SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-xxxx-xxxx

KOTLIN + JAVA A POUŽITIE V BLOCKING A NON-BLOCKING REŽIME

DIPLOMOVÁ PRÁCA

2020 **Bc. Tomáš Gono**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-xxxx-xxxx

KOTLIN + JAVA A POUŽITIE V BLOCKING A NON-BLOCKING REŽIME

DIPLOMOVÁ PRÁCA

Študijný program:	Aplikovaná informatika
Číslo študijného odboru:	2511
Názov študijného odboru:	9.2.9 Aplikovaná informatika
Školiace pracovisko:	Ústav informatiky a matematiky
Vedúci záverečnej práce:	doc. Ing. Milan Vojvoda, PhD.
Konzultant:	Ing. Martin Rástocký

Bratislava 2020 **Bc. Tomáš Gono**

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Aplikovaná informatika
Autor:	Bc. Tomáš Gono
Diplomová práca:	Kotlin + JAVA a použitie v blocking a non-blocking režime
Vedúci záverečnej práce:	doc. Ing. Milan Vojvoda, PhD.
Konzultant:	Ing. Martin Rástocký
Miesto a rok predloženia práce:	Bratislava 2020

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean et est a dui semper facilisis. Pellentesque placerat elit a nunc. Nullam tortor odio, rutrum quis, egestas ut, posuere sed, felis. Vestibulum placerat feugiat nisl. Suspendisse lacinia, odio non feugiat vestibulum, sem erat blandit metus, ac nonummy magna odio pharetra felis. Vivamus vehicula velit non metus faucibus auctor. Nam sed augue. Donec orci. Cras eget diam et dolor dapibus sollicitudin. In lacinia, tellus vitae laoreet ultrices, lectus ligula dictum dui, eget condimentum velit dui vitae ante. Nulla nonummy augue nec pede. Pellentesque ut nulla. Donec at libero. Pellentesque at nisl ac nisi fermentum viverra. Praesent odio. Phasellus tincidunt diam ut ipsum. Donec eget est. A skúška mäkčeňov a dĺžnov.

Kľúčové slová: Java, Kotlin, REST

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Author:	Bc. Tomáš Gono
Master's thesis:	Kotlin + JAVA usage in blocking and non-blocking mode
Supervisor:	doc. Ing. Milan Vojvoda, PhD.
Consultant:	Ing. Martin Rástocký
Place and year of submission:	Bratislava 2020

On the other hand, we denounce with righteous indignation and dislike men who are so beguiled and demoralized by the charms of pleasure of the moment, so blinded by desire, that they cannot foresee the pain and trouble that are bound to ensue; and equal blame belongs to those who fail in their duty through weakness of will, which is the same as saying through shrinking from toil and pain. These cases are perfectly simple and easy to distinguish. In a free hour, when our power of choice is untrammelled and when nothing prevents our being able to do what we like best, every pleasure is to be welcomed and every pain avoided. But in certain circumstances and owing to the claims of duty or the obligations of business it will frequently occur that pleasures have to be repudiated and annoyances accepted. The wise man therefore always holds in these matters to this principle of selection: he rejects pleasures to secure other greater pleasures, or else he endures pains to avoid worse pains.

Keywords: Java, Kotlin, REST

Obsah

Zoznam obrázkov a tabuliek

Zoznam algoritmov

Zoznam výpisov

Úvod

Tu bude krasny uvod s diakritikou atd.
A mozno aj viac riadkovy uvod.

1 Jazyk Kotlin

Podľa [1] Kotlin je - nový programovací jazyk postavený na Java platforme. Kotlin je stručnejší, programátorsky bezpečnejší, pragmatický jazyk, ktorý je taktiež zameraný na zachovanie interoperability s kódom napísaným v programovacom jazyku Java. Kotlin je možné používať takmer všade kde sa používa Java - na vývoj serverových aplikácií, aplikácii pre operačný systém Android, atď. Jazyk Kotlin umožňuje implementáciu projektov s nižším množstvom kódu, vyššou abstrakciou a s menej nepríjemnosťami ako pri jazyku Java. Prechod na programovanie v jazyku Kotlin z Javy je plynulý proces ktorý nevyžaduje strmú krivku učenia. Ako najvyššia výhoda jazyka Kotlin je udávaná silná interoperabilita s jazykom Java - možnosť zavedenia časti kódu v Kotline do už existujúceho Java projektu. Kotlin poskytuje viaceré možnosti integrácie kódu s prostriedkami z programovacieho jazyka Java:

- volanie Java metód,
- rozširovanie Java tried,
- implementovanie Java rozhraní,
- aplikáciu Java anotácii, atď.

1.1 Základné vlastnosti jazyka Kotlin

Jednou zo základných čŕt jazyka Kotlin zachovaných z programovacieho jazyka Java je podľa [1] statické typovanie premenných, narozdiel od iných - dynamicky typovaných - jazykov ktoré sú taktiež interpretované pomocou Java Virtual Machine (jazyky ako Groovy a JRuby). Avšak Kotlin narozdiel od Javy nepožaduje explicitne špecifikovať typ každej premennej v kóde. V mnohých prípadoch sa typ premennej dokáže identifikovať automaticky z kontextu. Ako najdôležitejšie dôvody na zachovanie statického typovania premenných uvádza [1]:

- *výkon* - volanie metód je rýchlejšie, pretože nie je treba zistiť ktorú z preťažených metód je potrebné zavolať pri behu programu,
- *spoľahlivosť* - kompilátor dokáže verifikovať správnosť typov takže je menej prípadov zlyhania programov,
- *udržovateľnosť kódu* - práca s neznámym kódom je jednoduchšia, pretože programátov vždy vie s akým objektom v danom mieste kódu pracuje,

- *podpora nástrojov* - pri statickom typovaní je možný jasný re-faktoring, automatické doplnenie kódu a ďalšie IDE nástroje.

Syntax jazyka podľa [2] Kotlin vyžaduje deklarácie premenných pred ich prvým použitím. Premenné sa deklarujú pomocou kľúčových slov *var* a *val*. Obsah premenných typu var je možné modifikovať počas priebehu aplikácie, pričom premenné typu val môžu mať obsah priradený len raz. Ak sa pri deklarácii premennej priradí aj hodnota nie je potreba špecifikovať typ premennej ktorý sa automaticky odvodí od priradenej hodnoty.

Nasledujúca vlastnosť ktorú poskytuje jazyk Kotlin je podľa [3] tzv. *"Null bezpečnosť"* - null safety. Pokiaľ nie je premenná špecificky označená operátorom ktorý umožnuje premennej obsahovať hodnotu null, kompilátor vyhlási chybu ak sa premennej priradí hodnota null alebo hodnota premennej ktorá môže obsahovať null. Taktiež nie je možné priamo pristupovať k metódam a premenným objektov cez premennú ktorá môže dosiahnuť hodnotu null. V tomto prípade je potrebné využiť jednu z nasledujúcich možností:

1. explicitne skontrolovať, či premenná obsahuje hodnotu null a spracovať oba prípady zvlášť,

2. využiť špeciálny operátor na bezpečný prístup k obsahu premennej ktorý vráti tento obsah ak premenná nie je null a null inak,

3. použiť tzv. ëlvisöperátor ktorý označuje že ak operand naľavo od tohto operátora nie je null, má sa vrátiť jeho hodnota, inak sa má vrátiť hodnota pravého operandu,

4. použitie non-null assertion operátora ktorým programátor explicitne požaduje vyvolanie NullPointerException ak sa v algoritme vyskytne snaha o prístup k obsahu premennej obsahujúcej hodnotu null.

Kotlin taktiež poskytuje jednoduchšiu prácu s reťazcami pomocou tzv. "string templates reťazcovými šablónami spomínanými v [1] a v [4]. Podobne ako v mnohých skriptovacích jazykoch je v jazyku Kotlin možné priamo konkatenovať reťazec s hodnotou premennej. Napr. príkaz skonkatenuje reťazec "*Hello,* "s hodnotou premennej *name*, čo

```
println("Hello, $name!")
```

zjednodušuje a zmenšuje obsah kódu potrebného na časté výpisy. V Jave by ekvivalentný príkaz vyzeral:

```
System.out.println( "Hello, "+ name + "!")
```

Stratégiou jazyka Kotlin je taktiež podľa [1] eliminácia zbytočného kódu pri vytváraní tried. Na obrázku 1 je zobrazená deklarácia jednoduchej triedy Person, ktorá obsahuje len dáta a metódy na získanie a nastavenie týchto dát - tzv. getter a setter metódy.

```
/* Java */
public class Person {
    private final String name;

    public Person(String name)
        { this.name = name;
    }

    public String getName()
        { return name;
    }
}
```

Obr. 1: Deklarácia triedy v jazyku Java, prevzaté z [1].

```
class Person(val name: String)
```

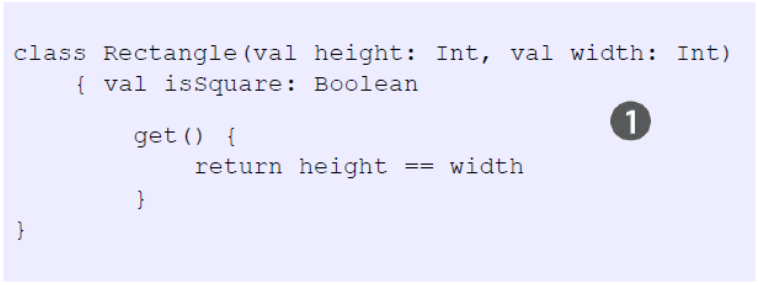
Obr. 2: Deklarácia triedy v jazyku Kotlin, prevzaté z [1].

Narozdiel od Javy je možné takúto triedu v Kotline deklarovať oveľa jednoduchšie, ako je vidieť na obrázku 2. Kotlin automaticky vytvára základné getter a setter metódy a taktiež považuje public ako predvolený modifikátor prístupu. Ako už bolo spomenuté ak je potrebné zakázať zapisovanie do hodnoty tak sa označí kľúčovým slovom val pričom v Jave by sa nevytvorila setter metóda. Samozrejme v jazyku Kotlin je možné deklarovať vlastné getter a setter metódy - ako je možné vidieť na obrázku - avšak na rozdiel od Javy je jednoduchšie rozlíšiť či ide o metódu triedy alebo o getter a setter na premennú.

Kotlin taktiež eliminuje potrebu replikovať balíkovú hierarchiu priečinkovou hierarchiou zdrojových kódov. Avšak podľa [1] je stále dobrá programátorská praktika zachovávať Java priečinkovú hierarchiu, najmä ak sa v projekte spájajú časti implementované v Jave a časti v Kotline.

1.2 Nové metódy jazyka Kotlin

Jazyk Kotlin poskytuje nové programátorské možnosti oproti jazyku Java zamerané hlavne na paralelizáciu algortimov. Najmä pri aplikáciach pre Android, pri jedno vlákno-



Obr. 3: Deklarácia triedy v jazyku Kotlin, prevzaté z [1].

vom programe by GUI počas vykonávania dlhých, výpočtovo náročných operácii nereagovalo. Tento problém rieši programovací jazyk Java vytváraním vlákien na spracovanie takýchto náročnejších operácii. Avšak manažment viacerých vlákien má tendenciu komplikovať zdrojový kód. Kotlin vie taktiež pracovať s vláknami podobne ako jazyk Java, avšak prináša nové prístupy. Hlavnou novou možnosťou tohto jazyka sú tzv. koprogramy "*coroutines*anglicky. [5] uvádza coroutines ako nové riešenie týchto operácií. Coroutines narozdiel od klasických vlákien umožňujú vykonávať výpočtovo náročné operácie pozastavením vykonávania kódu hlavného vlákna - pričom však neblokujú toto vlákno - a pokračovaním vykonávania kódu tohto vlákna neskôr. Takto je teda možné vytvoriť asynchrónne neblokujúce kódy ktoré však na prvý pohľad vyzerajú ako synchrónne bloky.

Ďalšou novou vlastnosťou jazyka Kotlin je v [1] uvedené pridanie podpory pre funkcionálne programovania, nie len objektovo-orientované. Niektoré z hlavných konceptou tohto prístupu sú:

- *tzv. "prvotriedne funkcie"* (first-class functions) - s funkciami (s časťami správania) sa dá pracovať ako s hodnotami. Funkcie sa dajú ukladať do premenných, použiť ich ako parametre funkcií alebo ich použiť ako návratové hodnoty iných funkcií.

- *Nemeniteľnosť* (immutability) - možnosť pracovať s nemeniteľnými objektami čo garantuje že ich stav sa nemôže zmeniť po ich vytvorení.

- *Žiadne vedľajšie efekty* - možnosť využívať tzv. *"rýdze"* funkcie (pure functions), ktoré pri rovnakom vstupe vždy vrátia rovnaký výstup a taktiež nemodifikujú stav žiadneho objektu ani neinteragujú s vonkajším svetom,

- *funkcie vyššieho rádu* - podľa [5] sú takéto funkcie schopné prijímať iné funkcie ako argumenty a taktiež vrátiť funkcie ako návratové hodnoty.

Ďalším novým prístupom jazyka Kotlin je podľa [5] rozširovanie tried pomocou tzv. ëxtension"funkcií. V [6] je uvedené, že ak pri deklarácii funkcie programátor uvedie ako prefix triedu oddelenú od deklarácie funckie operátorom ., je možné potom túto funkciu volať rovnako ako metódu triedy v Java, čiže operátorom . .

2 RESTful API

[7] označuje RESTful API ako rozhranie aplikačného programu - po anglicky Application Program Interface - ktoré je založené na princípoch tzv. "representational state transfer"(skratka REST) technológie ktorá využíva štandardné HTTP stavové kódy a metódy na vzdialenú komunikáciu a prenos dát. REST technológie sú tu uvedené ako preferované oproti tzv. Simple Object Access Protocol (skratka SOAP) prístupu, pretože prenášajú menší obsah dát, čím sa zrýchľuje ich dostupnosť po sieťach, najmä po internete.

RESTful API rozčleňuje transakciu - [8] transakcie označuje ako sekvencie výmen informácii a pridružených operácii - na menšie časti, pričom každú časť transakcie spracuje modul na to určený. RESTFUL API využíva HTTP metodológie:

- metódu *GET* na získanie dát,
- metódu *PUT* na modifikáciu stavu alebo aktualizovanie zdroja dát (môže byť napr. súbor, dátový objekt, dátový blok, atď),
- metódu *POST* na vytvorenie nového zdroja dát,
- metódu *DELETE* na vymazanie zdroja.

Pri tomto prístupe sú komponenty so vzdialeným - sieťovým - prístupom reprezentované ako zdroj dát ku ktorému má aplikácia prístup, implmentácia týchto komponent je nejasná. RESTful API požaduje aby prístupky k zdroju dát boli tzv. bez-stavové, t.j. podľa [9] musí každý dopyt od klienta obsahovať všetky informácie aby server dokázal tento dopyt pochopiť a vykonať správne akcie. Taktiež server musí poskytnúť v odpovediach všetky informácie na to, aby si klient mal informáciu o aktuálnom stave dát bez znalosti predchádzajúcich operácii alebo stavov.

Aplikácie ktoré dodržujú RESTful API musia taktiež podľa [7] spĺňať tieto požiadavky:

1. zdroje dát majú byť identifikovateľné pomocou URI a iba metódy HTTP GET, PUT, POST a DELETE môžu s týmito zdrojmi manipulovať,
2. aplikácie majú byť jasne rozdelené na serverovú časť a klientskú časť - server zabezpečuje prístup k dátam reprezentovaných zdrojmi, riadenie pracovnej záťaže modulov a bezpečnosť,
3. bez-stavovosť,

4. caching zdrojov dát pokiaľ to nie explicitne zakázané,

5. možnosť vytvárať architektúru RESTful služieb zloženú z viacerých serverových vrstiev,

6. možnosť namiesto statickej reprezentácie dát taktiež získať spustiteľný kód zo servera - nepovinné.

Záver

Conclusion is going to be where?
Here.

Zoznam použitej literatúry

1. JEMEROV, Dmitry a ISAKOVA, Svetlana. *Kotlin in Action: Version 11* [online]. MEAP. Manning Publications, 2017 [cit. 2020-01-27]. Dostupné z: http://sd.blackball.lv/library/Kotlin_in_Action_(2017).pdf.
2. ELDHUSET, Aasmund. *Declaring variables* [online] [cit. 2020-01-27]. Dostupné z: https://kotlinlang.org/docs/tutorials/kotlin-for-py/declaring-variables.html.
3. *Null Safety* [online] [cit. 2020-01-27]. Dostupné z: https://kotlinlang.org/docs/reference/null-safety.html.
4. PARASCHIV, Eugen. *Kotlin String Templates* [online] [cit. 2020-01-28]. Dostupné z: https://www.baeldung.com/kotlin-string-template.
5. SINGH, Vijay. *Kotlin vs Java: Most Important Differences That You Must Know* [online]. 2020 [cit. 2020-01-28]. Dostupné z: https://hackr.io/blog/kotlin-vs-java.
6. *Extensions* [online] [cit. 2020-01-27]. Dostupné z: https://kotlinlang.org/docs/reference/extensions.html.
7. ROUSE, Margaret. *RESTful API (REST API)* [online]. 2019 [cit. 2020-01-28]. Dostupné z: https://searchapparchitecture.techtarget.com/definition/RESTful-API.
8. ROUSE, Margaret [online] [cit. 2020-01-28]. Dostupné z: https://searchcio.techtarget.com/definition/transaction.
9. *Statelessness* [online] [cit. 2020-01-28]. Dostupné z: https://restfulapi.net/statelessness/.

Prílohy

A Štruktúra elektronického nosiča

/CHANGELOG.md

· file describing changes made to FEIstyle

/example.tex

· main example *.tex* file for diploma thesis

/example_paper.tex

· example *.tex* file for seminar paper

/Makefile

· simply Makefile – build system

/fei.sublime-project

· is project file with build in Build System for Sublime Text 3

/img

· folder with images

/includes

· files with content

/bibliography.bib

· bibliography file

/attachmentA.tex

· this very file

B Algoritmus

Algorithm B.1 Vypočítaj $y = x^n$

Require: $n \geq 0 \vee x \neq 0$

Ensure: $y = x^n$

```
y ⇐ 1
if n < 0 then
    X ⇐ 1/x
    N ⇐ −n
else
    X ⇐ x
    N ⇐ n
end if
while N ≠ 0 do
    if N is even then
        X ⇐ X × X
        N ⇐ N/2
    else {N is odd}
        y ⇐ y × X
        N ⇐ N − 1
    end if
end while
```

C Výpis sublime

```
../../ fei .sublime-project
```

Listing C.1: Ukážka sublime-project