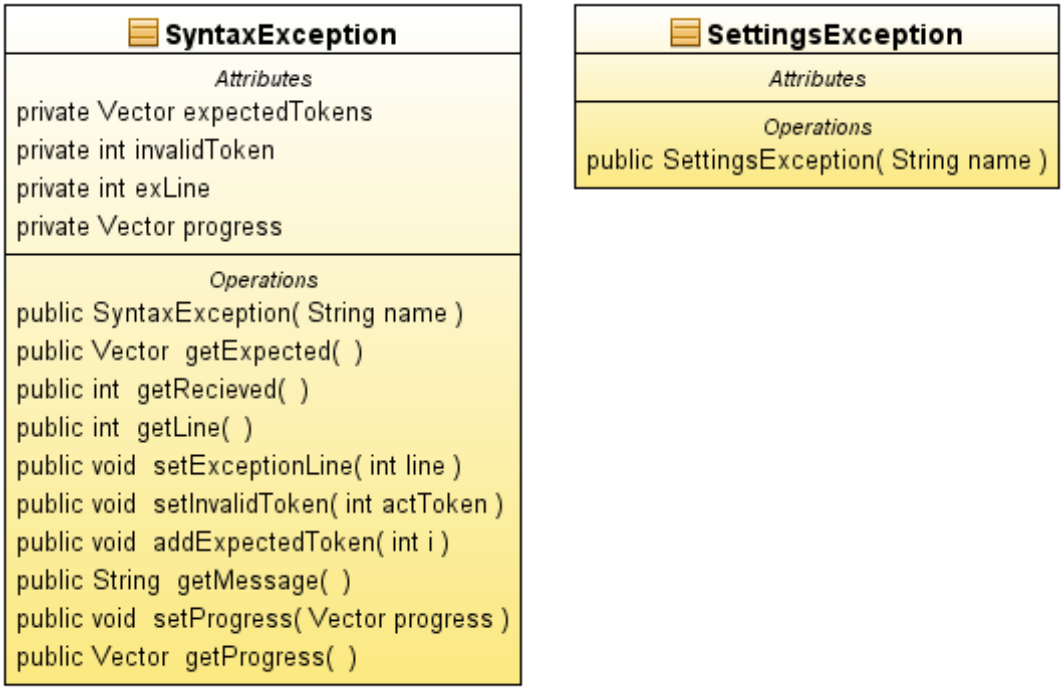
Obrázek B.2: Obsah balíku Model



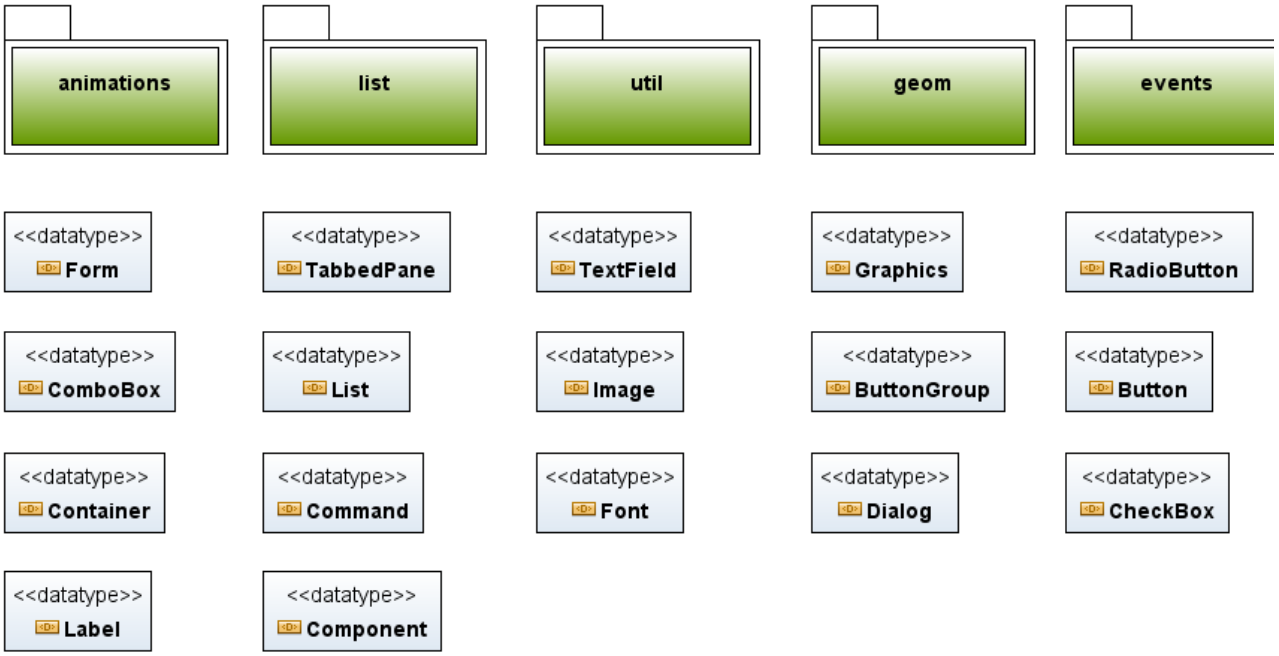
Obrázek B.3: Obsah balíku View



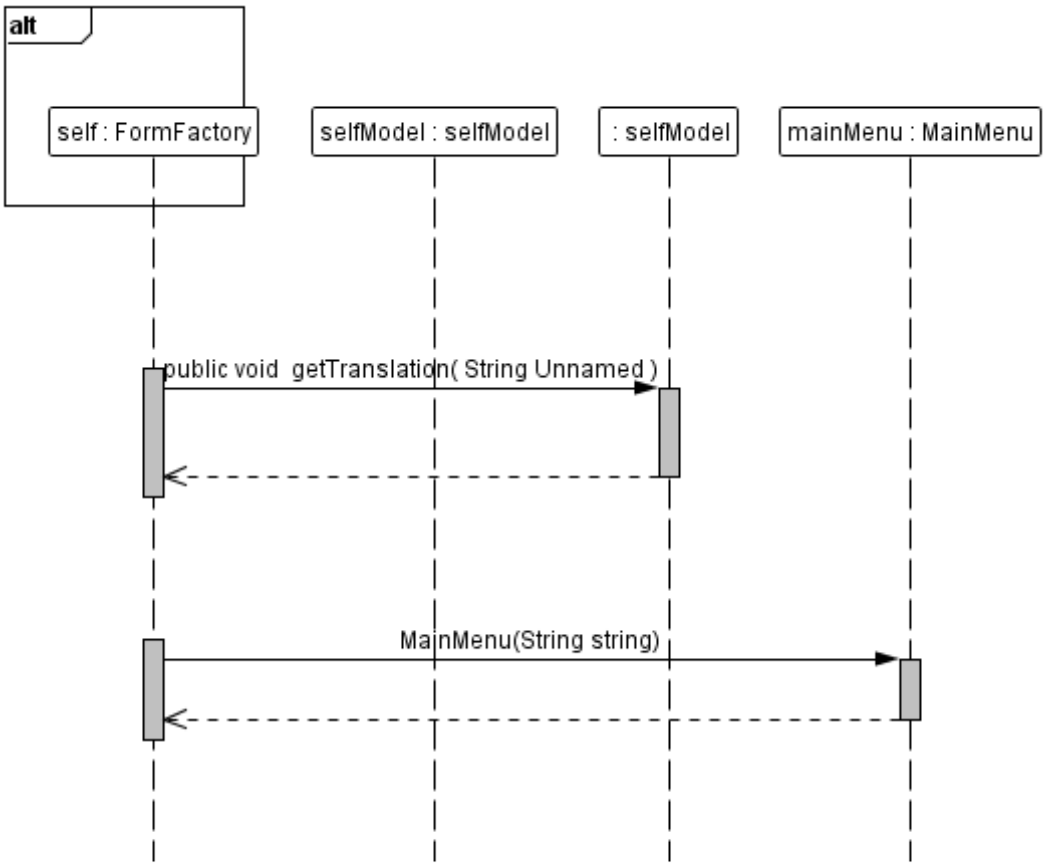
Obrázek B.4: Obsah balíku Controller



Obrázek B.5: Obsah balíku Exception



Obrázek B.6: Obsah balíku Com



Obrázek B.7: Sekvenční diagram FormFactory

Příloha C

Instalační a uživatelská příručka

C.1 Požadavky

- Mobilní telefon podporující JavaME v konfiguraci CLDC
- Profil MIDP a JSR75
- Hardwarová klávesnice

C.2 Instalace

Instalace aplikace se může lišit v závislosti na použitém telefonu, ale v zásadě stačí soubory aplikace do telefonu pouze nahrát. Telefon by měl sám zjistit, že se jedná o JavaME aplikaci a zobrazit jí v nabídce programy. U některých výrobců[1] je potřeba aplikaci nahrát do přímo určeného adresáře, který je předem připraven.

C.2.1 Windows Mobile

Aplikaci je možné provozovat i na platformě Windows Mobile, za předpokladu, že obsahuje dodatečně nainstalovaný emulátor jazyka JavaME. V současné době existují na trhu dvě rešení - **JBlend** od firmy Aplix a **Jbed** firmy Esmertec. Bohužel pouze JBed obsahuje všechny profily potřebné aplikací. Po nainstalování stačí přidat jar či jad soubor z paměťové karty a emulátor se už sám postará o spuštění aplikace.

C.2.2 Symbian

Platforma Symbian by měla aplikace JavaME podporovat od verze Symbian OS v9. Dle specifikace výrobce by měla být zaručena možnost spustit aplikace dle standartu CLDC 1.1 a MIDP 2.0 a slíbená je také podpora JSR 75. Bohužel funkčnost aplikace jsem nemohl ověřit a tak se zde musím spoléhat na tvrzení výrobce[13]. Zájemce odkazuji na volně dostupnou knihu JavaME on Symbian OS.

[1] Dle některých fór se jedná především o výrobce Siemens

C.3 Požadavky na formát slovníků

Jsou podporovány dva typy slovníků; slovníky aplikace Vocab Pro, které mají svůj formát daný a pak textové slovníky, které si uživatel může vytvořit a upravovat sám v libovolném textovém editoru. Takový slovník pak musí být uložen ve formátu UTF8 v textovém souboru s koncovkou .txt a slovíčka musí být zapsána v tomto tvaru:

```
anglický význam [výslovnost] = český význam
```

nebo

```
anglický význam = český význam
```

Pro ukončení slovíčka stačí stisknout enter a pokračovat na dalším řádku. Anglický význam může obsahovat různý počet slov a následující znaky: `?`, `!`, `-`, `.` a `'`. Výslovnost rozšiřuje anglický význam o některé znaky mezinárodní fonetické abecedy. Český význam může obsahovat libovolné znaky abecedy a znaky `(` a `)`.

Textové slovníky lze pouze načíst, uložení a export slovníku probíhá vždy do formátu aplikace Vocab Pro.

C.4 První spuštění

Při prvním spuštění aplikace dojde k načtení nastavení lokálního prostředí mobilního telefonu, podle kterého se pokračuje v češtině nebo angličtině. Dále budete informování o tom, že aplikace rozpoznala první spuštění s výzvou, zda chcete změnit nastavení aplikace.

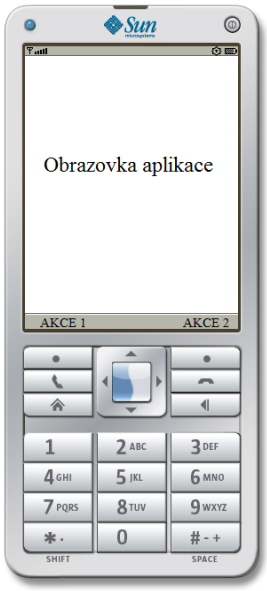
C.5 Ovládání aplikace

Uživatelské rozhraní je členěno do několika obrazovek, skrz které může uživatel procházet pomocí funkčních tlačítek. Na obrázku C.1 lze vidět typický mobilní telefon a rozvržení jeho tlačítek. Všimněte si, že ve spodní části displeje aplikace nabízí možné akce, které jsou přiřazené funkčním tlačítkům hned pod displejem. Dle zažité konvence mobilních aplikací může pravé tlačítko sloužit k následujícím akcím:

- **Návrat** na předchozí obrazovku
- **Zamítnutí** nabízené akce
- Zobrazení **podmenu** dalších akcí

Levé tlačítko naopak slouží především tyto akce:

- Přesun na **další obrazovku**
- **Potvrzení** nabízené akce



Obrázek C.1: Klasické rozvržení mobilního telefonu

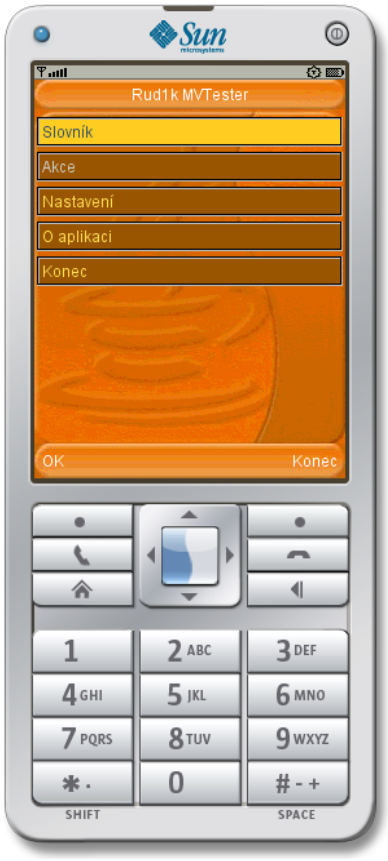
Pro ovládání aplikace lze použít i joystick, pokud je jím mobilní telefon vybaven. Stlačení joysticku směrem dolu vyvolá akci, kterou by jinak vyvolalo levé akční tlačítko. Pohybem nahoru či dolu lze listovat slovíčky slovníku a pohybem doprava či doleva lze procházet jednotlivými obrazovkami. Tlačítko zpět lze využít k návratu na předchozí obrazovku nebo rychlé zamítnutí akce. Zde se bohužel projevila již zmíněná fragmentace zařízení a tak knihovna LWUIT nepracuje správně s tlačítky některých přístrojů. Jedná se zejména o Windows Mobile zařízení a některé přístroje Siemens. Naopak s přístroji Sony Ericsson nebyl zaznamenán žádný problém.

Aplikace také podporuje ovládání pomocí dotykového displeje a tak můžete používat přímo zobrazené ovládací prvky na displeji přístroje. Zkušenosti jsou pouze s rezistentním typem, který rozpozná jedno místo dotyku. Tyto displeje používají především zařízení s Windows Mobile a Symbian.

C.6 Uživatelské rozhraní

Jak již bylo zmíněno, uživatelské rozhraní aplikace je tvořeno několika obrazovkami. Při spuštění se uživateli zobrazí hlavní obrazovka (viz obrázek C.2), přes kterou se celá aplikace ovládá. Z obrázku je patrné, že zobrazuje titulek popisující aktuální obrazovku a nabídku následujících akcí.

- **Slovník** pro práci se slovníky
- **Akce** pro práci s právě načteným slovníkem
- **Nastavení** pro nastavení parametrů aplikace
- **O aplikaci** pro zobrazení informací o aplikaci



Obrázek C.2: Spuštěná aplikace na mobilním telefonu

Obrázek C.3: Hlavní obrazovka aplikace

- **Konec** pro ukončení práce se slovníkem

V případě již otevřeného slovníku se zobrazí jeho jméno a počet slovíček. Pokud je slovník pozměněn (proběhlo například zkoušení, nebo došlo k editaci slovíček), je za jeho jménem hvězdička, upozorňující na potřebu slovník uložit (viz obrázek C.3).

C.7 Slovník - Základní operace se slovníky

Tato obrazovka slouží pro všechny potřebné úkony při práci se slovníky, jako je například otevření nebo uložení slovníku. Více obrázek C.4. Všimněte si, že aplikace povolí podle stavu aktuálního slovníku jen některé akce. Například pokud zatím nemáte žádný slovník otevřený, můžete pouze vytvořit **nový** slovník, nebo **otevřít** slovník uložený na paměťové kartě. V případě již otevřeného slovníku dojde ke zpřístupnění možnosti **ulož jako** a **export** slovníku. Položka **ulož** se stane aktivní až při neuložené změně ve slovníku. Vhodné je vědět, že aplikace kontroluje neuložený stav slovníku a v případě rizika ztráty dat budete vyzváni, zda chcete stávající slovník napřed uložit (viz obrázek C.4 vpravo).

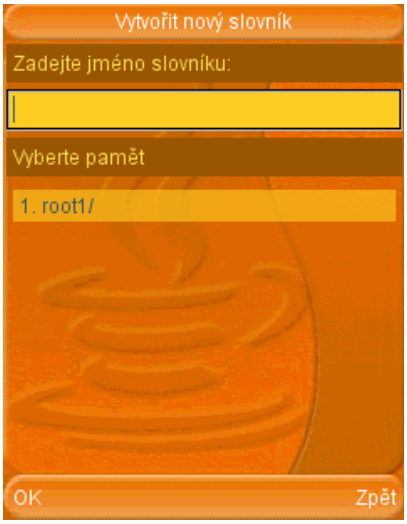
C.7.1 Vytvoření slovníku

V případě že chcete vytvořit nový slovník, zvolte první možnost v obrazovce Slovník - **Nový** (viz obrázek C.4 vlevo). Následně budete vyzváni k zadání jména nového slovníku a k výběru paměťové karty, kam se slovník uloží (viz obrázek C.5). Jméno slovníku je vyžadovaná, délkou neomezená položka a bez jeho zadání Vám aplikace nedovolí nový slovník vytvořit.

C.7.2 Otevření slovníku

Pokud chcete otevřít již existující slovník, zvolte druhou možnost v obrazovce Slovník - **Otevřít** (viz obrázek C.4 uprostřed). Aplikace bohužel není certifikovaná a tak je velká

Obrázek C.4: Akce se slovníky



Obrázek C.5: Vytvoření nového slovníku

Obrázek C.6: Průběh načítání slovníku

pravděpodobnost, že se Vás mobilní přístroj zeptá, zda chcete povolit přístup k souborům na paměťové kartě. Některé firmy[2] tento přístup dokonce v základním nastavení telefonu zablokují a tak je potřeba pro každou aplikaci zvlášť povolit zobrazení dotazu ohledně přístupu k souborům. Sony Ericsson dovoluje povolit aplikaci přístup napořád. V případě úspěchu se zobrazí obrazovka s jednoduchým průzkumníkem, se kterým se dostanete až k místu, kde máte slovníky uložené. V průběhu načítání slovníku se zobrazí jednoduchá obrazovka, která informuje o počtu již načtených slovíček a poté obrazovka upozorňující na dokončení načítání (viz. obrázek C.6 vlevo a uprostřed).

Vhodné je také podotknout, že se aplikace sama přizpůsobí formátu načítaného slovníku a je jedno, zda jste zvolili slovník ve formátu .dic nebo .txt. V případě chyby v syntaxi textového slovníku syntaktický analyzátor načítání přeruší a budete upozorněn na řádek a přesný důvod chyby. Příklad naleznete na obrázku C.6 vpravo - na kterém vidíme, že je problém s řádkem č. 5.

```
1. abuse           = urážen, zneužívat
2. accident [acident] = nehoda
3. act     [akt]     = jednání, hra
4. acting = hraní
5. actress = herecka
6. at last = konečně []
7. at least = alespon, nejméne
8. award = ocenení
9. brake = brzda
```

Pokud se podíváme pozorně, syntaktický analyzátor aplikace nás upozorní i na to, co je na řádku špatně. Vidíme, že slova at i last zastupují VYZN1, poté bylo detekováno znaménko = a slovo konečně zde stojí jako VYZN2. Poté se ale načetl symbol [který se vyhodnotil

[2]Na tento přístup jsem narazil především u firmy Nokia

Obrázek C.7: Průběh uložení a exportu slovníku

jako CHYBA. V políčku očekáváno je zobrazeno co přesně mělo následovat - v tomto případě pokračování na další řádek nebo konec souboru.

C.7.3 Uložení a export slovníku

Pokud je potřeba uložit změny ve slovníku, můžete slovník uložit pod starým jménem na místo, ze kterého jste slovník načetli a nebo pod jiným jménem tlačítkem **Uložit jako** (viz obr. C.4). V případě, že chcete slovník uložit pod jiným jménem, budete vyzváni k zadání nového jména a k výběru paměťové karty (viz obr. C.7 vlevo). Tlačítko **Export** slouží k vytvoření nového slovníku, do kterého se uloží jen slovíčka splňující zvolenou úspěšnost. (viz obr. C.7 uprostřed). V průběhu ukládání je zobrazena jednoduchá obrazovka, která informuje o počtu již uložených slovíček a poté obrazovka upozorňující na dokončení načítání (viz. obrázek C.7 vpravo).

C.8 Akce - Základní operace se slovníkem

Tato obrazovka je přístupná až po načtení některého slovníku a obsahuje veškeré operace, které lze s načteným slovníkem provádět. Například zde můžete spustit učení, nebo se nechat vyzkoušet ze slovíček. V případě potřeby si můžete také nechat zobrazit průběžné výsledky učení a zkoušení. Tlačítko restartovat slouží k vymazání statistiky všech slovíček slovníku. Restartování se provádí také v případě, že jsou všechna slovíčka naučená nebo mají určitý počet správných odpovědí při zkoušení. Tlačítkem Slovíčka se zobrazí obrazovka se seznamem slovíček, které jsou v daném slovníku obsaženy (viz. obrázek C.8).

C.8.1 Učení slovíček

Spuštění učení se provede tlačítkem učení na obrazovce Akce. Po spuštění se Vám budou postupně zobrazovat slovíčka ze slovníku a s možnostmi 1. Znám, 2. Neznám a 3. Přeskočit

Obrázek C.8: Akce se slovníkem

(viz obrázek C.9 vlevo). V případě, že si budete přát učení přerušit, použijte funkční klávesu přerušit. Statistika se zaznamenává průběžně a na konci učení se zobrazí. Důležité je poznamenat, že se automaticky neukládá do souboru se slovníkem, ale pouze do vnitřní paměti aplikace. V ní zůstane do té doby, než otevřete nebo vytvoříte jiný slovník.

C.8.2 Přezkušování slovíček

Přezkoušení slovíček naleznete pod tlačítkem Vyzkoušet na obrazovce Akce. Průběh zkoušení je velice podobný jako učení, místo předchozích možností jsou ale nabízena náhodná slovíčka ze slovníku. Pokud odpovíte správně na slovíčko, aplikace vám nabídne další do té doby než zkoušení přerušíte nebo dojde k vyčerpání slovíček. Při špatné odpovědi budete buď vyzván k opravě, nebo se přejde na další slovíčko se zobrazením správné odpovědi. To záleží na nastavení zkoušení. Statistika se opět zaznamenává průběžně a ke konci zkoušení se zobrazí. Opět zde platí, že je potřeba statistiku do souboru uložit ručně (viz obrázek C.9 uprostřed a vpravo).

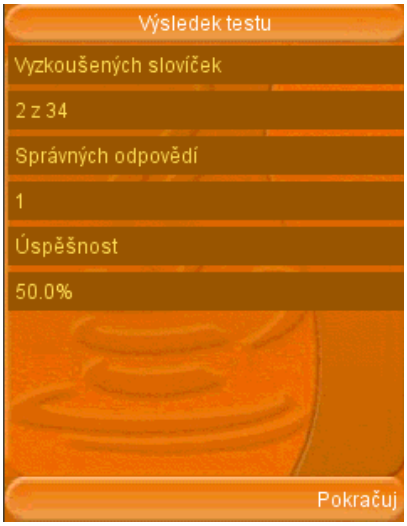
C.8.3 Výsledky učení a zkoušení

Statistika pro učení i zkoušení se nachází pod tlačítkem Výsledky. Obrazovka výsledků je rozdělena na tři záložky - Obecné, Učení a Zkoušení. V první záložce naleznete základní údaje o slovníku i počet naučených, či nenaučených slovíček. Záložka Učení zobrazuje počet kladných odpovědí vím a počet záporných odpovědí nevím v průběhu učení slovníku. Záložka Zkoušení obsahuje obdobné informace pro zkoušená slovíčka (viz obrázek C.10).

C.8.4 Zobrazení slovíček slovníku

Slovíčka, obsažená v otevřeném slovníku, jsou dostupná přes tlačítko Slovíčka na obrazovce Akce, které Vás přesune na Obrazovku Slovník. Zde jsou zobrazena slovíčka ze slovníku, které

Obrázek C.9: Učení a zkoušení slovíček



Obrázek C.10: Výsledná statistika

Obrázek C.11: Zobrazení slovíček slovníku

můžete editovat nebo mazat. Zde také můžete přidat nové slovíčko. Pokud se nezobrazila všechna slovíčka a chcete přejít na další stránku, v menu akcí pravého funkčního tlačítka vyberte akci Další nebo Předchozí (viz obrázek C.11).

C.8.4.1 Přidání a editace slovíček

V případě potřeby přidání nového slovíčka, zvolte v menu akcí pravého funkčního tlačítka akci Přidej (viz obrázek C.12 vlevo). Zobrazí se obrazovka, ve které můžete vyplnit cizí význam slovíčka, přepis výslovnosti a jeho český význam (viz obrázek C.12 uprostřed). V případě potřeby vložení fonetického znaku do políčka výslovnost, zmáčkněte hardwarové tlačítko #. Po jeho stisknutí se zobrazí seznam symbolů, které mohou být pro výslovnost použity (viz obrázek C.12 vpravo). Operaci dokončíte levým funkčním tlačítkem přidej. Tlačítko zruš Vás vrátí na předchozí obrazovku bez přidání slovíčka.

Editace slovíček probíhá podobně jako přidání nového slovíčka. V obrazovce Slovíčka přejeďte na slovíčko, které chcete editovat a v menu akcí pravého funkčního tlačítka vyberte akci Uprav. Zobrazí se stejná obrazovka jako v případě přidání nového slovíčka, tentokrát s předvyplněnými hodnotami. Operaci dokončíte levým funkčním tlačítkem ulož. Tlačítko zruš Vás vrátí na předchozí obrazovku bez změny slovíčka.

C.9 Nastavení - přizpůsobení aplikace potřebám uživatele

Při prvním spuštění dojde k načtení základního nastavení, které vyhovuje nejvíce uživatelským potřebám. Pokud ale potřebujete některé parametry změnit, můžete tak učinit pomocí obrazovky Nastavení. Na této obrazovce jsou následující možnosti:

- **Zobrazení** pro nastavení jazyka, vzhledu, velikost písma a řazení slovíček.
- **Zobrazení slovíček** pro přizpůsobení zobrazení slovíček slovníku.

Obrázek C.12: Přidání a editace slovíčka

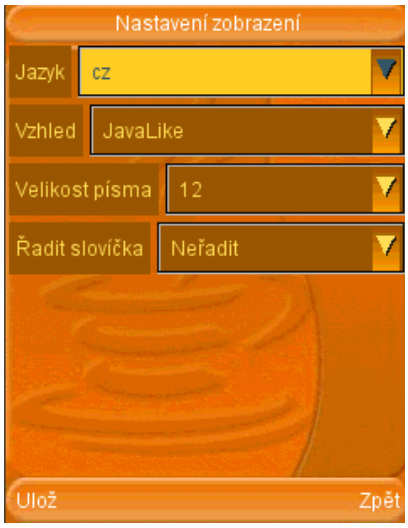
- **Zkoušení a učení** pro nastavení společných parametrů učení i zkoušení, mezi které patří pořadí slovíček a pořadí významů slovíčka.
- **Zkoušení** pro nastavení parametrů zkoušení.
- **Učení** pro nastavení parametrů učení.
- **Původní nastavení** pro obnovení základního nastavení.

C.9.1 Změna vzhledu, velikosti písma či jazyka

Aplikace nabízí dva možné vzhledy uživatelského rozhraní, jeden graficky bohatší a druhý prostší (viz obrázek C.14). V základním nastavení je použit první vzhled, uživatelům pomalejších přístrojů s menšími displeji je doporučeno nastavit druhý vzhled. Nastavení vzhledu naleznete na obrazovce Nastavení pod tlačítkem Zobrazení. Zde stačí rozkliknout nabídku u položky vzhled a vybrat jiný(viz obrázek C.13). Pro uložení změn je potřeba zmáčknout pravé funkční tlačítko Ulož. Na této obrazovce je také možnost přizpůsobit velikost písma rozměrům vašeho displeje, přepínat jazyk uživatelského rozhraní z češtiny do angličtiny a změnit způsob řazení slovíček - více v sekci Řazení slovíček.

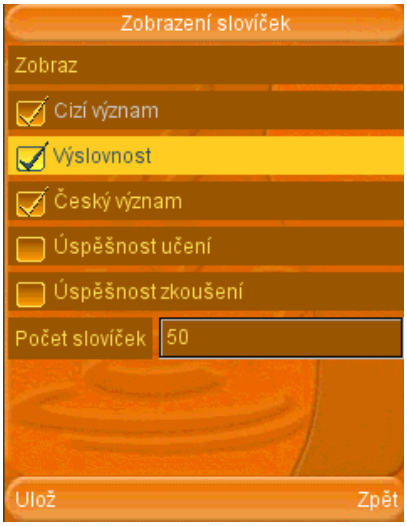
C.9.2 Řazení slovíček

Aplikace umožňuje řazení slovíček slovníku, bohužel nastavení jiného způsobu než neřadit nebo náhodně není doporučené pro větší slovníky nebo pro pomalejší mobilní přístroje. Dochází totiž k nepříjemným prodlevám při kterých aplikace slovíčka třídí dle nastaveného parametru.



Obrázek C.13: Změna vzhledu, velikosti písma či jazyka

Obrázek C.14: Porovnání vzhledu aplikace



Obrázek C.15: Parametry pro zobrazení slovíček slovníku

C.9.3 Upravení zobrazení slovíček slovníku

Pokud potřebujete změnit počet zobrazených slovíček slovníku na obrazovku, klikněte na tlačítko Zobrazení na obrazovce Nastavení a na následující obrazovce změňte položku Počet slovíček podle vašich potřeb. Na této obrazovce lze také přidat nebo odebrat prvky, které se pro každé slovíčko zobrazí (viz obrázek C.15).

C.9.4 Změna způsobu učení a zkoušení

Je podporováno několik možností učení a zkoušení. Například můžete obrátit význam slovíček a nechat aplikaci nabízet nejprve český význam s anglickými výrazy jako odpovědi. Tuto možnost povolíte zaškrtnutím políčka Prohodit významy na obrazovce Učení a Zkoušení (viz obrázek C.16 vlevo). Na této obrazovce také můžete změnit pořadí slovíček, ve kterém Vám budou nabízena. Zde ale platí stejné doporučení jako při řazení slovíček u zobrazení slovníku. Změny na této obrazovce se projeví jak v režimu učení, tak i v režimu zkoušení.

Učení lze ještě upravit tak, aby se automaticky ukázal druhý význam slovíčka automaticky po určitém časovém intervalu. Tuto možnost povolíte na obrazovce Učení pod tlačítkem Učení (viz obrázek C.16 vpravo).

U režimu zkoušení lze také nastavit, zda se má v nabízených možnostech odpovědi nacházet správná varianta, počet nabídnutých variant a zda má aplikace trvat na správné odpovědi. Tyto možnosti se nacházejí na obrazovce Nastavení Zkoušení pod tlačítkem Zkoušení (viz obrázek C.16 uprostřed).

Obrázek C.16: Parametry učení a zkoušení

Příloha D

Zadání pro test použitelnosti

Tato příloha zahrnuje přesné zadání pro test použitelnosti, který proběhl během testování aplikace.

D.1 Odborný článek

Classification[1]

There are several methods of classifying exploits. The most common is by how the exploit contacts the vulnerable software. A 'remote exploit' works over a network and exploits the security vulnerability without any prior access to the vulnerable system. A 'local exploit' requires prior access to the vulnerable system and usually increases the privileges of the person running the exploit past those granted by the system administrator. Exploits against client applications also exist, usually consisting of modified servers that send an exploit if accessed with client application. Exploits against client applications may also require some interaction with the user and thus may be used in combination with social engineering method.

Another classification is by the action against vulnerable system: unauthorized data access, arbitrary code execution, denial of service.

Many exploits are designed to provide superuser-level access to a computer system. However, it is also possible to use several exploits, first to gain low-level access, then to escalate privileges repeatedly until one reaches root.

Normally a single exploit can only take advantage of a specific software vulnerability. Often, when an exploit is published, the vulnerability is fixed through a patch and the exploit becomes obsolete for newer versions of the software. This is the reason why some blackhat hackers do not publish their exploits but keep them private to themselves or other crackers. Such exploits are referred to as 'zero day exploits' and to obtain access to such exploits is the primary desire of unskilled attackers, often nicknamed script kiddies

[1] Definice pojmu exploit tak, jak je popsaná zde [16]

D.2 Zadání úkolů

- Projděte si anglický text a poznamenejte neznámá slovíčka na papír
- Spusťte aplikaci
- Upravte nastavení zobrazení tak, aby odpovídal vašemu displeji a preferencím
- Vytvořte nový slovník pod názvem „Exploit“
- Přidejte do slovníku poznamenaná slovíčka

  Přidejte alespoň slovíčko `exploit  [ik'sploit] = využívat`
- Začněte s učením slovníku
- Po ukončení učení začněte zkoušení slovníku
- Změňte nastavení zkoušení - uberte počet nabízených významů na 3
- Restartujte slovník
- Znovu začněte se zkoušením
- Uložte slovník pod jménem exploit2
- Ukončete aplikaci

Příloha E

Obsah přiloženého CD

<SLOŽKA>	Bin	- zkompilovaný byte-code aplikace
<SLOŽKA>	Dictionary	- příklady slovníků pro aplikaci
<SLOŽKA>	Javadoc	- dokumentace javadoc
<SLOŽKA>	Source	- zdrojové kódy aplikace - projekt v Netbeans
<SLOŽKA>	text	- text bakalářské práce - formát pdf
<SLOŽKA>	textSource	- text bakalářské práce - formát LaTeX
<SLOŽKA>	Utils	- knihovna LWUIT a SDK
	readme.txt	- soubor s nápovědou