MASARYKOVA
UNIVERZITA

FAKULTA INFORMATIKY

Návrh semestrálneho projektu na predmet Programovanie v jazyku Java

Bakalárska práca

ALŽBETA STROMPOVÁ

Brno, jar 2022

MASARYKOVA
UNIVERZITA

FAKULTA INFORMATIKY

Návrh semestrálneho projektu na predmet Programovanie v jazyku Java

Bakalárska práca

ALŽBETA STROMPOVÁ

Vedúci práce: RNDr. Radek Ošlejšek, Ph.D.

Katedra počítačových systémů a komunikací

Brno, jar 2022

FACULTAS ARTIS INFORMATICÆ • UNIVERSITAS MASARYKIANA •

Vyhlásenie

Vyhlasujem, že táto bakalárska práca je mojím pôvodným autorským dielom, ktoré som vypracovala samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používala alebo z nich čerpala, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Alžbeta Strompová

Vedúci práce: RNDr. Radek Ošlejšek, Ph.D.

Poďakovanie

Rada by som sa poďakovala RNDr. Radkovi Ošlejškovi, Ph.D. za vedenie mojej bakalárskej práce, pripomienky a cenné rady. Zároveň by som rada poďakovala svojej rodine a priateľom za podporu a čas počas štúdia a písania práce.

Zhrnutie

Bakalárska práca sa zaoberá návrhom nového semestrálneho projektu na predmet PB162 - Programování v jazyce Java. Semestrálny projekt sa delí na iterácie, ktoré odrážajú preberanú látku počas semestra. Súčasťou práce je návod na zverejnenie a úpravu iterácií. Celý projekt, s pomocnými skriptami na jeho správu, sa nachádza na školskom serveri používajúci Git. Aplikačnou doménou projektu sú hry na šachovnici. Projekt sa používa na precvičenie a overenie znalostí študentov, ktoré získavajú v priebehu semestra.

Kľúčové slová

Java, PB162, Git, OOP, objektovo orientované programovanie, návrhové vzory

Obsah

Úvod 1

1 Metodológia riešenia 2
- 1.1 Požiadavky . . . 3
- 1.2 Technológie . . . 3
- 1.3 Návrh aplikačnej domény . . . 4
- 1.4 Návrhové vzory . . . 6
- 1.5 Platforma . . . 6
- 1.6 Vytvorenie a rozdelenie projektu . . . 7
- 1.7 Napísanie zadaní a testov . . . 7
- 1.8 Úprava skriptov . . . 7

2 Implementácia iterácii 9
- 2.1 Prvá iterácia . . . 9
- 2.2 Druhá iterácia . . . 10
- 2.3 Tretia iterácia . . . 11
- 2.4 Štvrtá iterácia . . . 11
  - 2.4.1 Návrhový vzor Prototype . . . 12
- 2.5 Piata iterácia . . . 12
- 2.6 Šiesta iterácia . . . 14
  - 2.6.1 Návrhový vzor Strategy . . . 15
  - 2.6.2 Návrhový vzor Memento . . . 15
- 2.7 Siedma iterácia . . . 16
  - 2.7.1 Návrhový vzor Factory method . . . 17
- 2.8 Ôsma iterácia . . . 19
- 2.9 Deviata iterácia . . . 19
  - 2.9.1 Návrhový vzor Builder . . . 19
- 2.10 Desiata iterácia . . . 21

3 Správa repozitára 23
- 3.1 Založenie projektu . . . 23
- 3.2 Zverejnenie iterácii . . . 25
- 3.3 Úprava projektu . . . 26

Záver 29

A Prílohy **30**

Bibliografia **31**

Zoznam tabuliek

1.1 Harmonogram iterácií počas roka 2
1.2 Rozdelenie konceptov do iterácii 5

Zoznam obrázkov

2.1 Hierarchia dedičnosti hier v piatej iterácii 13
2.2 Textová reprezentácia šachovnice v piatej iterácii 14
2.3 Hierarchia návrhového vzoru stratégia po ôsmej iterácii . 15
2.4 Hierarchia návrhového vzoru memento v šiestej iterácii . 16
2.5 Hierarchia dedičnosti návrhového vzoru factory method po siedmej iterácii 17
2.6 Textová reprezentácia šachovnice po siedmej iterácii . . . 18
2.7 Hierarchia dedičnosti po aplikovaní návrhového vzoru Builder 20
2.8 Záverečná hierarchia dedičnosti v semestránom projekte . 21
2.9 Formát textovej reprezentácie iniciálnej pozície v šachu . 22

3.1 Štruktúra repozitárov po založení projektu 25
3.2 Štruktúra repozitárov na konci semestra 26
3.3 Grafické znázornenie skriptu *propagate-changes-locally.sh* . 28

Úvod

Na Fakulte informatiky Masarykovej univerzity existuje viacero kurzov Javy. Úvodným predmetom je *PB162 Programování v jazyce Java*, kde sa učia základné znalosti moderného objektového programovania, základy objektového návrhu a implementácie v jazyku Java [1]. Predmet je v odporučenom priebehu štúdia zaradený do druhého ročníka, ale zapisuje si ho aj množstvo študentov prvého ročníka.

Na precvičenie a overenie znalosti, ktoré študenti získajú v priebehu kurzu, sa okrem skúšok používa semestrálny projekt. Pozostáva z desať častí(iterácií). Projekt je pre väčšinu študentov prvý väčší projekt, kde sa naučia základy objektovo orientovaného programovania alebo si precvičia jeho znalosti.

Cieľom mojej bakalárskej práce je pripraviť nový semestrálny projekt, ktorý overí znalosti študentov získané v priebehu kurzu. Projekt kopíruje koncepty, ktoré sú preberané na prednáškach, a obsahuje textové zadanie napísané v anglickom jazyku, predpripravené objekty a jednotkové testy. Celý projekt je uložený na Gite kvôli postupnému zverejňovaniu a aktualizovaniu projektu do budúcnosti. Zároveň nový semestrálny projekt využíva najnovšie technológie, ktoré sa využívajú aj v praxi.

Najskôr som si preštudovala sylabus a existujúci semestrálny projekt. Následne som navrhla nové zadanie tak, aby pokrývalo vyučované témy. Po implementovaní finálnej verzie som kód rozdelila na desať častí, ktoré odpovedajú preberanej látke v predmete. V 1. kapitole sa nachádza popis postupu implementácie semestrálneho projektu. Popis rozdelenia projektu je popísaný v 2. kapitole. Kapitola 3 zhŕňa technické informácie o správe projektu na Gite a jeho postupnom zverejňovaní študentom počas semestra.

1 Metodológia riešenia

Semestrálny projekt sa skladá z desatich častí(iterácií). Implementácia iterácii slúži ako príprava na skúšky a precvičenie vedomosti nadobudnutých v priebehu kurzu. Každý týždeň výuky okrem prvého týždňa a týždňov, kedy prebieha vnútro-semestrálna skúška, sa zverejňuje nová iterácia. Harmonogram iterácii počas semestra je ukázaný v tabuľke 1.1. Každá iterácia sa skladá z predpripraveného kódu, textového zadania v anglickom jazyku a testov, ktoré kontrolujú, či študenti správne naprogramovali iteráciu podľa zadania. Testy sú jednotkové, takže sú zamerané na jednotlivé časti programu bez vzťahu a závislosti na okolí [2]. Vďaka testom si študenti vyskúšajú aj *Test Driven Development*[1].

Tabuľka 1.1: Harmonogram iterácií počas roka

Týždeň v semestri	Číslo iterácie
1. týždeň	žiadna iterácia
2. týždeň	Prvá iterácia
3. týždeň	Druhá iterácia
4. týždeň	Tretia iterácia
5. týždeň	Štvrtá iterácia
6. týždeň	Piata iterácia
7. týždeň	1. vnútro-semestrálna skúška
8. týždeň	Šiesta iterácia
9. týždeň	Siedma iterácia
10. týždeň	Ôsma iterácia
11. týždeň	2. vnútro-semestrálna skúška
12. týždeň	Deviata iterácia
13. týždeň	Desiata iterácia

1. *Test Driven Development* je proces vývoja softvéru, ktorý sa opiera o softvérové požiadavky, ktoré sa prevedú na testovacie prípady ešte pred úplným vývojom softvéru [3]

1.1 Požiadavky

Semestrálny projekt precvičuje všetky potrebné koncepty preberané na *PB162 Programování v jazyce Java*. Iterácie kopírujú preberanú látku v priebehu semestra a ich vypracovanie odpovedá približne dvom hodinám programovania. Zadania sú napísane v anglickom jazyku a každá iterácia je pokrytá jednotkovými testami. Základné požiadavky kladené na testy sú nezávislosť, rýchlosť, izolovanosť a zmysluplnosť. Nezávislosť znamená, že testy nie sú závislé na sebe navzájom, na vývojovom prostredí, ani na operačnom systéme. Zároveň spustenie testov nemá byť časovo náročne, kód testov má byť oddelený od implementácie a testovať všetky možné vstupy [2].

Aplikačnú doménu projektu je treba zvoliť tak, aby sa na nej dali ukázať všetky znalosti, ktoré sa na predmete študenti naučia. Popritom študentom by aplikačná doména mala byť blízka a zaujímavá.

1.2 Technológie

Aktuálny semestrálny projekt funguje na JDK[2] verzie 11, ktorej končí prémiová podpora v septembri 2023 [4]. Premiérová podpora poskytuje prístup k aktualizáciám a vylepšeniam produktov, ako aj technickú pomoc a zdroje podpory, ktoré pomáhajú optimalizovať výkon a efektívne implementovať nové funkcie. JDK obsahuje nástroje na vývoj a testovanie programov napísaných v programovacom jazyku Java a bežiacich na platforme Java [5]. V roku 2021 bola vydaná verzia JDK 17, ktorá poskytuje nové možnosti. Od vydania JDK 11 je to prvá verzia s dlhodobou podporou a je o 8,66% rýchlejšia ako momentánne používaná verzia 11 [6].

V aktuálnom projekte nie sú ukázane niektoré nové koncepty už pôvodných dlhodobo podporovaných verzií. Napríklad predvolené a statické metódy rozhrania. Zároveň verzia 17 ponúka okrem iného záznamy[3], nápomocnejšie *NullPointException*, vyhľadávanie vzorov pre výrazy *switch* a *instanceof*.

Od napísania aktuálneho projektu sa obsahovo zmenili prednášky, pribudli návrhové vzory a koncepty z nových verzií JDK. V novom

2. Java Development Kit
3. po anglicky *record*

semestrálnom projekte je dôraz na novinky JDK 17, návrhové vzory, prúdy, ale naďalej precvičuje aj základné znalosti, ako sú napríklad zapuzdrenie, dedičnosť, polymorfizmus, výnimky, vstup a výstup.

1.3 Návrh aplikačnej domény

Po preštudovaní študijných materiálov predmetu som vypísala všetky koncepty, ktoré je potrebne precvičovať v semestrálnom projekte. Následne som ich rozdelila do iterácii, aby odpovedali preberanej látke počas semestra. Rozdelenie konceptov do iterácii, po skonzultátovaní s vedúcim práce, je ukázaný v tabuľke 1.2.

Aktuálny semestrálny projekt pracuje s geometrickými tvarmi. Výsledkom je približne 20 tried a 10 rozhraní, ktoré spolu vytvárajú komplexný projekt. Študenti si tam môžu vyskúšať prácu s rozľahlým projektom, v ktorom sú triedy na sebe závislé a komunikujú spolu. Dôležitým aspektom je, že študenti po úspešnom naprogramovaní vidia spustiteľný výsledok. V aktuálnom projekte to zabezpečuje predpripravená trieda, ktorá vykresľuje obrázky.

Na základe požiadaviek som zvolila aplikačnú doménu šach. Existuje množstvo možnosti ako implementovať šach. Cieľom bolo vybrať, takú dekompozíciu, aby na nej bolo možné precvičiť všetky potrebné koncepty a zároveň, aby bola logická. Pospájaním jednotlivých konceptov s funkcionalitami šachu, a konzultáciou s vedúcim práce a vybranými učiteľmi predmetu, sme sa rozhodli zovšeobecnieť aplikačnú doménu. Namiesto šachu sa stali aplikačnou doménou hry na šachovnici, aby bolo možne využiť dedičnosti pri rôznych typoch hier. Konkrétne sa jedná o zjednodušený šach a dámu. Hry na šachovnici poskytujú veľa príležitosti na precvičenie základných princípov objektovo orientovaného programovania, s ktorými sa študenti počas kurzu zoznamujú. Zároveň nová aplikačná doména umožňuje precvičenie algoritmizácie a rôznych návrhových vzorov. Na záver si môžu študenti medzi sebou zahrať hru, ktorú sami naprogramovali.

Tabuľka 1.2: Rozdelenie konceptov do iterácii

Číslo iterácie	Precvičované koncepty
Prvá iterácia	vytvorenie základných tried práca s primitívnymi typmi zapuzdrenie
Druhá iterácia	algoritmizácia nad poľami nemenné typy konštanty pretypovanie
Tretia iterácia	vymenované typy metóda *toString* komunikácia medzi triedami
Štvrtá iterácia	vstup od používateľa rozhrania preťažovanie metód
Piata iterácia	dedičnosť abstraktná trieda *StringBuilder* prepisovanie metód
Šiesta iterácia	identita objektov základné kolekcie algoritmizácia nad kolekciami
Siedma iterácia	kolekcie a prúdy základne výnimky dedičnosť
Ôsma iterácia	usporiadané kolekcie anonymné triedy
Deviata iterácia	vlastne výnimky vnútorne triedy generické typy
Desiata iterácia	vstup a výstup tvorba testov

1.4 Návrhové vzory

V semestrálnom projekte sú ukázane študentom niektoré zo základných návrhových vzorov. Návrhový vzor je typickým riešením bežne sa vyskytujúcich problémov pri návrhu softvéru. Nejde o hotový dizajn, ktorý je možné pretransformovať priamo do kódu. Nestačí nájsť vzor a skopírovať ho do svojho programu, ako je to možné s bežne dostupnými funkciami alebo knižnicami. Návrhový vzor je všeobecný popis alebo šablóna, ktorú je možné prispôsobiť, aby vyriešila opakujúci sa problém s návrhom softvéru v kóde [7]. Študenti na základe pokynov v zadaní implementujú návrhový vzor bez nutnosti predchádzajúcich znalosti o ňom. Cieľom predmetu nie je naučiť študentov konkrétne návrhové vzory ale ukázať, že niektoré použité dekompozície sú typické a odpovedajú známym návrhovým vzorom používaným v praxi. Názvy návrhových vzorov neprekladám z angličtiny, lebo nemajú zaužívaný slovenský preklad.

1.5 Platforma

Tvorba a rozdelenie semestrálneho projektu prebieha na Gite, softvéri na sledovanie zmien v akejkoľvek sade súborov. Git je *open-source* distribuovaný systém, ktorý je navrhnutý tak, aby rýchlo a efektívne zvládal všetko od malých až po veľké projekty. Semestrálny projekt je uložený na školskom serveri `www.gitlab.fi.muni.cz` bežiacom na platforme Gitlab, ktorý je plne integrovaná platforma pre vývoj softvéru založená na systéme Git. GitLab okrem funkcií, ktoré má Git, má aj množstvo výkonných funkcií na zlepšenie pracovného postupu [8]. Zvyčajne sa používa na koordináciu práce medzi programátormi, ktorí spolupracujú na vývoji zdrojového kódu. Zaznamenáva zmeny v súbore alebo skupine súborov v priebehu času, aby bolo možné neskôr si vyvolať konkrétne verzie [9].

Pri práci s Gitlabom je uložená lokálna verzia na počítači a je možné mať viacero vzdialených repozitárov, kde sa ukladá lokálna verzia. Každý repozitár môže obsahovať viacero vetiev. Vetva je verzia pracovného stromu projektu. Pre každú množinu súvisiacich zmien, ktoré sú vykonané, je možné vytvoriť vetvu. Vďaka tomu je každá skupina

zmien oddelená od seba, čo umožňuje vykonávať zmeny paralelne bez toho, aby sa navzájom ovplyvňovali [8].

1.6 Vytvorenie a rozdelenie projektu

Implementovala som semestrálny projekt, ktorý reprezentoval stav po poslednej iterácii. Tento stav študenti dostanú po implementácii všetkých desiatich iterácii. Po konzultácii a odstránení pokročilejších konceptov z projektu som začala postupne deliť kód do jednotlivých iterácii. Vytvorila som si novú vetvu, z ktorej som odstraňovala kód. Tvorba iterácii prebiehala od poslednej, takže vytvorila som si vetvu deviatej iterácie a odstránila z nej všetok kód, ktorý bol naplánovaný na desiatu iteráciu. Postup som opakovala, až kým som sa nedostala k prvej iterácii. Následne som si niekoľkokrát prešla všetky vetvy a robila menšie úpravy po konzultácii s vedúcim práce. Rozdelený projekt som nahrala na server.

1.7 Napísanie zadaní a testov

Písanie zadaní a testov na rozdiel od rozdeľovania projektu prebiehalo od iterácie s najmenším poradovým číslom. Skopírovala som si kód predchádzajúcej iterácie a nahradila ním aktuálnu iteráciu. Vďaka verzovaciemu systému som videla všetky zmeny. Postupne som písala zadanie a odstraňovala som zmeny až kým iterácia nebola v pôvodnom stave. Následne som doplnila zadania a usporiadala poradie jednotlivých úloh, aby logicky nadväzovalo.

Na základe zadaní som napísala jednotkové testy. Inšpirovala som sa testami aktuálneho semestrálneho projektu a domácich úloh predmetu. Vďaka možnosti opravovať iterácie študentom ako pomocník predmetu som sa stretla s riešeniami, ktoré prešli testami, ale boli chybné. V novom semestrálnom projekte som sa snažila tomuto vyhnúť vysokým pokrytím testami.

1.8 Úprava skriptov

Poslednou častou implementácie je upravenie existujúcich skriptov, ktoré sa momentálne používajú na správu projektu. V novom semes-

trálnom projekte je základnou vetvou *main*, ale skripty predpokladajú názov *master*. Zároveň som doplnila a upravila dokumentáciu skriptov.

Názvy repozitárov, kde sa nachádza projekt obsahujú aktuálny rok. Skripty nie je potrebné meniť naprieč rokmi, lebo aktuálny rok je dopĺňaní automaticky pomocou príkazu *'date +%Y'*.

2 Implementácia iterácii

Semestrálny projekt pozostáva z desiatich iterácii, ktoré slúžia na precvičenie si látky preberanej na prednáškach predchádzajúci týždeň. Každá iterácia obsahuje testy, ktoré kontrolujú správnosť implementácie aj dodržiavanie základných konvencii. Napríklad správne použitie veľkosti písmen v názvoch atribútov. Pre uznanie správnej implementácie riešenia je nutné, aby prešli všetky testy.

Cieľom tejto kapitoly je popísanie zadaní iterácii. Pre popis je zvolená forma popisu jednotlivých iterácii tak, ako nadväzujú na seba. U každej iterácie sa text zameriava len na podstatne informácie z kódu, zadania a relevantné objektové koncepty, ktoré sú súčasťou novej látky v daný týždeň. Presne znenie zadaní iterácií je súčasťou prílohy.

2.1 Prvá iterácia

Prvá iterácia slúži na zoznámenie sa s vývojovým prostredím, objektovo orientovaným programovaním a samotným programovacím jazykom Java. Od začiatku sú študenti nútení dodržiavať zapuzdrenie. Atribúty majú privátnu viditeľnosť a je potrebné si vytvoriť metódy na získavanie a nastavovanie atribútov. Zapuzdrenie je dôležitý mechanizmus, ktorý pomáha podporovať dobré zásady softvérového inžinierstva a chráni premennú pred všetkými druhmi pokusov o prístup z iných častí programu [10].

Študenti majú za úlohu vytvoriť nový balík(adresár), aby si vybudovali dobré návyky na organizovanie kódu. Ďalej vytvárajú jednoduché triedy s atribútmi, ktoré majú primitívne aj objektové typy. Súčasťou predmetu je taktiež naučiť študentov dodržiavať konvencie a správne písať dokumentáciu. Testy skontrolujú, či sú splnené požiadavky na dokumentáciu a nie sú porušované konvencie zavedené na predmete.

Nasledujúcou úlohou je práca s primitívnymi typmi, konkrétnejšie matematické operácie s číslami. Študenti si v iteráciách vyskúšajú časti kódu, ktorý naimplementovali pomocou vypisovania do konzoly. V prvej iterácii majú za úlohu vytvorenie nových inštancii, nastavenie atribútov pomocou konštruktorov aj pomocou pomocných metód. Je ukázaný rozdiel medzi volaním lokálnych metód a volaním metód

inej triedy. Výsledok vypíšu do konzoly a testy skontrolujú správnosť riešenia.

Po dokončení implementácie prvej iterácie semestrálny projekt obsahuje triedy reprezentujúce hráča, figúrku a súradnice na šachovnici. Hráč ani figúrka zatiaľ ešte nemajú všetky potrebné atribúty ako napríklad farbu.

2.2 Druhá iterácia

Druhá iterácia precvičuje nemenné objekty, konštanty, jednoduchú algoritmizáciu nad poľami a používanie podmienok. Študenti majú za úlohu vytvoriť *Utility* triedu, tiež známu ako pomocná trieda, ktorá obsahuje iba statické metódy a nie je možné vytvoriť jej inštanciu. Pomocná trieda v semestrálnom projekte slúži na prevod šachovej notácie na súradnice na šachovnici a naopak. Implementácia si vyžaduje vykonávanie matematických operácii a pretypovanie medzi číselným typom a typom reprezentujúci znak.

Ďalej v iterácii je potrebné zmeniť triedu na záznam, ktorý je novým druhom deklarácie typu od JDK 14. Podobne ako vymenovaný typ, aj záznam je obmedzenou formou triedy. Záznam je nemenná dátová štruktúra, ktorá na deklaráciu potrebuje len názvy a typy atribútov [11]. Java kompilátor vygeneruje privátne nemenné atribúty s metódami na ich získanie, konštruktor, ktorého vstupné parametre sú všetky atribúty záznamu, a metódy *equals*, *hashCode* a *toString*. Nemenné objektové typy umožňujú výrazne efektívnejšie rozdelenie práce medzi viac procesorov, a tým aj zvýšenie efektivity a zlacnenie prevádzky celého systému [12].

Šachovnica je reprezentovaná ako trieda s dvojdimenzionálnym polom ako atribútom. Úloha študentov je implementovať metódy, ktoré pristupujú k jednotlivým prvkom poľa a overujú podmienky ako napríklad, či sa nachádza na danom mieste figúrka.

Po dokončení implementácie druhej iterácie v semestrálnom projekte pribudla trieda reprezentujúca šachovnicu a pomocná trieda na prevod šachovej notácie.

2.3 Tretia iterácia

Tretia iterácia slúži na precvičenie si prepisovania metódy *toString* a s tým spojenú prácu s reťazcami, vymenovaných typov a komunikáciu medzi triedami. Úlohou študentov je prepísať metódu *toString*, ktorá poskytuje pre daný objekt jeho textovú reprezentáciu. Ďalej je potrebné vytvoriť tri vymenované typy, ktorým sa pridajú vlastné metódy, aby videli, že k vymenovaným typom je možné sa správame ako ku triedam, ktoré exportujú jednu inštanciu pre každú vymenovanú konštantnú hodnotu prostredníctvom verejného statického nemenného poľa. Vymenované typy sú v skutočnosti konečné, pretože nemajú prístupné konštruktory [13].

Študentov sa snaží zadanie naviesť na správnu implementáciu, ale necháva im aj voľnosť. Pri vytváraní triedy reprezentujúcej hru majú sami zvoliť, ktoré atribúty by mali byť nemenné.

V implementácii sa nachádzajú dve metódy iných tried s rovnakým názvom, aby študenti videli rozdiel vo volaní metód inštancie vlastnej triedy a metód inštancii iných tried.

Po dokončení implementácie tretej iterácie v semestrálnom projekte pribudla trieda reprezentujúca hru a vymenované typy reprezentujúce farbu, stav hry a typ figúrky. Zároveň boli aktualizované triedy hráč a figúrka o nové atribúty.

2.4 Štvrtá iterácia

Štvrtá iterácia slúži na precvičenie si preťažovania metód, prácu so vstupom od používateľa, implementáciu a tvorbu vlastných rozhraní. Pomocou *Scanner*(*System.in*) získajú vstup od používateľa a spracujú ho pretypovaním a metódami z *Utility* triedy, ktorú si vytvorili v druhej iterácii.

Súčasťou iterácie je predpripravené funkcionálne rozhranie, aby pracovali s rozhraním predtým než si vytvoria vlastné. Funkcionálne rozhranie obsahuje práve jednu abstraktnú metódu [13]. Úlohou študentov je vytvoriť rozhranie, ktoré demonštruje vlastnosť hrateľnosť.

2.4.1 Návrhový vzor Prototype

Predpripravené rozhranie obsahuje len jednu metódu na vyrábanie kópie objektu. Úlohou študentov je implementovať rozhranie pomocou, ktorého využijú návrhový vzor prototype. V deviatej iterácii následne upravia rozhranie, aby používalo generické typy.

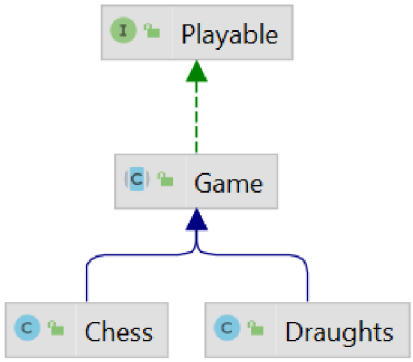
Prototype je tvorivý návrhový vzor, ktorý umožňuje kopírovať existujúce objekty bez toho, aby bol kód závislý od ich tried. Vzor prototype deleguje proces klonovania na samotné objekty, ktoré sa klonujú. Na implementáciu vzoru je potrebné vytvorenie spoločného funkcionálneho rozhrania pre všetky objekty, ktoré podporujú klonovanie. Toto rozhranie umožňuje klonovať objekt bez potreby zverejňovania privátnych atribútov danej triedy. Implementácia metódy klonovania je vo všetkých triedach veľmi podobná. Metóda vytvorí objekt aktuálnej triedy a prenesie všetky hodnoty atribútov starého objektu do nového. Ak neexistuje, tak je potrebné vytvorenie konštruktoru, ktorý nastaví všetky atribúty [14]. Keďže sa vytvára kópia inštancie vo vnútri triedy, tak je možné kopírovať privátne atribúty, pretože väčšina programovacích jazykov(vrátane Javy) umožňuje objektom prístup k privátnym atribútom iných inštancii rovnakej triedy.

Po dokončení implementácie štvrtej iterácie v semestrálnom projekte pribudlo rozhranie vytvorené študentmi, ktoré predstavuje hrateľnosť. Zároveň triedam reprezentujúce šachovnicu a hru boli pridané metódy.

2.5 Piata iterácia

Piata iterácia slúži na precvičenie si dedičnosti, abstraktných tried a práce s reťazcami v cykle. Úlohou študentov je vytvorenie dvoch tried ktoré reprezentujú šachy a dámu. Z pôvodnej triedy reprezentujúcu hru je potrebné spraviť abstraktnú triedu, z ktorej šach a dáma budú dediť. Spoločnú funkcionalitu vložia do abstraktnej triedy a špecifické metódy implementujú do jednotlivých tried. Výslednú hierarchiu je vidno na obrázku 2.1, kde modré šípky predstavujú dedenie z nadtriedy a zelená prerušovaná šípka predstavuje implementáciu rozhrania triedou. Pomocou tejto hierarchie sú ukázané výhody dedičnosti aj abstraktných tried. Zabraňujú opakovaniu kódu, sprehľadňujú kód a pripravia kód na jednoduché pridanie ďalšej funkcionality. Pridanie

ďalšej hry by si vyžadovalo iba vytvorenie jednej triedy a napísanie pár metód.



Obr. 2.1: Hierarchia dedičnosti hier v piatej iterácii

Zadanie vyžaduje prepísanie *toString* metódy v šachovnici, aby pri hraní hier bolo prehľadne vidieť v akej pozícii sa nachádzame. Na spájanie reťazcov v cykle sa používa *StringBuilder* kvôli lepšej časovej zložitosti. Zatiaľ figúrky v iterácii reprezentujú písmenka, čo má za dôsledok nerozlíšiteľnosť figúrok opačnej farby. V siedmej iterácii je to vyriešené pomocou unicode znakov.

Po dokončení implementácie piatej iterácie v semestrálnom projekte pribudli triedy reprezentujúce šach a dámu. Z triedy predstavujúci hru sa stala abstraktná. Zároveň typ figúrky bol rozšírený o typy figúrok z dámy. Na obrázku 2.2 je zobrazené ako sa šachovnica zobrazuje po vypísaní do konzoly.

```
   1   2   3   4   5   6   7   8
  -------------------------------
A | R | P |   |   |   |   | P | R |
  -------------------------------
B | K | P |   |   |   |   | P | K |
  -------------------------------
C | B | P |   |   |   |   | P | B |
  -------------------------------
D | Q | P |   |   |   |   | P | Q |
  -------------------------------
E | K | P |   |   |   |   | P | K |
  -------------------------------
F | B | P |   |   |   |   | P | B |
  -------------------------------
G | K | P |   |   |   |   | P | K |
  -------------------------------
H | R | P |   |   |   |   | P | R |
  -------------------------------
```

Obr. 2.2: Textová reprezentácia šachovnice v piatej iterácii

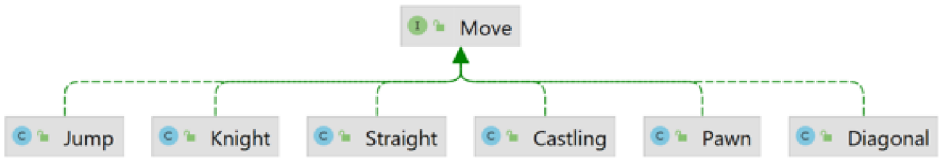
2.6 Šiesta iterácia

Šiesta iterácia slúži na precvičenie si porovnávania objektov a jednoduchú prácu s kolekciami. Existujúcim triedam je potrebné prepísať metódu *equals* a tým definovať, kedy sa dva objekty rovnajú. Testy kontrolujú, či správne prepísali aj metódu *hashCode*, ktorú je nutné prepísať vždy, keď prepisujeme *equals*, aby napríklad nevznikli konflikty pri ukladaní do kolekcií [15].

2.6.1 Návrhový vzor Strategy

V semestrálnom projekte v šiestej až ôsmej iterácii sa implementuje návrhový vzor strategy na zavedenie rôznych druhov pohybov figúrok na šachovnici. Strategy je behaviorálny návrhový vzor, ktorý umožňuje definovať rodinu algoritmov, umiestniť každý z nich do samostatnej triedy a zabezpečiť, aby boli ich objekty vzájomne zameniteľné. Kľúčom k aplikovaniu vzoru je navrhovanie rozhranie pre strategy a jeho kontext, ktorý je dostatočne všeobecný na to, aby podporoval celý rad algoritmov. Nemali by byť nutné zmeniť rozhranie, aby podporovali nový algoritmus [16].

Každá figúrka má zoznam pohybov, kde pohyb je trieda, ktorá dedí z rozhrania *Move*. Rozhranie obsahuje len jednu nestatickú metódu na zistenie všetkých možných ťahov danej figúrky, ktorá môže mať rôzny počet druhov pohybov. Napríklad šachová figúrka veža má len jeden druh pohybu na rozdiel od šachovej figúrky kráľ, ktorý má tri druhy pohybov. V rozhraní sa nachádza aj jedna pomocná statická metóda, ktorá má uľahčiť implementáciu algoritmov a demonštrovať použitie statických metód v rozhraní. Na obrázku 2.3 vidno hierarchiu návrhového vzoru strategy po ôsmej iterácii. V šiestej iterácii obsahuje len *Diagonal*, *Jump* a *Straight*.

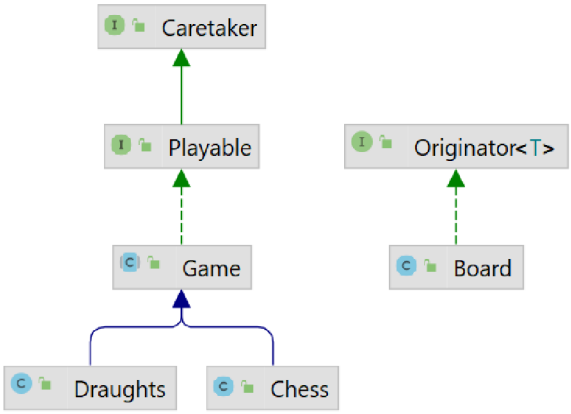


Obr. 2.3: Hierarchia návrhového vzoru stratégia po ôsmej iterácii

2.6.2 Návrhový vzor Memento

Hra na šachovnici by mala mať možnosť vrátiť ťah. Na túto funkcionalitu slúži behaviorálny návrhový vzor memento, ktorý umožňuje uložiť a obnoviť predchádzajúci stav objektu bez odhalenia podrobností o jeho implementácii. Návrhový vzor memento, ktorý je ukázaný na obrázku 2.4, kde zelená šípka znamená rozširovanie rozhrania rozhraním, pozostáva z:

- snímky stavu pôvodcu známej ako *memento*, ktorý niekedy splýva s pôvodcom.
- pôvodcu známeho ako *originator*, ktorý môže vytvárať snímky svojho vlastného stavu a v prípade potreby obnoviť svoj stav zo snímok.
- správcu známeho ako *caretaker*, ktorý si uchováva snímky a ovláda kedy pôvodca snímku vytvorí alebo obnoví [16].



Obr. 2.4: Hierarchia návrhového vzoru memento v šiestej iterácii

Po dokončení implementácie šiestej iterácie v semestrálnom projekte pribudol balík s triedami reprezentujúcimi skok, diagonálny a priamy pohyb. Zároveň bola prepísaná rovnosť tried a implementovaný návrhový vzor memento.

2.7 Siedma iterácia

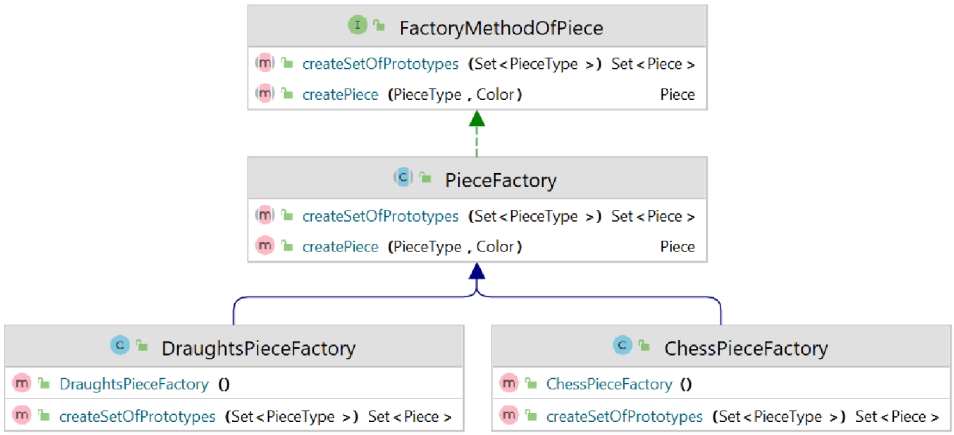
Siedma iterácia slúži na precvičenie si pokročilých kolekcii, prúdov, dedičnosti a základných výnimiek. Do vymenovaného typu je potrebné pridať metódu, ktorá vráti unicode znak reprezentujúci figúrku. Zadanie zakazuje použitie príkazov *if* alebo *switch* z dôvodu navedenia

na použitie vnorených kolekcii konkrétnejšie mapy a triedy reprezentujúcej dvojicu hodnôt.

Úlohou siedmej iterácie je tiež niekoľko metód, ktoré pracujú nad kolekciami. Zadanie vyžaduje použiť prúdy, ktoré ponúkajú funkcionálne metódy nad kolekciami. Testy kontrolujú, či študenti dodržiavajú zapuzdrenie a vracajú nové alebo nemodifikovateľné kolekcie.

2.7.1 Návrhový vzor Factory method

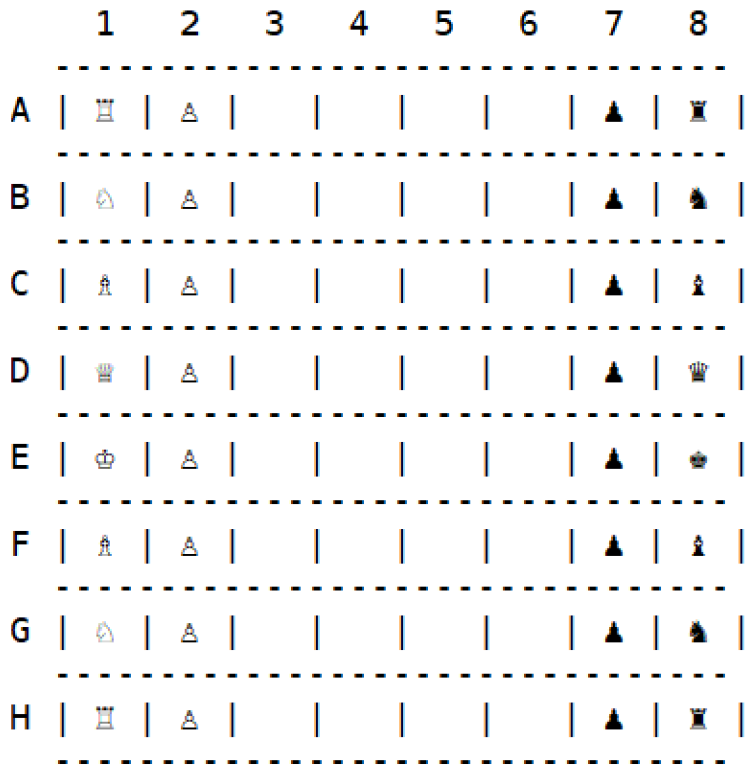
Factory method je tvorivý návrhový vzor, ktorý poskytuje rozhranie na vytváranie objektov v nadtriede, ale umožňuje podtriedam meniť typ objektov, ktoré budú vytvorené. V semestrálnom projekte je vytvorená abstraktná trieda reprezentujúca továreň na výrobu figúrok. Každej hre je vytvorená podtrieda, ktorá predá nadtriede množinu, v ktorej sú všetky možné typy figúrok v danej hre. Keď metóda dostane nepodporovaný vstup, tak vyhodí výnimku. Továreň obsahuje metódu na vytváranie figúrky na základe farby a typu. V atribúte si uchováva mapu figúrok a ich typov. Pomocou metódy z druhej iterácie, ktorá je súčasťou návrhového vzoru prototype, vracia prototypy figúrok.



Obr. 2.5: Hierarchia dedičnosti návrhového vzoru factory method po siedmej iterácii

Hierarchia použitého návrhového vzoru factory method v semestrálnom projekte je ukázaná na obrázku 2.5. Obrázok obsahuje aj hlavičky konštruktorov a metód, kvôli lepšiemu predstaveniu návrhového vzoru.

Po dokončení implementácie siedmej iterácie v semestrálnom projekte pribudla abstraktná továreň a továrne na výrobu figúrok pre šachu a dámu. Zároveň v balíku pohybov bol pridaný pohyb pešiaka a pohyb do tvaru písmena L. Textová reprezentácia šachovnice bola aktualizovaný(pozri obrázok 2.6), kvôli krajšiemu výpisu do konzoly.



Obr. 2.6: Textová reprezentácia šachovnice po siedmej iterácii

2.8 Ôsma iterácia

Ôsma iterácia slúži na precvičenie si usporiadaných kolekcii, anonymných tried a porovnanie časovej zložitosti zoznamu a množiny. Úlohou študentov je naimplementovat prirodzené usporiadanie na súradniciach pomocou implementácie rozhrania *Comparable*, kde si vyskúšaju aj prácu s generickými typmi. V jednej z metód je nutné si vytvoriť anonymnú triedu rozširujúcu triedu *Comparator*, ktorá bude porovnávať súradnice v opačnom poradí ako prirodzené usporiadanie naimplementované na súradniciach.

V ôsmej iterácii je potrebné zaviesť pohyb reprezentujúci rošádu. Na overenie splnenia podmienok rošády je potrebné použiť históriu ťahov, a teda návrhový vzor memento. Týmto pohybom dokončia všetky potrebné druhy pohybov figúrok v návrhovom vzore strategy.

Ďalej študenti otestujú časovú zložitosť zoznamu a množiny postupným pridávaním a odoberaním náhodných prvkov. Následne je potrebné výsledok napísať do konzoly.

Po dokončení implementácie ôsmej iterácie v semestrálnom projekte pribudla v balíku pohybov trieda reprezentujúca rošádu. Triede reprezentujúcej súradnice bolo pridané prirodzené usporiadanie a v šachu bola prianá metóda, ktorá pomáha zisťovať ohrozenie políčka.

2.9 Deviata iterácia

Deviata iterácia slúži na precvičenie vnútornej triedy, práce s generickými typmi, tvorby vlastných výnimiek a prácu s nimi. Študenti zovšeobecnia rozhranie *Prototype* použitím generických typov a naplno využijú potenciál návrhového vzoru 2.4.1.

Následne študenti vytvoria dve kontrolované a dve nekontrolované výnimky pomocou, ktorých upravia implementáciu. Vyskúšajú si zachytávanie aj vyhadzovanie výnimiek. Napríklad pri načítaní a pretypovaní vstupu od používateľa.

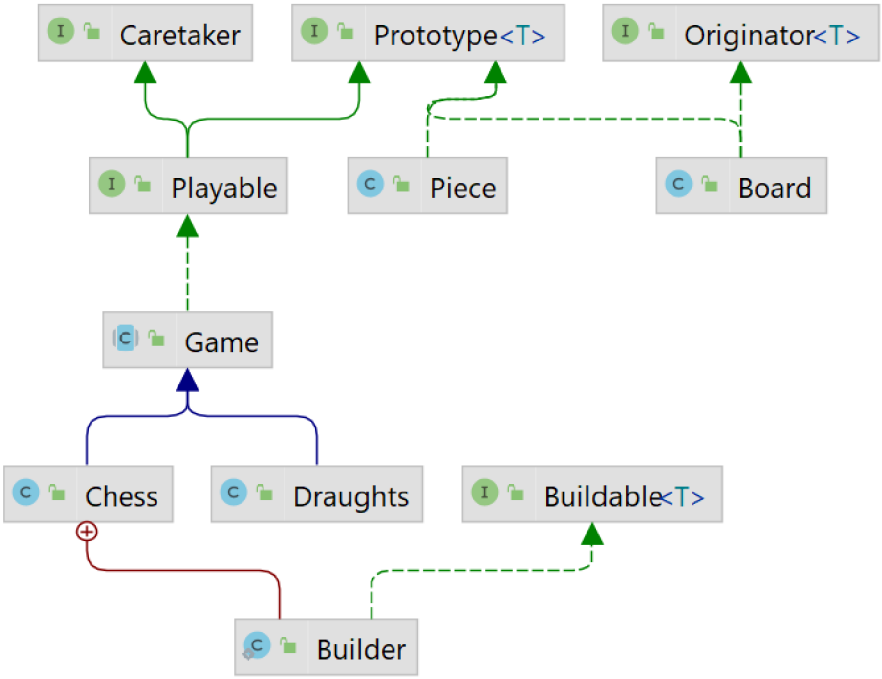
2.9.1 Návrhový vzor Builder

Pomocou vnorenej statickej triedy v triede reprezentujúcej šach je naimplementovaný tvorivý návrhový vzor builder, ktorý umožňuje vytvárať zložité objekty krok za krokom. Vzor umožňuje vytvárať

rôzne typy a reprezentácie objektu pomocou rovnakého konštrukčného kódu. V semestrálnom projekte je použitý na vytváranie rôznych inštancii šachu pomocou reťazového volania metód. Napríklad:

```
Chess game = new Chess.Builder()
        .addPlayer(new Player("Mat", Color.
           WHITE))
        .addPlayer(new Player("Pat", Color.
           BLACK))
        .addPieceToBoard(new ChessPieceFactory
           ().createPiece(PieceType.KING, Color.
           WHITE), 'e', 1)
    .build();
```

Na obrázku 2.7 je vidno hierarchiu dedičnosti po aplikovaní návrhového vzoru, kde červená čiara predstavuje vnútornú triedu.

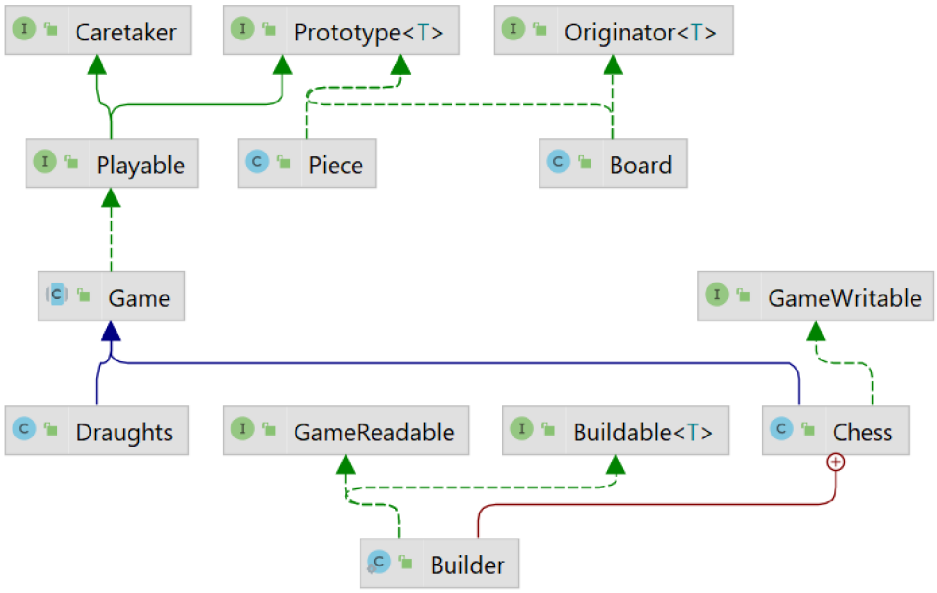


Obr. 2.7: Hierarchia dedičnosti po aplikovaní návrhového vzoru Builder

Po dokončení implementácie deviatej iterácie v semestrálnom projekte pribudla vnútorná statická trieda Builder a balík výnimiek so štyrmi výnimkami vytvorenými študentmi. Zároveň bola upravená implementácia hry za použitia výnimiek.

2.10 Desiata iterácia

Desiata iterácia slúži na precvičenie si tvorby testov, vstupu a výstupu. Predpripravené rozhrania poskytujú základnú štruktúru kódu(pozri obrázok 2.8) a zároveň ukazujú použitie predvolenej metódy[1]. Úlohou je naimplementovať možnosť uloženia a načítania hry z textového súboru v požadovanom formáte(pozri obrázok 2.9). Zároveň študenti naimplementujú možnosť uložiť hru vo formáte JSON[2], ktorý je nezávislý od programovacieho jazyka a je jednoduchý formát na výmenu údajov, lebo je čitateľný pre ľudí a ľahký na generovanie pre stroje [17].



Obr. 2.8: Záverečná hierarchia dedičnosti v semestránom projekte

1. po anglicky *default method*
2. *JavaScript Object Notation*

```
Mat-WHITE;Pat-BLACK
ROOK,WHITE;PAWN,WHITE;_;_;_;_;PAWN,BLACK;ROOK,BLACK
KNIGHT,WHITE;PAWN,WHITE;_;_;_;_;PAWN,BLACK;KNIGHT,BLACK
BISHOP,WHITE;PAWN,WHITE;_;_;_;_;PAWN,BLACK;BISHOP,BLACK
QUEEN,WHITE;PAWN,WHITE;_;_;_;_;PAWN,BLACK;QUEEN,BLACK
KING,WHITE;PAWN,WHITE;_;_;_;_;PAWN,BLACK;KING,BLACK
BISHOP,WHITE;PAWN,WHITE;_;_;_;_;PAWN,BLACK;BISHOP,BLACK
KNIGHT,WHITE;PAWN,WHITE;_;_;_;_;PAWN,BLACK;KNIGHT,BLACK
ROOK,WHITE;PAWN,WHITE;_;_;_;_;PAWN,BLACK;ROOK,BLACK
```

Obr. 2.9: Formát textovej reprezentácie iniciálnej pozície v šachu

Po dokončení implementácie desiatej iterácie v semestrálnom projekte pribudla načítania a zapísania hry do súboru. Zároveň v balíku *src/test/java/cz/muni/fi/pb162/project/testing/* pribudla trieda obsahujúca testy.

3 Správa repozitára

Zverejňovanie a úprava zadaní sa sprostredkúva pomocou školského serveru[1]. Plný prístup do súkromného repozitára semestrálneho projektu, majú len vyučujúci predmetu. Súkromný repozitár semestrálneho projektu pozostáva z:

- vetvy *main*, ktorá obsahuje základne informácie a odkazy,
- desiatich vetiev(napríklad *iteration-01*) s predpripravenými objektami, zadaniami a testami určenými pre študentov,
- desiatich vetiev(napríklad *iteration-01-impl*) so vzorovým riešením.

Názov semestrálneho projektu obsahuje aktuálny rok. Návody opísané v tejto kapitole uvádzam pre rok 2023. Pre použitie iného roku je potrebné v príkazoch zmeniť rok. Názvy verejného a súkromného repozitára sú *2023-pb162-seminar-project* a *2023-pb162-seminar-project-private*. Staré skripty na uľahčenie manipulácie s repozitármi sú uložené v repozitári s názvom *pb162-seminar-project-management* na serveri. Na manipuláciu s novým semestrálnym projektom ich bolo potrebne zmeniť. Momentálne sú v repozitári uložené staré skripty z dôvodu prebiehajúcej výuky a využívania staršieho semestrálneho projektu. V návode predpokladám, že už sú nahraté nové skripty. Ekvivalentným riešením je, že miesto naklonovania repozitára so skriptami, je potrebné si stiahnuť nové skripty do adresára, do ktorého si naklonujeme repozitár so semestrálnym projektom.

3.1 Založenie projektu

Na začiatku semestra je potrebné vytvoriť verejný repozitár, ktorý bude prístupný študentom a postupne sa v ňom budú zverejňovať zadania iterácii. Grafické znázornenie repozitárov po založení projektu je ukázané na obrázku 3.1.

1. https://gitlab.fi.muni.cz

Postup založenia projektu:

- Naklonovanie repozitára so skriptami a semestrálneho projektom so vzorovými riešeniami.

```
git clone git@gitlab.fi.muni.cz:pb162/pb162-
   seminar-project-management.git
cd pb162-seminar-project-management
git clone git@gitlab.fi.muni.cz:pb162/2023-
   pb162-seminar-project-private.git
```

- Vytvorenie kópie privátneho projektu kliknutím na *fork* na stránke repozitára projektu. Nastavenie *namespace* ako *pb162* a premenovanie projektu odstránením koncovky *-private*.

- Odstránenie odkazu na privátny projekt.

```
Settings -> General -> Advanced -> Remove
   fork relationship
```

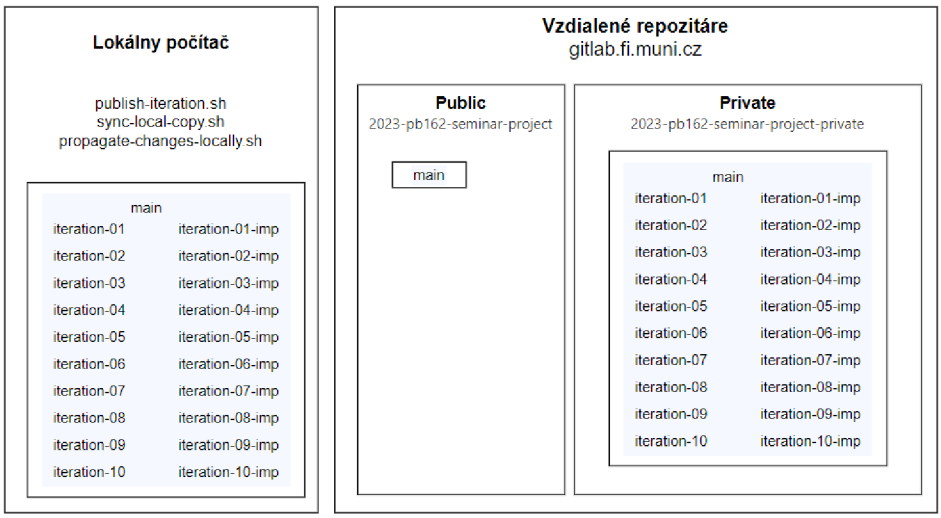
- Vymazanie všetkých vetiev okrem vetvy *main*.

```
Repository -> Branches
```

- Nastavenie viditeľnosti na *public* a vypnutie všetkých práv (po anglicky *permisions*) okrem:

  - *Repository: Everyone with Access.*
  - *Repository: Forks: Everyone with Access.*
  - *Repository: CI/CD: Everyone with Access.*

```
Settings -> General -> Visibility, project
   features, permissions ->  Project
   visibility
```



Obr. 3.1: Štruktúra repozitárov po založení projektu

3.2 Zverejnenie iterácii

V repozitári so skriptami sa nachádza skript na zverejnenie iterácii. Skript vyžaduje jeden vstupný parameter reprezentujúci číslo iterácie, ktorú chceme zverejniť. Napríklad *publish-iteration.sh 01*.

Rozpísaný postup zverejnenia iterácii zo skriptu:

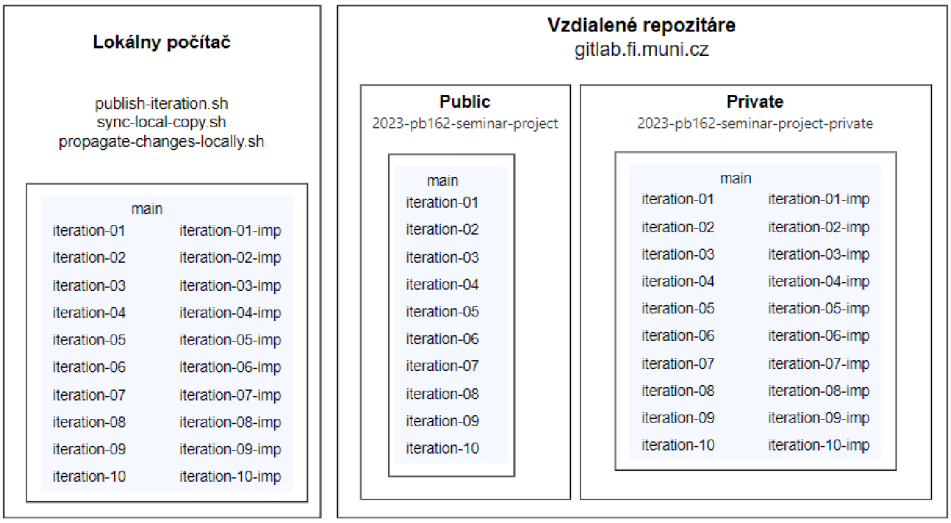
- Overenie naklonovania privátneho repozitára. V prípade, že ešte nie je naklonovaný, tak skript skončí a vypíše chybovú hlášku.
- Overenie zadania parametra, ktorý reprezentuje číslo iterácie, ktorú chceme zverejniť. V prípade, že ešte nie je zadané, tak skript skončí a vypíše chybovú hlášku.
- Pridanie odkazu na verejný repozitár pod názvom *public* ako vzdialený repozitár.

  ```
  git remote add public 2023-pb162-seminar-
     project.git
  ```

- Pridanie novej vetvy, s číslom iterácie uloženej *$1*(vstupný parameter), do verejného repozitára.

```
git push public iteration-\$1:iteration-\$1
```

Na obrázku 3.2 vidno štruktúru repozitárov na konci semestra. Na začiatku semestra verejný repozitár obsahuje len jednu vetvu. Postupným zverejňovaním iterácii sa pridávajú vetvy zo súkromného do verejného repozitára.



Obr. 3.2: Štruktúra repozitárov na konci semestra

3.3 Úprava projektu

Pre vykonávanie zmien v semestrálnom projekte je nutné mať naklonovaný repozitár so skriptami aj semestrálnym projektom podľa postupu spomínaného v sekcii 3.1.

Následne je potrebné zosynchronizovať lokálnu verziu s repozitárom na Gitlabe pomocou skriptu *sync-local-copy.sh*. Rozpísaný postup synchronizácie zo skriptu:

- Overenie naklonovania privátneho repozitára. V prípade, že ešte nie je naklonovaný, tak skript skončí a vypíše chybovú hlášku.
- Uloženie si aktuálnej pozície.

- V cykle postupné prepínanie medzi všetkými vetvami, stiahnutie a nahratie verzie zo vzdialeného repozitára.

```
git checkout vetva
git pull
git push
```

- Vrátenie sa do uloženej pozície.

Všetky zmeny, ktorými chceme upraviť semestrálny projekt, je potrebné uložiť pomocou príkazu *commit*. Následne pomocou interaktívneho skriptu *propagate-changes-locally.sh* ich zaviesť do potrebných vetiev. Skript dokáže propagovať len poslednú uloženú zmenu. Aplikácia zmien medzi vetvami sa uskutočňuje pomocou príkazu *cherry pick*, ktorý umožňuje integrovať vybrané jednotlivé zmeny z ktorejkoľvek vetvy do aktuálnej vetvy [18].

Skript *propagate-changes-locally.sh* postupne zavedie zmenu do všetkých potrebných vetiev. Tento proces vieme rozdeliť na tri časti, ktoré sú farebne znázornené na obrázku 3.3.

- Modré šípky. Zmeny vykonané vo vetve bez vzorovej implementácie aplikuje do vetvy s implementáciou a rovnakým číslom.

- Zelené šípky. Zmeny vykonané vo vetve bez vzorovej implementácie alebo vo vetve *main* aplikuje do vetiev bez implementácie s vyšším poradovým číslo.

- Fialové šípky. Zmeny vykonané vo vetve so vzorovou implementáciou aplikuje do vetiev s implementáciou s vyšším poradovým číslo.

Napríklad zmena vykonaná vo vetve *iteration-09* sa aplikuje počas prvej časti do *iteration-09-impl*, počas druhej časti do *iteration-10* a počas tretej časti do *iteration-10-impl*.

Obr. 3.3: Grafické znázornenie skriptu *propagate-changes-locally.sh*

Záver

Cieľom mojej bakalárskej práce bolo preštudovať študijne materiály a existujúci semestrálny projekt predmetu *PB162 Programování v jazyce Java* a na jeho základe navrhnúť nové zadanie rozdelené do desiatich iterácii. Zadanie malo kopírovať preberanú látku počas semestra a náročnosť vypracovania jednotlivých iterácii mala odpovedať približne dvoch hodinám programovania. Zároveň každá iterácia mala mať textové zadanie napísane v anglickom jazyku a byť pokrytá jednotkovými testami.

Preštudovala som si semestrálny projekt a interaktívnu osnovu predmetu, ktorá obsahuje prezentácie a doplnkové študijne materiály. Zároveň som sa podieľala na výuke predmetu, vďaka čomu som opravovala vypracované iterácie študentom.

Pri návrhu nového zadania som sa snažila vymyslieť tému blízku študentom, ktorá by obsahovala možnosť na predcvičenie všetkých potrebných konceptom preberaných na predmete. Zároveň som chcela pridať možnosť precvičiť si nové koncepty z najnovšej verzie JDK, ktoré boli tento rok začlenené do výuky.

V úvode práce bol predstavený problém a postup implementácie. Následne je rozpísané, ktoré koncepty sa precvičujú v jednotlivých iteráciách. Na záver je popísaný návod na správu projektu. Konkrétne na zverejnenie a úpravu semestrálneho projektu na školskom serveri s názvom Gitlab.

V budúcnosti je možné semestrálny projekt aktualizovať o nové koncepty vyučujúce v predmete pomocou návodu popísaného v tretej kapitole. Pre lepšiu hrateľnosť výslednej aplikácie je možné pridanie umelej inteligencie proti ktorej by študenti hrali alebo grafické rozhranie predstavujúce šachovnicu.

A Prílohy

Elektronická príloha v informačnom systéme obsahuje priečinky:

- *implementation* so zdrojový kódom desiatej, a teda poslednej iterácie aj so vzorovým riešením,
- *assignments* s desiatimi textovými zadaniami iterácii v anglickom jazyku a
- *skripts* s upravenými skriptami na manipuláciu s repozitármi.

Bibliografia

1. *FI:PB162 Programování Java* [online]. Brno: Masaryk University, 2022 [cit. 2022-05-18]. Dostupné z: https://is.muni.cz/predmet/fi/jaro2022/PB162.
2. HAVELKA, Arnošt; PECINOVSKÝ, Rudolf. *JUnit 5: jednotkové testování na platformě Java*. Praha: Grada Publishing, 2018. ISBN 978-80-271-0733-9.
3. JANZEN, David; SAIEDIAN, Hossein. Test-driven development concepts, taxonomy, and future direction. *Computer*. 2005, roč. 38, č. 9, s. 43–50. Dostupné z DOI: 10.1109/MC.2005.314.
4. *Oracle Java SE Support Roadmap* [online]. Oracle Corporation [cit. 2022-05-17]. Dostupné z: https://www.oracle.com/java/technologies/java-se-support-roadmap.html.
5. *Java Downloads* [online]. Oracle Corporation, 2022 [cit. 2022-03-19]. Dostupné z: https://www.oracle.com/java/technologies/downloads.
6. *How much faster is Java 17?* [online]. OptaPlanner, 2021-09-15 [cit. 2022-03-19]. Dostupné z: https://www.optaplanner.org/blog/2021/09/15/HowMuchFasterIsJava17.html.
7. REFACTORING GURU. *Design patterns* [online] [cit. 2022-05-18]. Dostupné z: https://refactoring.guru/design-patterns.
8. *Git* [online]. Gitlab, 2022-03-28 [cit. 2022-05-14]. Dostupné z: https://docs.gitlab.com/ee/topics/git/.
9. CHACON, Scott; STRAUB, Ben. *Pro Git*. .2nd ed. Apress, [n.d.]. ISBN 978-80-904248-1-4. Dostupné tiež z: https://progit2.s3.amazonaws.com/en/2016-03-22-f3531/progit-en.1084.pdf.
10. BIBERSTEIN, Marina; GIL, Joseph (Yossi); PORAT, Sara. Sealing, Encapsulation, and Mutablility. In: KNUDSEN, Jørgen Lindskov (ed.). *ECOOP 2001 — Object-Oriented Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, s. 28–52. ISBN 978-3-540-45337-6.

11. *Java Downloads* [online]. Oracle Corporation, 2020-11-03 [cit. 2022-03-19]. Dostupné z: `https://docs.oracle.com/en/java/javase/14/language/records.html`.

12. PECINOVSKÝ, Rudolf. *Java 14: kompletní příručka jazyka*. Praha: Grada Publishing, 2020. ISBN 978-80-271-1369-9.

13. BLOCH, Joshua. *Effective Java*. 3. vyd. USA: Prentice Hall PTR, 2008. ISBN 0321356683. Dostupné tiež z: `https://kea.nu/files/textbooks/new/Effective%20Java%20%282017%2C%20Addison-Wesley%29.pdf`.

14. PECINOVSKÝ, Rudolf. *Návrhové vzory: 33 vzorových postupů pro objektové programování*. Brno: Computer Press, 2007. ISBN 978-80-251-1582-4.

15. PITNER, Tomáš. *Java - začínáme programovat: podrobný průvodce začínajícího uživatele*. Praha: Grada, 2002. ISBN 80-247-0295-9.

16. GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995. Dostupné tiež z: `http://www.javier8a.com/itc/bd1/articulo.pdf`.

17. FRIESEN, Jeff. *Java XML and JSON*. Springer, 2016. Dostupné tiež z: `https://link.springer.com/content/pdf/bfm%253A978-1-4842-1916-4%252F1.pdf`.

18. *Git®: Notes for Professionals*. Goalkicker.com Free Programming Books, 2018.