Robot Karel pro výuku jazyka Java

Roman Miko

Bakalářská práce
2018

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2017/2018

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Roman Miko**
Osobní číslo:	**A17602**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Robot Karel pro výuku jazyka Java**
Téma anglicky:	**"Karel the Robot" for the Tuition of the Java Language**

Zásady pro vypracování:

1. **Prostudujte stávající implementace Robota Karla, především ty s otevřenou licencí ke zdrojovým kódům.**
2. **Navrhněte strukturu tříd, reprezentující Karla a jeho město tak, aby výsledná knihovna byla co nejjednodušší a tedy vhodná pro úvodní lekce programování v jazyce Java.**
3. **Navrhněte atraktivní grafické uživatelské rozhraní pro nastavení města a provádění Karlových programů, včetně ladění.**
4. **Implementujte knihovnu a sadu příkladů, použitelných ve výuce.**
5. **Zpracujte dokumentaci uživatelskou i implementační.**

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **PATTIS, Richard E, Jim ROBERTS a Mark STEHLIK. Karel the robot: a gentle introduction to the art of programming. 2nd ed. /. New York: Wiley, c1995, xiii, 160 p. ISBN 04-715-9725-2.**
2. **BERGIN, Joseph, Jim ROBERTS a Mark STEHLIK. Karel: a gentle introduction to the art of object-oriented programming. 2nd ed. /. New York: Wiley, c1997, xii, 187 p. ISBN 04-711-3809-6.**
3. **DAVISON, Andrew. Programování dokonalých her v Javě: [programování her a grafiky v Javě]. Vyd. 1. Brno: Computer Press, 2006, 902 s. ISBN 80-7226-944-5.**
4. **DARWIN, Ian F. Java: kuchařka programátora : [vzory a řešení pro vaše aplikace]. Vyd. 1. Brno: Computer Press, 2006, 798 s. ISBN 80-251-0944-5.**
5. **HEROUT, Pavel. Java - grafické uživatelské prostředí a čeština. 1. vyd. České Budějovice: KOPP, 2006, 316 s. ISBN 80-7232-237-0.**

Vedoucí bakalářské práce: **Ing. Tomáš Dulík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **15. prosince 2017**

Termín odevzdání bakalářské práce: **25. května 2018**

Ve Zlíně dne 15. prosince 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

L.S.

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Jméno, příjmení: **Roman Miko**

Název bakalářské práce: **Robot Karel pro výuku jazyka Java**

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 24. 5. 2018

R. Miko
.....
podpis diplomanta

ABSTRAKT

Cílem této bakalářské práce je navrhnout a vytvořit programovací jazyk Robota Karel v jazyku Java, který bude využíván v úvodních studijních kurzech programování na FAI UTB ve Zlíně. V teoretické části bakalářské práce je programovací jazyk Robot Karel představen a charakterizován. Prezentuje se zde souhrn významných nebo zajímavých celosvětových a místních řešení jazyka Robot Karel. Dále praktická část bakalářské práce obsahuje navrhnutou strukturu tříd reprezentující Robota Karla a jeho město. V této části je také popsána vytvořená struktura Robota Karla a její programové rozhraní. Na závěr jsou představeny příklady pro výuku Javy s využitím jazyka Robot Karel.

Klíčová slova: Robot Karel, výuka programování, jazyk Java, rekurze, logické úlohy

ABSTRACT

The aim of this bachelor thesis is to design and create the programming language of Robot Karel in Java, which will be used in the initial programming courses at FAI UTB in Zlín. In the theoretical part of the bachelor thesis, the programming language of the Robot Karel is introduced and characterized. Here is a summary of significant or interesting global and local solutions for the Robot Karel. Further, the practical part of the bachelor thesis contains the designed structure of the classes representing the Robot Karel and his city. This section also describes the structure of the Robot Karel and its programming interface. Finally, examples for teaching Java language using the language of Robot Karel are presented.

Keywords: Robot Karel, Teaching of Programming, Java language

Tímto bych rád poděkoval vedoucímu bakalářské práce Ing. Tomášovi Dulíkovi, Ph.D. za věnovaný čas a věcné připomínky. Děkuji své nejbližší rodině za čas, který mi pro psaní této práce věnovali. Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Rozvoj a četnost technologií spojených s výpočetní technikou, které jsou akcelerovány zvyšující se efektivitou mikroelektroniky, vytváří a vyžaduje potřebu znalostí informatiky a software všeobecně od každého z nás. Ať jsme dítětem, dospělým, nebo důchodcem, ať chceme nebo ne, jsme obklopeni a obklopováni stále chytřejšími věcmi, které chceme nebo musíme ovládat. Tato potřeba jejich ovládání, a nejen počítače, ať už z pohledu uživatele, nebo vývojáře, je neodmyslitelnou běžnou dovedností dnes v mnoha pracovních oborech, od zemědělce, seřizovače v továrně až po lékaře nebo vědeckého pracovníka. Profese, kde se nevyužívají počítače, či řídící automaty minimálně jako informační nástroj, je už dnes velmi málo. To je důvodem, proč dnes na všech školách učitelé vysvětlují a učí počítačovou gramotnost a programování. Cílem není jen zvládnutí programování ve vybraném programovacím jazyku, ale hlavně pochopení algoritmizace a naučení se logickému a programátorskému myšlení. Důraz na přemýšlení obecně nad algoritmy a logickou stavbou programu je důležitější než znalost samotného programovacího jazyka, neboť až studenti přijdou do praxe, tak je kolikrát je pro ně aktuální úplně jiný programovací jazyk, než který byl vyučován. K takovéto výuce stačí jednoduchý programovací jazyk s několika příkazy a cykly. Do této skupiny patří také programovací jazyk Karel The Robot (dále robot Karel), vyvinutý Richardem E. Pattisem, Stanford University, California, USA. V úvodních hodinách kurzů programování na FAI UTB se také používá tento jazyk robot Karel, kde jeho původní vývojové prostředí neposkytuje usnadnění programování např. pomocí debuggeru zdrojového kódu, o predikci psaní kódu, využívání knihoven s funkcemi ani nemluvě, tak jak dnešní nástroje vývojových prostředí nabízí.

Tato bakalářská práce v teoretické části poskytuje rešerši různých realizací robota Karla, grafické prostředí a v praktické části se zabývá návrhem knihoven funkcí robota Karel, napsaných v jazyce Java. To umožní výuku studentů v syntaxi jazyka Java a řešení úloh algoritmizace, a to i úplným začátečníkům. Knihovna funkcí na této bázi nabízí řešení algoritmů úkolů pro robota Karla jak jednoduchých, tak složitějších. Popsané příklady vedou studenta k pochopení algoritmizace a vytváření programů.

I. TEORETICKÁ ČÁST

1 VÝUKOVÝ JAZYK KAREL

Robot Karel je programovací jazyk, který je navrhnut jen s jednoduchým programovacím prostředím, s několika příkazy a jednoduchou syntaxí jako studijní nástroj pro porozumění a osvojení elementárních principů programování.

1.1 Historie

Robot Karel přišel na svět díky Richardu E. Pattisovi, určený jako nástroj pro výuku programování robotů na Stanfordově univerzitě ve Spojených státech od konce 70. let 20 století. Při vývoji tohoto nástroje vycházel autor nejen ze znalosti jazyka Logo, cíleného pro výuku programování, ale i z principů stavby programovacího jazyka Pascal.
Název robot Karel, anglicky Karel The Robot, použil autor jako poctu českému spisovateli Karlovi Čapkovi a jeho divadelní hře R. U. R., kde je poprvé použito slovo robot jako pojmenování stroje s inteligencí (nejspíš od slova robota, ale to jsou jen dohady).
Robot Karel se záhy, i díky v té době populárním mikropočítačům, velmi rychle rozšířil po celém světě a v různých, ať základních či modifikovaných podobách je populární a používá se dodnes na různých HW i SW platformách pro výuku základů programování.

1.2 Koncepce

Autor robota Karla, Richard E. Pattis definoval a popsal svou původní koncepci jazyka robot Karel v knize „Karel The Robot: A Gentle Introduction to the Art of Programming“ [1] z roku 1981.

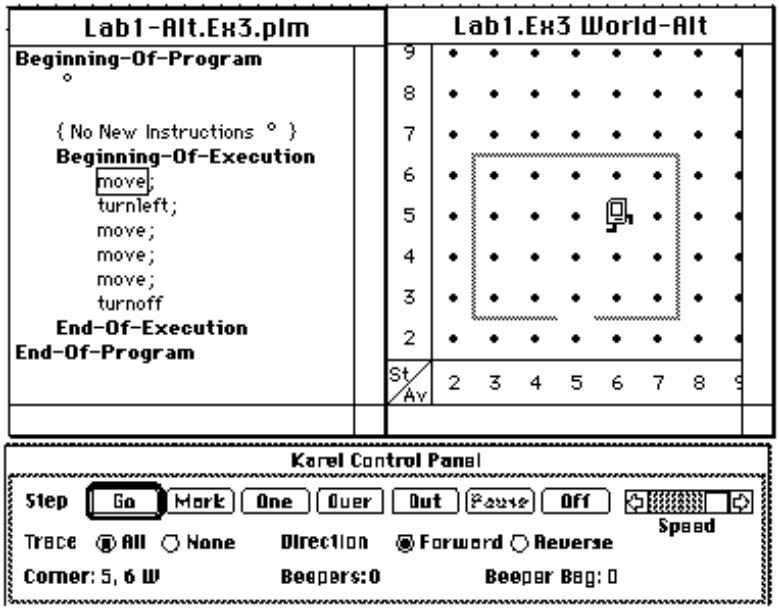
1.2.1 Kdo je robot Karel, kde žije a jak se pohybuje

Robot Karel je inteligentní stroj, který je ovládán příkazy k činnosti v rámci svého pracovního prostoru. Pracovní prostor je definován malým vymezeným čtvercovým dvourozměrným polem - městem (town), kde jeho pohyb je možný jen ve vodorovném (streets) a svislém (avenues) směru. Pokud stojí, tak je vždy na průsečíku vodorovného a svislého směru (corner) a může se otáčet. Pohyb je limitován jednak hranicemi pracovního prostoru a jednak překážkami v podobě objektu zdi (wall). Robot Karel také může sbírat nebo pokládat objekt bzučák (beeper) [2].

Základními příkazy robot Karel může vykonat vždy jednu událost:
move – robot Karel provede posun o jeden průsečík vpřed ve směru, ve kterém je natočen
turnLeft – robot Karel se otočí o 90° vlevo
pickBeeper – robot Karel zvedne jeden bzučák na průsečíku, na kterém stojí
putBeeper – robot Karel položí bzučák na průsečík, na kterém stojí,
turnOff – robot Karel se vypne

Robot Karel může testovat, zda má ve svém směru překážku – zeď, nebo pod sebou na průsečíku má bzučák, ve kterém směru stojí. Programováním zmíněných základních příkazů a konstrukcí podmínek, cyklů, rekurze student robota Karla naučí řešit složitější problémy. Například: orientovat se ve městě - umožní mu pohyb podél zdí, resp. hranic města, umožní mu sbírat/pokládat bzučáky.

Originální verze vývojového prostředí robota Karla (Obr. 1.)



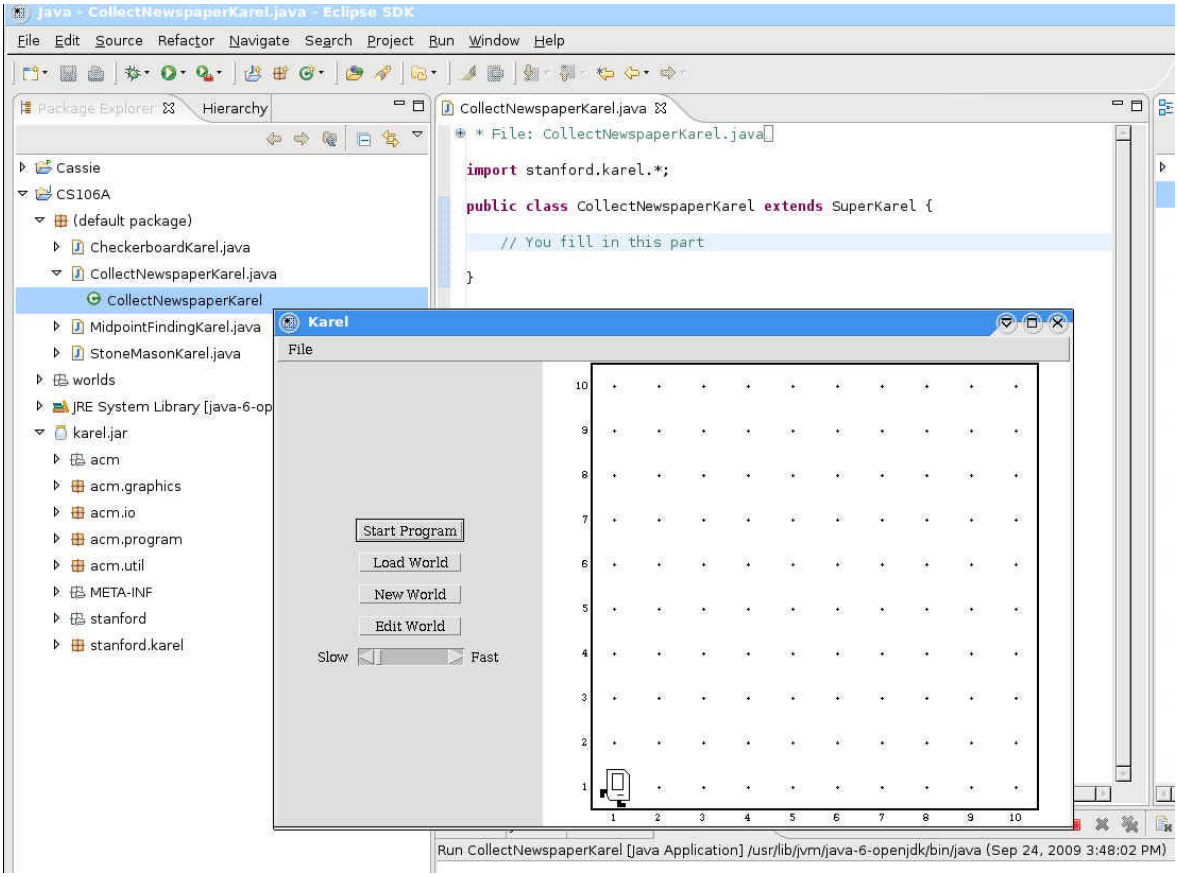
Obr. 1. Originální verze vývojového prostředí robota Karla [1].

1.3 Rešerše a zhodnocení realizací jazyka Karla

Robot Karel byl a stále je inspirací pro další programování a také díky mnoha zaníceným programátorům po celém světě vznikly nové implementace jazyka robota Karla na různých platformách v různých programovacích jazycích. Co je v tomto případě fenoménem, je celosvětově výjimečná popularita robota Karla u nás v České republice.

1.3.1 Karel v zahraničí

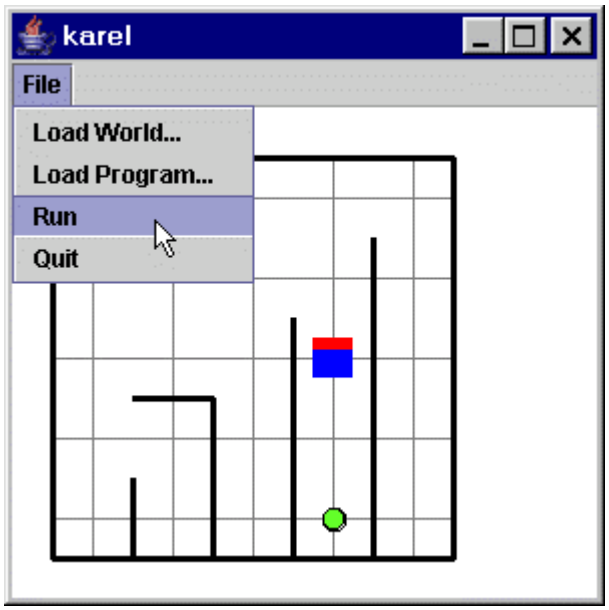
Původní implementace na Stanfordské univerzitě byla oživena implementací robota Karla v programovacím jazyce Java a jako knihovna je k dispozici ve vývojovém prostředí Eclipse interně na této univerzitě pro úvodní výuku objektově orientovaného programování. Dokumentace je na [2]. Projekt Karel je ke stažení na [3]. Náhled na grafické rozhraní robota Karla (Obr. 2.).



Obr. 2. Vývojové prostředí Eclipse s použitím knihovny Karel Stanfordské univerzity pro jazyk Java [3].

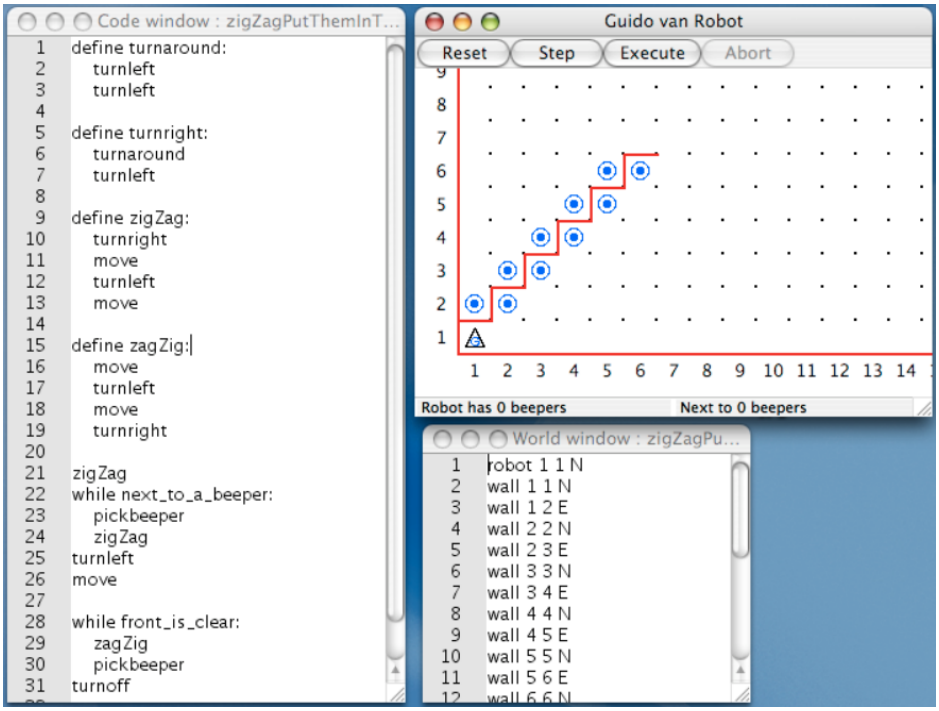
z University of Waterloo, Kanada. Zde je také v programovacím jazyku Java implementována knihovna simulující jazyk robot Karel, která se používá ve výuce v kurzu počítačových věd CS1[4]

KarelJ [5] je opět řešen v jazyku Java, je multiplatformní a je používán na Seidenberg School Of Computer Science and Information Systems, PACE University. Pohled na aplikaci, (Obr. 3.).



Obr. 3. Aplikace KarelJ

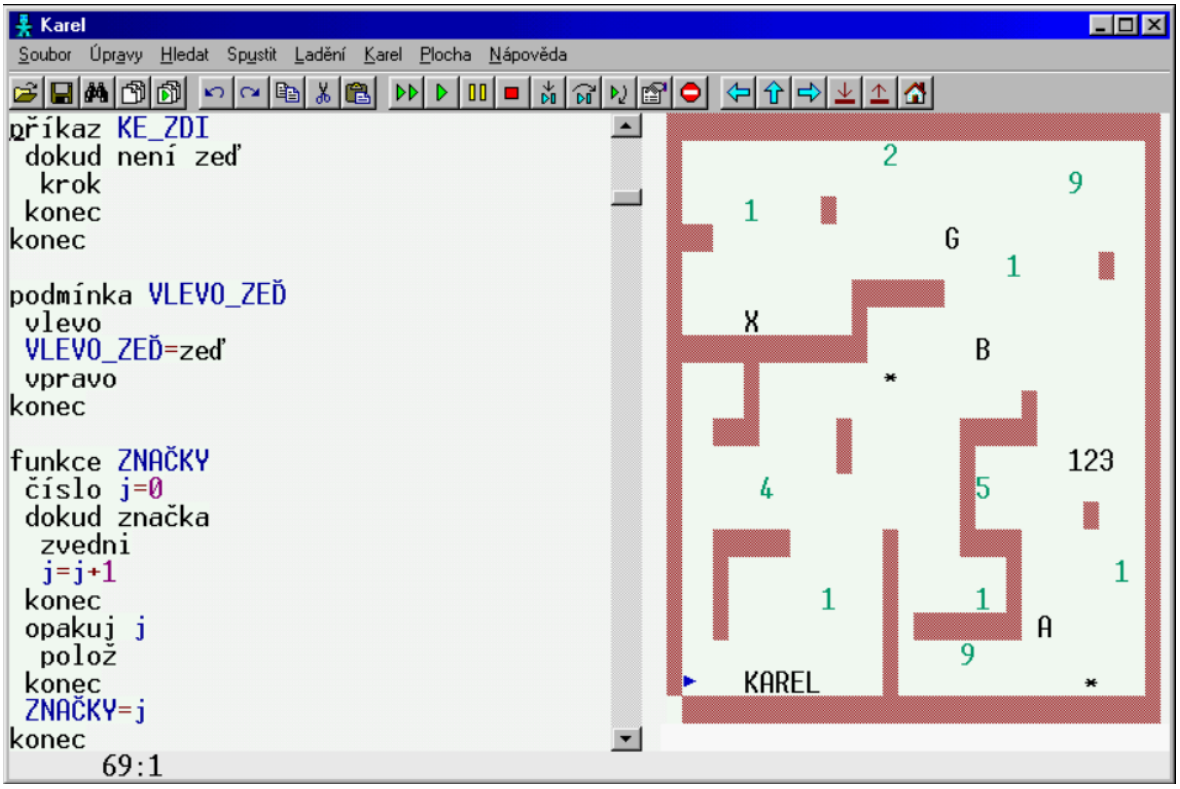
Guido van Robot je studentský projekt [6] (Yorktown High School in Arlington, Virginia), který je napsán v prostředí jazyka Python a je multiplatformní. Vývojové prostředí Guido van Robot, (Obr. 4.).



Obr. 4. Vývojové prostředí Guido van Robot.

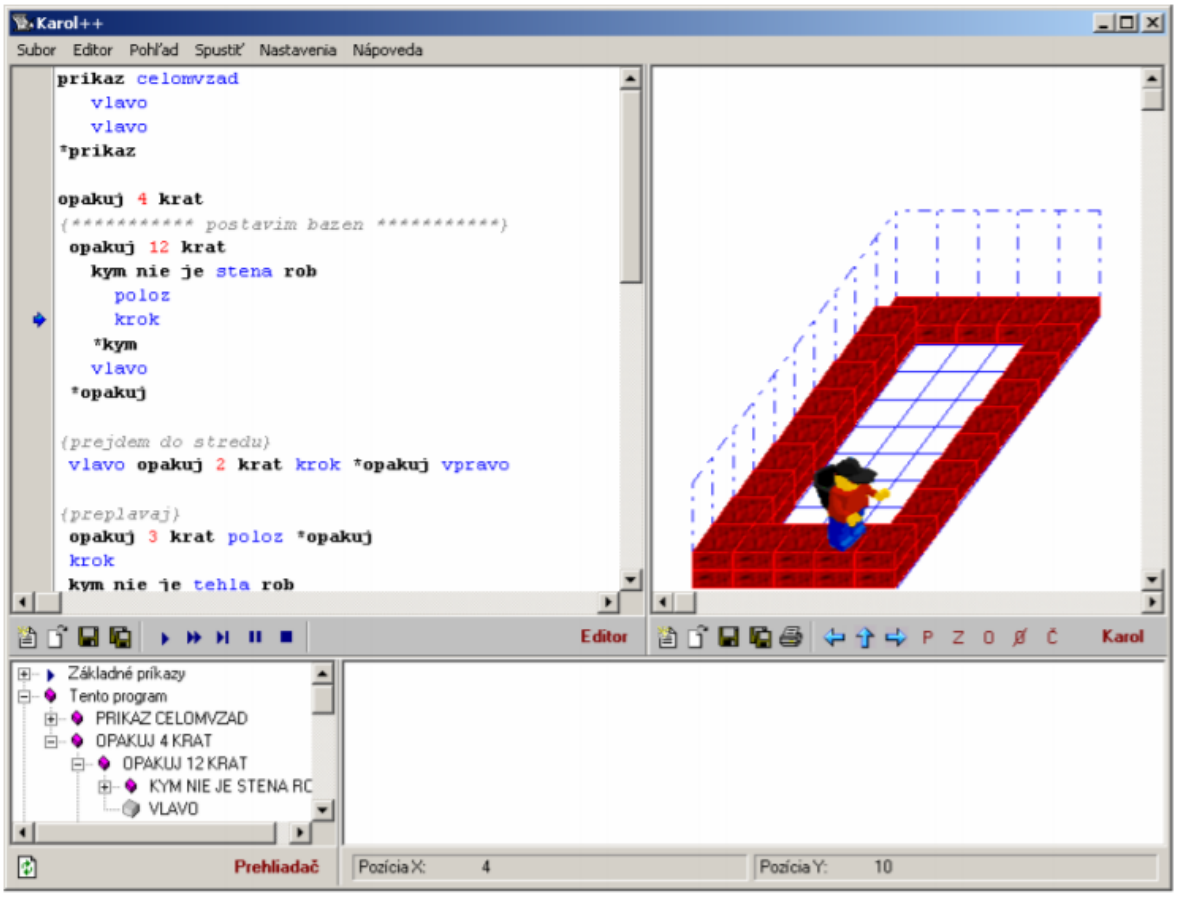
1.3.2 Karel u nás a na Slovensku

Robot KAREL 4.2 byl vytvořen Petrem Laštovičkou v C++ [7]. Tento robot Karel je vybaven mimo základních příkazů také možností používání celočíselných proměnných, vícerozměrná pole, aritmetické a logické operace, procedury a funkce s parametry. Program má vestavený textový editor, kompilátor, debugger a interpret přeloženého kódu. Ukázka programovacího prostředí Karel od Petra Laštovičky je na obrázku (Obr. 5.).



Obr. 5. Programovací prostředí Petra Laštovičky

Karol++ je ročníkový projekt napsaný v C++ Ondreje Kršky na Univerzitě Komenského v Bratislavě, Slovensko. Krškův projekt má moderněji vytvořený editor se zvýrazněním syntaxe a zobrazuje robota v 3D náhledu [7]. Krško tak navazuje na projekt Karel3D svého učitele Marána Vitteka z 80-tých let 20. století. Krška později pro svou diplomovou práci vytvořil projekt Robot Karol.NET pro různé platformy operačních systémů Microsoft [7], ale není šířený mezi uživatele.



Obr. 6. Karol++

Existuje mnoho dalších realizací, které reprezentují vývoj a používání původní myšlenky jednoduchého programování R. E. Pattisse, které zde nejsou zmíněny. Robot Karel se stále vyučuje, především na školách v USA, ale i jinde ve světě, i u nás. O popularizaci tohoto principu výuky se v 80-tých letech minulého století se v bývalém Československu zasloužil R. Pecinovský, T. Bartovský, M.Vittek a dalšími jejich pokračovateli v popularizaci jsou např. I. Ryant, O. Jedlička, který vytvořil populární a používanou internetovou aplikaci Karel v Javascriptu. Robot Karel prošel za více než třicetiletou historii vývojem od procedurálního k objektovému pojetí výuky programovacích jazyků. Já jsem se v této bakalářské práci zaměřil především na implementace, které jsou na bázi objektově orientovaných jazyků.

2 KAREL V JAZYCE JAVA

Karel je programovací jazyk, který byl vytvořen za účelem pochopení základů programování a osvojení základních algoritmu.

2.1 Java pro Karla

Robot Karel využívá standartní syntaxi jazyka Java pro větvení, cyklení a tvoření funkcí. Protože algoritmy vytvořené s příkazy robota Karla, budou funkční i v standartními Jave.

II. PRAKTICKÁ ČÁST

3 NÁVRH ROBOTA KARLA V JAZYCE JAVA

Návrh knihovny robota Karla vytvořené v rámci bakalářské práce umožňuje v běžném vývojovém prostředí jazyka Java vytvořit vizuální podobu města pro robota Karla. Robot Karel se v tomto městě ovládá základními funkcemi knihovny, které zprostředkovávají pohyb a poskytují robotovi informace o dění kolem něj. Uživatel má tak názorné prostředí a představu o tom, jak se má robot ve svém městě pohybovat a chovat, aby splnil zadaný úkol. V jazyce Java se student učí, jak má robot daný úkol řešit. Začátečníkům knihovna poskytne názornější představu o tom, jak se liší myšlení v reálném světě a ve světě softwaru. Podle obtížnosti úkolu, který má robot řešit, student uplatňuje odpovídající obtížnost konstrukcí jazyka Java.

3.1 Popis ovládání robota Karla v jazyce Java

V této kapitole bude popsána upravená verze jazyka robot Karel pro nasazení na výuku jazyka Java. Robot Karel se může volně pohybovat po svém městě. Musí ale dávat pozor, aby nenarazil do zdí, kterými je jeho město ohraničeno, případně členěno. Na pozici, na níž aktuálně stojí, může umístit zelenou kostičku jako značku. Těchto kostiček může na sebe naskládat maximálně 6. Ve městě má Karel také svůj vlastní domeček. Funkce knihovny, sloužící k ovládání robota Karla uživatelem, se nazývají příkazy. Karel zná čtveřici příkazů k vykonávání pohybu. Karel umí vyhodnocovat 10 podmínek, které umožňují uživateli získat informace o tom, co se kolem Karla nachází.

3.1.1 Vytvoření Karlova Města

Celé město robota Karla se vytváří v souboru main.java v třídě `Main` a hlavní spouštěcí funkci `public static void main(String[] args)` zde vytvoříme novou instanci třídy Karel (`Karel karel = new Karel();`). Instance Karla musí být provedena na začátku. Bez ní Karel neví, kde se nachází a jak vypadá jeho město. Při nevytvoření této instance nemůže robot Karel vykonávat svoje instrukce. Při vložení textového řetězce s názvem souboru, kde je uložené Karlovo město se podle textového souboru vytvoří město. Vytvoření instance by tedy vypadalo `Karel karel = new Karel(„soubor.txt“);`. Nedojde-li k správnému nalezení textového souboru, město se vytvoří prázdné s Karlem na pozici 1,1. Karlovo město jde také vybudovat s určitou šířkou a výškou. To lze docílit zapsání počtu polí do šířky a výšky do konstruktoru následovně `Karel karel = new Karel(početPolíčekNašířku, početPolíčekNaVýšku);`.

Následovným zavolaní funkce `karel.začni()` je inicializováno a vybudováno grafické prostředí Karlova města.

3.1.2 Popis základních příkazů

Krok

Funkce je deklarována: `public void krok()`.

Pomocí tohoto příkazu se Karel posune o jedno políčko (baňku) dopředu. Pokud je před Karlem zeď, tak do ní narazí a program skončí chybou. Tuto situaci je potřeba ošetřit pomocí podmínek popsaných v následující podkapitole.

Otočení vlevo

Funkce je deklarována: `public void vlevoVbok()`.

Tento příkaz je zkrácenou verzí TURNLEFT z původní verze Karla. Tento příkaz neposune Karla o jedno políčko doleva, jak by si někdo mohl myslet, nýbrž ho otočí na stejném poličku o 90° směrem doleva.

Položení značky

Funkce je deklarována: `public void polozZnacku()`.

Slouží k položení značky na políčko, na které Karel právě stojí. Na jedné buňce muže být položeno maximálně 6 značek. Pokud jsou vkládány značky na zaplněné políčko, tak program skončí s chybou. Zaplnění příslušného políčka lze testovat funkcí `pocetZnacek()`, viz. dále.

Zvednutí značky

Funkce je deklarována: `public void zvedniZnacku()`.

Přikáže Karlovi zvednout jednu vrchní značku z hromádky značek položených na políčku, na kterém se Karel nachází. Pokud se Karel pokusí zvednou značku na políčku, na níž žádné značky nejsou, tak program skončí chybou. Před použitím tohoto příkazu je vhodné otestovat, zda-li jsou na daném políčku značky. K tomu slouží podmínka jeZdeZnačka() popsaná v následující kapitole.

Počet značek

Funkce je deklarována: `public int pocetZnacek()`.

Za pomoci této funkce je Karel schopný určit přesný počet značek na políčku, na kterém Karel právě stojí. Funkce vrací celočíselnou hodnotu indikující množství značek.

3.1.3 Podmínky

Podmínky slouží k větvení programu. Pomocí nich se může robot Karel reagovat na určitou situaci a rozhodnout, jestli půjde tou či onou cestou. Určitou situací je myšleno, zdalipak zkoumaná podmínka je pravdivá či nikoliv. Pokud podmínka je pravdivá, tak se vypraví jednou větví programu. Pokud je nepravdivá, procházíme druhou větví programu. Syntaxe podmínek je u našeho robota Karla totožná se syntaxí Javy a to následující:

```
if (podmínka) {
       příkazy
} else {
       příkazy
}
```

Za klíčovým slovem `IF` (když) je v závorkách uvedena podmínka, která se testuje, zde je pravdivá nebo není pravdivá. Jestliže je podmínka pravdivá, tak se provedou příkazy ve složených závorkách za podmínkou. Pokud je podmínka vyhodnocena za nepravdivou provedou se příkazy ve složených závorkách za klíčovým slovem `ELSE` (jinak).

Další varianta zápisu podmínky je tato:

```
if (podmínka) {
       příkazy
}
```

V tomto případě se při pravdivosti podmínky provedou pouze příkazy ve složených závorkách za podmínkou. Jestliže uvedená podmínka není pravdivá, pak se nic neuskuteční. Struktura této varianty je ekvivalent předcházející variantě, ve které nejsou ve složených závorkách za klíčovým slovem ELSE žádné příkazy.

Popis podmínek jazyka robota Karla.

Zeď před Karlem

Funkce podmínky je deklarována: `public boolean jePredemnouZed()`.

Podmínkou Karel ověřuje, zda je na políčko před ním zeď. Je-li Karel na políčku a je otočen určitým směrem, že políčko před ním je zeď, tak je vrácena hodnota TRUE (pravda). V opačném případě je vrácena hodnota FALSE (lež). Za zeď je v Kartovém světě považovány i kraje města. Podmínku lze využít ke zjištění zdi Karlem, aby mohl náležitě odpovědět např. otočení.

Zeď nalevo od Karla

Funkce podmínky je deklarována: `public boolean jeVlevoZed()`.

Podmínkou Karel ověřuje, zda je na políčku vlevo od směru, ve kterém je natočen, je zeď. Pokud je na tomto políčku zeď funkce vrátí hodnotu TRUE, jinak je vrácena hodnota FALSE.

Zeď napravo od Karla

Funkce podmínky je deklarována: `public boolean jeVpravoZed()`.

Podmínkou Karel testuje, zda je napravo od směru, ve kterém je natočen, je zeď. Jestliže je na tomto políčku zeď, funkce vrací hodnotu TRUE. V opačném případě je vrácena hodnota FALSE.

Dále negace podmínek

Volno před Karlem

Funkce podmínky je deklarována: `public boolean jePredemnouVolno()`.

Podmínka je opakem podmínky `jePredemnouZed()`. Jestli že není políčko před Karlem v jeho směru zeď funkce vrací hodnotu TRUE. V opačném případě je vrácena hodnota FALSE.

Volno nalevo od Karla

Funkce podmínky je deklarována: `public boolean jeVlevoVolno()`.

Podmínka je negací funkce `jeVlevoZed()`. Podmínkou Karel ověřuje, zda je na políčku vlevo od směru, ve kterém je natočen, je volno. Pokud na tomto políčku není zeď funkce vrátí hodnotu TRUE, jinak je vrácena hodnota FALSE.

Volno napravo od Karla

Funkce podmínky je deklarována: `public boolean jeVpravoVolno()`.

Podmínka je opakem funkce `jeVpravoZed()`. Podmínkou Karel testuje, zda je napravo od směru, ve kterém je natočen, je volno. Jestliže na tomto políčku není zeď, funkce vrací hodnotu TRUE. V opačném případě je vrácena hodnota FALSE.

Zde následuje detekce na políčkách.

Zjištění značky

Funkce podmínky je deklarována: `public boolean jeZdeZnacka()`.

Touto podmínkou Karel testuje, zda-li je na políčku, kde se aktuálně nachází, je jedna nebo více značek. Funkce vrátí hodnotu TRUE, pokud se na Karlové pozici nachází alespoň jedna nebo více značek. V opačném případě vrátí hodnotu FALSE.

Zjištění domečku

Funkce podmínky je deklarována: `public boolean jeZdeDomeček()`.

Podmínkou Karel zkouší, jestli na políčku, na kterém stojí, je jeho domeček. Pokud Karel se nachází ve svém domečku, funkce vrací TRUE. Jinak je vrácena hodnota FALSE.

Detekce natočení robota Karla

Funkce jednotlivých podmínek jsou deklarována:

```
public boolean jeOtočenýNaSever().
public boolean jeOtočenýNaJih().
public boolean jeOtočenýNaVychod().
public boolean jeOtočenýNaZapad().
```

Za pomoci těchto podmínek může Karel zjistit, který světovou stranu je orientován. Jestliže je Karel natočen na patřičnou stranu funkce vrátí hodnotu TRUE. Pokud je zorientovaný na jakoukoliv jinou stranu, návratová hodnota je FALSE.
Např. pro funkci jeOtočenýNaSever(). Pokud je natočený na sever funkce vrátí TRUE. Jinak je návratová hodnota FALSE.

3.1.4 Cykly

Cyklus slouží k opakování v cyklu uzavřených příkazů. Příkazy se budou vykonávat stále dokola, dokud bude platná podmínka na začátku cyklu. V běžné praxi se používají tři typy cyklů. A to cyklus s podmínkou na začátku, cyklus s podmínkou na konci a cyklus s předem známým počtem opakování.
Cyklus s podmínkou na začátku funguje tak, že se nejprve testuje podmínka a pokud platí, tak se provedou příkazy v cyklu. Pokud podmínka neplatí, cyklus skončí. To znamená, že pokud je podmínka neplatná před vykonáním cyklu, tak se příkazy uvnitř cyklu neprovedou. Opakem je cyklus s podmínkou na konci. V tomto případě se nejprve provedou příkazy v cyklu, a pak se testuje podmínka. Při platnosti podmínky se cyklus provede znovu, jinak cyklus končí. Z toho plyne, že tomto typu cyklu se příkazy provedou minimálně jednou, kdežto v cyklu s podmínkou na začátku se příkazy nemusí provést vůbec.
Posledním typem cyklů je cyklus s předem známým počtem opakování. Implementací tohoto cyklu jsou tzv. FOR cykly, které fungují tak, že se na začátku cyklu inicializuje proměnná, např. i=0 a s každým průběhem cyklu se tato proměnná inkrementuje. Cyklus se provádí tak dlouho, dokud se proměnná i nerovná předem zadané hodnotě. V implementaci Karla pro výuku Java jsou využity nativní cykly jazyka Java.
Nejprve si popíšeme syntaxi cyklu s podmínkou na začátku.

```
while (podmínka) {
       příkazy
}
```

Jak bylo již výše zmíněné, jako první se vyhodnotí podmínka za klíčovým slovem `WHILE`. Když je podmínka platná, tak se vykonají příkazy ve složených závorkách. Pokud podmínka platná není příkazy se neprovedou a program bude pokračovat za nimi.
Dále si popíšeme syntaxi cyklu s podmínkou na konci.

```
do {
	příkazy
} while (podmínka);
```

Zde se nejprve provedou příkazy ve složených závorkách před klíčovým slovem `WHILE` a až poté se vyhodnotí podmínka v závorkách. Pokud je podmínka vyhodnocena pravdivě, jsou příkazy ve složených závorkách za klíčovým slovem `DO` provedeny znova. Jestliže je podmínka vyhodnocena jako nepravda, program pokračuje dál.
Nakonec si vysvětlíme cyklus s předem daním počtem cyklů.

```
for (int i = 0; i < PočetCyklů; i++) {
	příkazy
}
```

Cyklu for zopakuje příkazy ve složených závorkách přesně tolikrát, jaká je hodnota PocetCyklu.

3.1.5 Funkce

Funkce zavádíme pro zjednodušení zdrojového kódu. Umožní mám rozdělit rozsáhlé problémy na menší podproblémy. Podproblémy lze po jednom napsat jako jednotlivé funkce, které po složení řeší daný rozsáhlí problém. Další výhodou funkcí je jejich možnost opakování a lze je v programu využít víckrát. Výsledkem je kratší a jednodušší zdrojová program.
Funkce se definuje následovně.

```
viod názevFunkce() {
	příkazy
}
```

V programu musí existovat funkce s názvem `main`. Tato funkce je hlavní a je volaná při zpuštění programu. Z funkce main se pak volají další funkce.

3.1.6 Rekurze

Rekurze je speciální případ volání funkce, kdy je volanou funkcí volající funkce, tedy rekurzí se rozumí volání sebe sama. Rekurzi lze dělit na dva základní typy, a to na přímou a nepřímou rekurzi [14].

Přímá rekurze pracuje tak, jak bylo popsáno výše. Procedura volá sebe sama. Příkladem může být např. tento kód.

```
viod kroky() {
	if (jePredemnouVolno) {
		krok();
		kroky();
	}
}
```

Podmínce v ukázkovém kódu se říká ukončovací podmínka. Ukončovací podmínka je pro správnou činnost rekurze nutná. Pokud by tam nebyla, program by v tomto případě skončil chybou při nárazu Karla do zdi. V jiných případech by mohlo dojít k zacyklení a následnému zaplnění dostupné paměti v zásobníku volání uvedeného dále.

Nepřímá rekurze vzniká tak, že se dvě a více funkcí volají navzájem a tato volání uzavřou kruh. Příklad pro dvě funkce může být následující.

```
viod prvni() {
	if (jePredemnouVolno) {
		druha();
	}
}
viod druha() {
	krok();
	první()
}
```

Tento ukázkový příklad dělá to samé jako předcházející příklad, avšak využívá nepřímé rekurze.

3.2 Popis ovládání Karlova města

Karlovo město je čtvercová nebo obdélníková síť, ve které se robot Karel pohybuje po jednotlivých políčkách (buňkách). Město může být různě členěno pomocí zdí, přes které Karel nemůže projít a musí se jim vyhýbat, jinak do nich narazí. Samotnou hranici města Karel vnímá jako zeď a nemůže ji nikdy překročit. Město je orientováno tak, že jeho severní hranice je identická s horní lištou okna. Na všech políčkách města, na nichž není zeď, se mohou nacházet značky. Tyto značky má Karel možnost pokládat či sbírat. Maximální možný počet značek na jednom políčku je 6. Poslední objekt vyskytující se v Karlově městě je jeho domeček. Domeček musí být ve městě bezpodmínečně umístěn. Slouží jako označení místa ve městě, kde se Karel po spuštění programovacího rozhraní

nachází, pokud není definována jeho jiná pozice. Domeček může také sloužit jako označení bodu ve městě, kam má Karel dojít. Město po vykreslení nemůže být větší, než je nastavené rozlišení obrazovky počítače. Maximální počet políček města (velikost města), jež mohou být vykresleny, je dán jejich samotnou velikostí. Základní velikost políčka je nastavena na 40 pixelů. Minimální velikost města je jedno políčko.

3.2.1 Mapa města

Pomocí tzv. mapy si uživatel může vytvořit vlastní podobu města, ve kterém se bude robot Karel pohybovat. Mapu města si uživatel dodržením určitých pravidel může vytvořit v textovém souboru nebo v navrhnutým grafickým prostředí. Aby se robot Karel v takto vytvořeném městě mohl pohybovat, je nutné v programu použít konstruktor Karel(„názevSouboru“) a jako vstupní parametr uvést název souboru, který musí být uložen ve složce saves.

Vytvoření mapy v textovém editoru

Za pomocí speciálních znaku lze vytvářet mapu jednotlivých měst v libovolném textovém editoru. Tyto speciální znaky zastupují jednotlivé prvky z Karlova města.

Znaky zastupující jednotlivé prvky města jsou:

- `X` pro hranice města
- `#` pro zeď
- `K` pro Karla
- `H` pro Domeček
- `1-6` pro Značky
- _ pro prázdné políčko

Levý horní roh hranice města musí být reprezentován prvním znakem na prvním řádku v textovém souboru. První řádek souboru (pouze znaky X) znázorňuje severní hranici města. Další řádky souboru znázorňují město samotné. Aby byla jasně stanovena západní a východní hranice města, musí tyto řádky začínat a končit znakem X. Poslední řádek pak reprezentuje jižní hranici města a je identický s prvním řádkem souboru (pouze znaky X). Všechny řádky znázorňující hranice města a město samotné musí mít stejný počet znaků. Tyto řádky se neskenují jako mapa města. Uvnitř města se znaky zastupující zeď, Karla, domeček a značky mohou použít libovolně.

Výsledná mapa města v textové formě může vypadat např. následovně:

```
XXXXXXXXXXXX
X5###2__1#HX
X_#1###_##_X
X__________X
X_#####_##_X
X5#___#__#2X
X##_#_1#__#X
X#__#_#3#_#X
X6_##_#___#X
X###___##_#X
XK___#____#X
XXXXXXXXXXXX
```

Grafické znázornění této textové mapy Karlova města bude znázorněné v další podkapitole.

Vytvoření mapy v grafickém rozhraní Karla

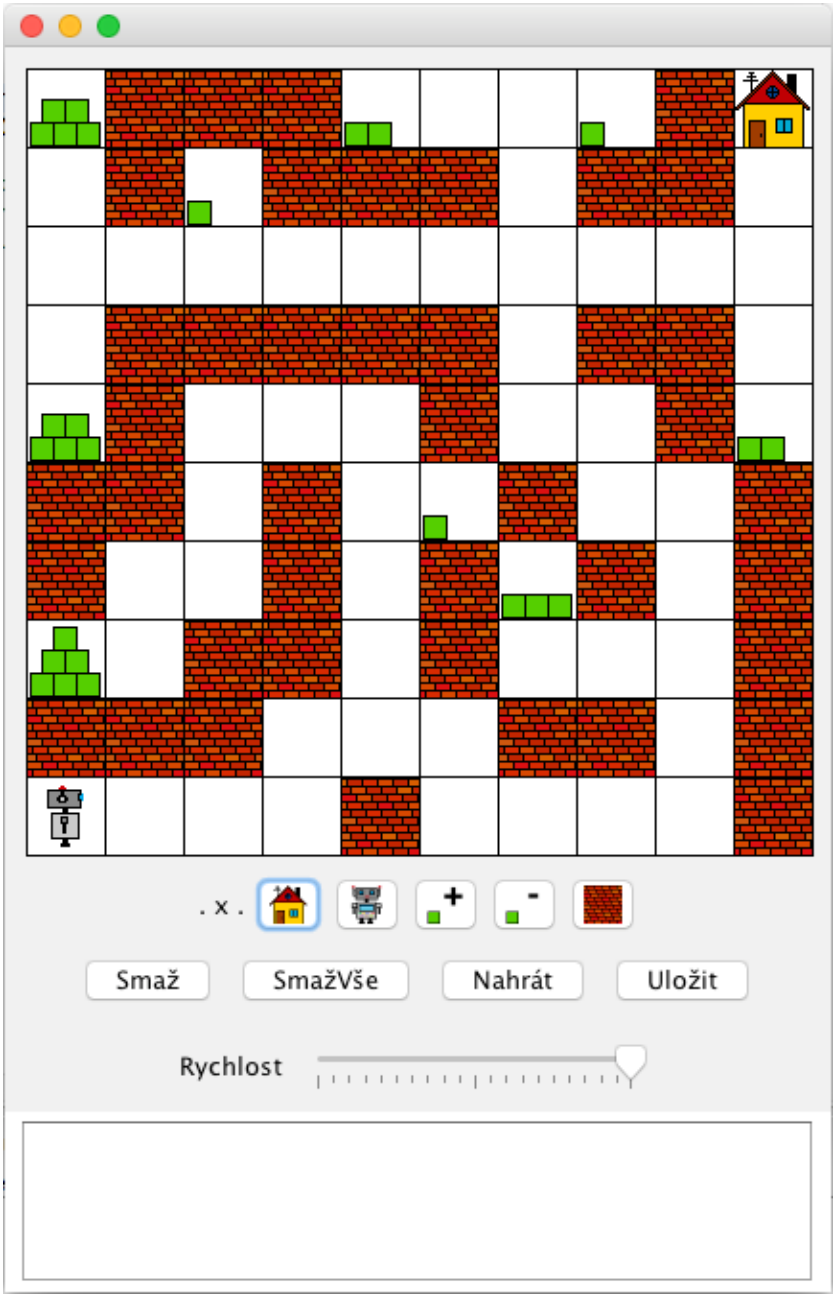
Pro vytváření mapy města v grafickém prostředí robota Karla je zapotřebí vytvořit instanci třídy Karel v hlavního programu robota Karla ve zdrojovém souboru main.java. Ještě je nutné zavolat funkci začni. Výsledná funkce main by měla vypadat následovně.

```
public static void main() {
       Karel karel = new Karel(x,y);
       karel.zacni();
       }
}
```

Kde vstupní parametr v konstruktoru `x` určuje počet políček do šířky a y je počet políček na výšku.

Takto vytvořený program robota Karla zajistí, že Karel nebude během stavění město měnit.

Zobrazení grafického prostředí vypadá následovně.



Obr. 7. Zobrazení GUI

Pod městem jsou tlačítka nástrojů pro editaci mapy Karlova města. Postup editace je následující. Nejprve stiskneme tlačítko pro výběr charakteru políčka. Následným kliknutím do mapy města na požadovaných souřadnicích se umístí vybraný objekt.
První tlačítko na prvním řádku tlačítek pod městem s vyobrazeným domečkem slouží k umístění Domečku. Protože Domeček může být jen jeden, domeček na předchozích souřadnicích se nahradí prázdným pole. Druhé tlačítko v pořadí s Karlem slouží k přemísťování Karla po mapě města. Třetí a čtvrté tlačítko slouží k pokládání a zvedání

značek. Všechny tyto objekty nemůžou být umístěny na políčko, kde se již nachází zeď. Posledním tlačítkem v řadě je zeď.

Další řada tlačítek slouží k mazání, nahrávání a ukládání mapy města. První políčko v druhé řadě slouží k mazání jednotlivých polí. Po kliknutí kurzorem myši na zvolenou souřadnici města, se dané políčko nahradí prázdným. Tlačítko SmažVše vrátí celou mapu pole do počátečního stavu. Tlačítko Uložit vypíše aktuální mapu města do textové podoby a uloží jej do souboru karel.save. Uložený soubor karel.save se bude nacházet v projektovém adresáři ve složce saves. Tlačítko Nahrát vytvoří Karlovo město ze souboru z textovou mapou světa nazvaný karel.save.

4 IMPLEMENTACE ROBOTA KARLA

Knihovna robota Karla vytvořená v rámci práce se skládá ze zdrojových souborů Karel.java, Main.java a adresáře KarelImages a Saves. Ve zdrojovém souboru Karel.java jsou obsaženy všechny dostupné programové a uživatelské funkce knihovny, které se podílí na správném fungování programového rozhraní knihovny. V Main.java je pak hlavní funkce main, ve které se nachází Karlů program. Třída Karel ze zdrojového souboru Karel.java musí být vložena do uživatelského programu, tedy do Main.java, direktivou import. Adresář KarelImages obsahuje textury, které jsou potřebné pro správné vykreslování grafického rozhraní robota Karla a adresář Saves obsahuje textové mapy měst robota Karla. Oba adresáře musí být umístěny ve výstupní složce uživatelského projektu.

4.1 Instalace knihovny robota Karla

Pro využití knihovny robota Karla, je zapotřebí vložit do projektu, ve kterém se bude knihovna využívat, zdrojové soubory Karel.java a Main.java. Do Main.java musí být importována Třída Karel. Karel může být používán v kterémkoliv vývojovým prostředí pro jazyk Java. V každém vývojovém prostředí se umístění obsahu nastavení různí, ale potřebné body nastavení projektu pro jeho správný chod zůstávají stejné.

4.1.1 Vývojové prostředí Eclipse

Jelikož je programovací jazyk robot Karel určený pro začátky programování v jazyce Java, je kladen důraz, aby uživatel mohl začít programovat v jazyce robot Karel co nejdříve. Eclipse vhodné vývojové prostředí, protože disponuje funkcí importu projektu. Uživateli pak stačí jenom importovat adresář.

4.2 Popis vnitřních struktur třídy Karel

4.2.1 Výčtové typy enum

Výčtový typ `Typ` označuje význam objekt umístěných na políčku. Složí k názornému určení objektu, který se na dané buňce nachází.
Dalším výčtovým typem je `SelectedButten`. Ten slouží k názornému určení, které editovací tlačítko je zrovna vybrané.
Poslední z výčtových typů je `Směr`, která slouží k přehlednému určení orientace robota Karla.

4.2.2 Třídy a důležité metody

Třída `Field` slouží k určení, zda-li se na políčku nachází zeď nebo značka. Také je zde určen počet značek na jednom políčku.

4.2.3 Proměnné

V proměnných `karelx` a `karely` jsou aktuální souřadnice, na kterých karel momentálně stojí.
Souřadnice `dumx` a `dumy` jsou proměnné současné pozice Karlova domečku.
`kTown` je dvojrozměrné pole třídy `Field`. Velikost Polí je vždy shodná s rozměry Karlova Města. Podle tohoto pole se renderuje Karlovo město.

5 SADA PŘÍKLADŮ PRO VÝUKU ROBOTA KARLA

V této kapitole budou popsány příklady s řešením. Kapitola je rozdělena na 3 podkapitoly, z nichž se první zabývá příklady, pomocí kterých se uživatel seznámí s jazykem Karel a s používáním podmínek, cyklů, rekurze a definicí funkce. V další podkapitole se nacházejí příklady zaměřené na procvičení použití cyklů. Poslední podkapitola je věnována příkladům řešených nejlépe pomocí rekurze. Všechny příklady a řešení jsou v příloze bakalářské práce. Každý příklad obsahuje zadání a řešení, která mohou být doplněna obrázky a zdrojovými kódy.

5.1 Základní úlohy

V této podkapitole budou popsány příklady sloužící k seznámení se s programovacím jazykem robot Karel a jeho možnostmi.

5.1.1 První krok

Tato úloha slouží k procvičení základních příkazů jako je krok dopředu a pokládání značek.

Zadání: „Posuň Karla o jedno políčko dopředu a polož dvě značky.“ Robot stojí na buňce vlevo dole a je natočen na východ (doprava). Hrací pole neobsahuje žádné zdi ani značky.

Řešení: Na obrázku 14 je zobrazen funkční zdrojový kód a výsledné hrací pole po provedení tohoto kódu. Při řešení této úlohy je nutné dbát na správné poradí příkazů. Stejné příkazy v jiném pořadí by nesplnili zadání.

Zde je funkční kód řešení příkladu.

```
public static void main(String[] args) {
	Karel karel = new Karel();
	karel.zacni();
	karel.krok();
	karel.polozZnacku();
	karel.položZnacku();
}
```

5.1.2 Vyčisti políčko

V této úloze se uživatel lehce seznámí s podmínkami a cykly nebo rekurzí, podle způsobu řešení.

Zadání: „Posbírej všechny značky na políčku, na které stojí Karel. Zdrojový kód musí fungovat i pro jiný počet značek."

Řešení: Nyní se nabízejí dvě možnosti řešení. A to pomocí cyklu nebo rekurze.

a) Cyklus

```
public static void main(String[] args) {
	Karel karel = new Karel();
	karel.zacni();
	while (karel.jeZdeZnacka()) {
		karel.zvedniZnacku();
	}
}
```

b) Rekurze

```
Karel karel;
public static void main(String[] args) {
	karel = new Karel();
	karel.zacni();
	zvedni();
}
void zvedni() {
	if (karel.jeZdeZnacka()) {
		karel.zvedniZnacku();
		zvedni();
	}
}
```

5.1.3 Vytvoř čtverec

Cílem této úlohy je donutit uživatele zamyslet se nad předvyplněným zdrojovým kódem a upravit ho pomocí cyklu nebo rekurze.

Zadání: „Vykresli vlevo dole čtverec ze značek o rozměrech 4x4. Uprav zadaný zdrojový kód."

Předvyplněný zdrojový kód.

```
public static void main(String[] args) {
	Karel karel = new Karel();
	karel.zacni();
	if (!karel.jeZdeZnacka()) {
		karel.polozZnacku();
		karel.krok();
		karel.polozZnacku();
		karel.krok();
		karel.polozZnacku();
		karel.krok();
	}
}
```

Řešení: Předpokládané řešení bylo pomocí rekurze nebo cyklu, kde pro rekurzi stačilo vložit celý if do funkce a za poslední příkaz `krok()` přidat příkaz `vlevoVbok()` a rekurzivní volání této funkce. Pro řešení pomocí cyklu bylo nutné přepsat původní příkaz if na while a za poslední příkaz step přidat `vlevoVbok()`. V tomto příkladě převládá použití rekurze. Bylo zde i poměrně dost méně nápaditých řešení, jako je např. dopsat za poslední `krok()`, příkaz `vlevoVbok()`, kód zkopírovat a vložit ho čtyřikrát za sebe.

5.1.4 Plná ulice

Tento příklad je řešením podobný předchozím příkladům, ale uživateli navíc ukazuje možnost použití definice funkce.

Zadání: „Na každou buňku ve spodním řádku vlož maximální počet značek.“

Řešení: Opět se vyskytuje možnost řešení pomocí cyklu nebo rekurze. Nyní je pouze nutné počítat s tím, že např. po přepsání kódu na rekurzi se Karel zastaví před zdí, ale značky už nepoloží, protože přestane platit ukončovací podmínka. Proto je potřeba na konci funkce main zavolat proceduru `vypln()`.

```
Karel karel;
public static void main(String[] args) {
	karel = new Karel();
	karel.zacni();
	while (karel.jePredemnouVolno) {
		vypln();
		karel.krok()
```

```
        }
}
void vypln() {
        while (karel.pocetZnacka() < 6) {
                karel.polozZnacku();
        }
}
```

5.1.5 Do rohu

Tento úkol je zaměřen na procvičení používání cyklu.

Zadání: „Dostaň Karla do pravého horního rohu. Musí fungovat z jakéhokoliv místa.“ Karel se nachází ve městě bez zdí.

Řešení: Nejprve je nutné vyřešit směr natočení robota. K tomu lze využít cyklu s negací podmínkou jeOtocenNaSever(), kde se bude Karel otáčet doleva, dokud není natočen na sever. Dále je vhodné definovat funkce nazvanou keZdi, ve které Karel dojde k protější zdi. Když je robot natočen na sever, tak se do pravého horního rohu dostane jednoduše sekvencí volání keZdi, vpravoVbok, keZdi.

```
Karel karel;
public static void main(String[] args) {
        karel = new Karel();
        karel.zacni();
        while (!karel.jeOtocenyNaSever()) {
                karel.vleloVbok();
        }
        keZdi();
        vpravoVbok();
        keZdi();
}
void vpravoVbok() {
        karel.vleloVbok();
        karel.vleloVbok();
        karel.vleloVbok();
}

void keZdi() {
        while (karel.jePredemnouVolno()) {
                karel.krok();
        }
}
```

ZÁVĚR

Mým úkolem bylo vyhledat a charakterizovat existující implementace Robota Karel. To je popsáno v teoretické části bakalářské práce. Charakterizováno je zde řešení, které iniciovalo zájem o Robota Karel a některá další řešení, která jsou také populární a používají se, díky aktualizacím do současných programovacích jazyků, dodnes pro vyučování algoritmizace, programového myšlení studentů. Ne všechny nabízí přístup ke svým zdrojovým kódům.
V praktické části je naprogramován návrh implementace Robota Karel v jazyce Java. Ten kopíruje původní myšlenku zadání co nejjednoduššího programování Robota Karel v jazyce Java pro její výuku. K tomu je naprogramováno grafického rozhraní, jak pro editaci a ladění výukových programů Robota Karla, tak pro nastavení jeho města.
Pro inspiraci ve výuce Javy jsou v praktické části také připraveny příklady programů Robota Karla. Ty jsou navrženy tak, aby na nich mohly být testovány všechny úlohy algoritmizace, které jazyk Robot Karel umožňuje svými příkazy a podmínkami.
Součástí bakalářské práce je dokumentace uživatelská a implementace pro rychlou orientaci v programovém vybavení Robota Karel.

SEZNAM POUŽITÉ LITERATURY

[1] ROBERTS, Eric. Karel the Robot [online]. Stanford University, 2016 [cit. 2017-03-26]. Dostupné z: http://web.stanford.edu/class/cs208e/handouts/04-Karel.pdf

[2] ROBERTS, Eric. Karel the Robot Learns Java [online]. Stanford University, 2005 [cit. 2017-03-26]. Dostupné z:

https://web.stanford.edu/class/cs106a//resources/karel-the-robot-learns-java.pdf

[3] HANCOCK, Terry. GettingStanford's "Karel the Robot" to Run in Debian'sEclipse [online]. 2009 [cit. 2017-03-26]. Dostupné z: http://freesoftwaremagazine.com/articles/getting_stanfords_karel_robot_run_debians_eclipse/

[4] University of Waterloo: CS1. [online]. [cit. 2018-05-24]. Dostupné z: https://cs.uwaterloo.ca/~bwbecker/papers/javaPedagogies.pdf

[5] PACE University: KarelJ. [online]. [cit. 2017-03-26]. Dostupné z: https://csis.pace.edu/~bergin/KarelJava2ed/Karel++JavaEdition.html

[6] ELKNER, Jeffrey. The Guido van Robot [online]. 2004 [cit. 2017-03-26]. Dostupné z:

http://gvr.sourceforge.net/about.php

[7] LAŠTOVIČKA, Petr. Robot KAREL 4.2 [online]. 2011 [cit. 2017-03-26]. Dostupné z:

http://petr.lastovicka.sweb.cz/ostatni.html#karel

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

GUI	Graphical User Interface
TXT	Text file

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: PŘENOSNÉ MÉDIUM CD

Přenosné médium obsahuje následující soubory a adresáře:

- fulltext.pdf
  - Bakalářská práce ve formátu PDF.
- MapyMesta
  - Adresář obsahující mapy měst použité v ukázkových příkladech využití Knihovny robota Karla.
- PrikladyVyuzitiKnihovnyRobotaKarla
  - Adresář obsahující workspace vývojového prostředí Eclipse s ukázkovými příklady Knihovny robota Karla.
- Textury
  - Adresář obsahující textury potřebné pro správné fungování programového rozhraní Knihovny robota Karla.
- ZdrojoveKodyKnihovnyRobotaKarla
  - Adresář obsahující zdrojový a hlavičkový soubor Knihovny robota Karla.