**Vysoká škola polytechnická Jihlava**

Aplikovaná informatika

**Dokumentace úkolů**

Seminární práce

Autor práce: Kryštof Malinda, Ondřej Kohout

Předmět: Programovaní v Javě

Vyučující: Ing. Marek Musil

Jihlava 06. 12. 2023

Obsah

[1 Úkol 1 (Téma 2) 4](#__RefHeading___Toc996_4227626128)

[2 Úkol 2 (Téma 2) 5](#__RefHeading___Toc996_4227626128_kopie_1)

[3 Úkol 3 (Téma 4) 6](#__RefHeading___Toc996_4227626128_kopie_2)

[4 Úkol 4 (Téma 4) 7](#__RefHeading___Toc996_4227626128_kopie_3)

[5 Úkol 5 (Téma 3a) 8](#__RefHeading___Toc996_4227626128_kopie_4)

[6 Úkol 6 (Téma 3b) 9](#__RefHeading___Toc996_4227626128_kopie_5)

[7 Úkol 7 (Téma 6a) 10](#__RefHeading___Toc996_4227626128_kopie_6)

[8 Úkol 8 (Téma 6b) 11](#__RefHeading___Toc996_4227626128_kopie_7)

[8.1 Úkol 8 část 1 11](#__RefHeading___Toc996_4227626128_kopie_8)

[8.2 Úkol 8 část 2 11](#__RefHeading___Toc1110_4227626128)

[8.3 Úkol 8 část 3 11](#__RefHeading___Toc1112_4227626128)

[8.4 Úkol 8 část 4 12](#__RefHeading___Toc1114_4227626128)

[9 Úkol 9 (Téma 7) 13](#__RefHeading___Toc1123_4227626128)

[10 Úkol 10 (Téma 5) 14](#__RefHeading___Toc1125_4227626128)

# Úkol 1 (Téma 2)

V prvním kódu (NacteniHodnot.java) jsem vytvořil statické metody pro načítání hodnot různých datových typů s ošetřením chybových stavů pomocí try-catch bloků. Pro zpracování chyb jsem implementoval dvě metody, handleInputError a handleFormatError, které poskytují informace o špatném formátu nebo překročení rozsahu hodnoty.

V druhém kódu (ConsoleApp.java) jsem vytvořil metody pro jednotlivé části zadání. V metodě u\_i() jsem využil(a) try-catch bloky pro načítání hodnot a informoval uživatele o neplatných hodnotách. V metodě u\_ii() jsem provedl deklaraci a inicializaci proměnných různých datových typů. Metoda u\_iii() načítá jméno a příjmení a vrací je spojené do jednoho řetězce. Metoda u\_iv\_datovetypy() vytváří přehledovou tabulku datových typů v jazyce Java. Nakonec jsem implementoval metodu u\_v() pro zobrazení výsledků výrazů.

ZnakText, kde ve main metodě jsou volány různé metody pro práci s ASCII znaky. Metoda TiskInfoZnak využívá DejInfoZnak pro získání informací o zadaném znaku, zatímco metoda tiskASCII generuje formátovanou ASCII tabulku pro písmena A-Z, jejich malé varianty a číslice 0-9. Druhá metoda DejASCII implementuje stejnou tabulku, ale s využitím StringBuilder pro efektivnější manipulaci s řetězcem. Celkově máte dobře strukturovaný kód s efektivním formátováním výstupu a elegantní manipulací s řetězci. Pro další optimalizaci můžete zvážit centralizaci definice ASCII tabulky k zabránění duplikaci kódu mezi metodami tiskASCII a DejASCII.

# Úkol 2 (Téma 2)

HraPosunPoSachovnici a RealizaceHryKonzola, které implementují hru pohybu na šachovnici v konzoli. Třída HraPosunPoSachovnici obsahuje logiku pohybu, ukládání historie a základní informace o stavu hry. Třída RealizaceHryKonzola pak využívá tuto logiku pro konzolovou interakci s uživatelem.

V HraPosunPoSachovnici jsem implementoval metody pro pohyb figurky na šachovnici (pohybChar), ukládání historie (addHistroy a moveHistory), a získávání pozice figurky. Kromě toho máte konstruktor pro vytvoření šachovnice a inicializaci proměnných.

V RealizaceHryKonzola jsem implementoval metodu startGame, která spouští hru a reaguje na vstup od uživatele, přičemž umožňuje pohyb figurky, zobrazení historie a ukončení hry. Metoda printBoard slouží k vizualizaci aktuálního stavu šachovnice v konzoli. Dále máte metodu validationOfInput pro ověření správnosti vstupu uživatele.

# Úkol 3 (Téma 4)

Třída KomplexniCislo, která modeluje komplexní čísla v programu pomocí dvou složek: reálné části (real) a imaginární části (imag). Tato třída poskytuje konstruktor pro inicializaci komplexního čísla, metody pro získání reálné a imaginární části (getReal a getImag), a přetížené metody pro aritmetické operace s komplexními čísly a čísly (sčítání, odčítání, násobení).

Instance KomplexniCislo pomocí konstruktoru a provedl s nimi několik aritmetických operací, včetně sčítání (plus), odčítání (minus) a násobení (krat). Dále jste přidal(a) operaci sčítání s číslem. Kód také obsahuje přetíženou metodu toString, která vrací textový popis komplexního čísla ve formátu "reálná část + imaginární části i".

V Main programu jsem vytvořil instance komplexních čísel, provedl s nimi operace a vypsali výsledky na konzoli. Celkově je kód dobře čitelný, strukturovaný a demonstruje principy objektově orientovaného programování, zejména práci s objekty a přetížení metod.

# Úkol 4 (Téma 4)

Třídy pro reprezentaci barev a implementoval rozšíření pro práci s modely RGB a CMYK. Základní třída Barva obsahuje složky pro reprezentaci barvy (červená, zelená, modrá), intenzitu a metody pro výpočet intenzity a formátování barvy do řetězce. Následně jsem vytvořil třídy BarvaRGB1, BarvaRGB2 a BarvaCMYK, které dědí od základní třídy a rozšiřují ji o specifické vlastnosti týkající se modelů barev RGB a CMYK.

Třídy BarvaRGB1 a BarvaRGB2 reprezentují barvy v modelu RGB. Tyto třídy mají konstruktor pro nastavení složek barvy a metodu pro změnu těchto složek. Vámi implementované metody jsou jednoduché a přehledné.

Třída BarvaCMYK reprezentuje barvu v modelu CMYK. Má konstruktor pro inicializaci barvy v tomto modelu a metodu pro změnu složek CMYK. Dále obsahuje privátní metodu vypoctiRGB(), která na základě hodnot CMYK vypočítá hodnoty RGB a intenzitu barvy.

Ve vašem Main programu jsem vytvořil instance všech tříd a vypsal formátované informace o barvách na konzoli. Celkově je kód dobře strukturovaný, a demonstruje principy dědičnosti, polymorfismu a práci s konstruktory a metodami.

# Úkol 5 (Téma 3a)

Java konzolová aplikace obsahuje sadu operací, mezi něž uživatel vybírá prostřednictvím zadání odpovídajícího čísla. Mezi operacemi patří výpočet mocnin a odmocnin čísla, goniometrických funkcí, zobrazení obsahu složky na disku, kopírování, odstranění a přejmenování souboru nebo adresáře, výpočet vlastností pravoúhlého a obecného trojúhelníku, ověření platnosti dvou zadaných dat a zobrazení aktuálního data a času. Kód je strukturován efektivně s využitím oddělených metod pro každou operaci, což zvyšuje čitelnost a údržbu kódu. Vstupy jsou řádně ošetřeny a aplikace pracuje s různými typy operací souborového systému.

# Úkol 6 (Téma 3b)

VstupniOperace (NacteniHodnot): Tato třída poskytuje statické metody pro načítání hodnot z konzole, včetně celých a desetinných čísel. Obsahuje metody nactiInt() pro načtení a vrácení celého čísla a nactiDouble() pro načtení a vrácení desetinného čísla.

KvadratickaRovniceVypocty: Statická třída poskytuje metody pro výpočty související s kvadratickými rovnicemi, jako je výpočet hodnoty kvadratické rovnice, určení počtu reálných kořenů a výpis hodnot kořenů na konzoli.

KvadratickaRovnice: Třída reprezentující kvadratickou rovnici umožňuje manipulaci s jejími koeficienty pomocí getterů a setterů.

MojeMath: Statická třída pro výpočty průměru a rozptylu sekvence čísel. Obsahuje metody prumer(), rozptyl() a prumer\_a\_rozptyl().

Main: Testovací kód obsahuje scénáře pro ověření funkcí tříd, včetně načítání hodnot z konzole, výpočtů kvadratické rovnice a výpočtů průměru a rozptylu.

Poznámky: Třída NacteniHodnot vyžaduje vytvoření instance skeneru pro načítání hodnot z konzole. Celkově projekt poskytuje nástroje pro práci s kvadratickými rovnicemi a výpočty statistických ukazatelů ze vstupních hodnot.

# Úkol 7 (Téma 6a)

HraPosunPoSachovnici a RealizaceHryKonzola. První třída implementuje jednoduchou hru, kde hráč může pohybovat znakem po šachovnici, ukládat historii pohybů a získávat statistiky o pohybech. Druhá třída obsahuje kód pro ovládání hry pomocí konzole.

V HraPosunPoSachovnici jsou definovány enumerátory pro směry pohybu, včetně metod pro pohyb a ukládání historie pohybů. Výsledná šachovnice je vizualizována v konzoli. Kromě toho je implementována funkcionalita pro výpočet geometrické vzdálenosti a sestavení výsledné cesty na základě historie pohybů.

RealizaceHryKonzola obsahuje kód pro spuštění hry v konzoli, včetně načítání rozměrů šachovnice od uživatele a interakce s hrou pomocí příkazů zadávaných z konzole.

Projekt je strukturován a poskytuje jednoduché uživatelské rozhraní pro ovládání hry. Implementuje principy objektově orientovaného programování a interakci s uživatelem prostřednictvím konzole.

Třída Souradnice obsahuje jednoduchou definici souřadnic s veřejnými atributy x a y, které jsou inicializovány v konstruktoru.

Třída ToolHistoriePohybu poskytuje nástroje pro práci s historií pohybu, která je reprezentována seznamem objektů typu Souradnice. Metoda spoctiVzdalenost vypočítá geometrickou vzdálenost mezi prvním a posledním bodem v historii. Funkce vrací hodnotu typu double.

Metoda sestavVyslednouCestu sestavuje textový řetězec reprezentující všechny body v historii pohybu ve formátu [x;y] -> [x;y] -> .... Tato funkce vrací výsledný řetězec typu String. Oba nástroje jsou navrženy tak, aby byly použity ve třídě HraPosunPoSachovnici pro získání informací o pohybu postavičky.

# Úkol 8 (Téma 6b)

## Úkol 8 část 1

V poskytnutém Java kódu v souboru Ukol1 jsou implementovány funkce pro parsování vstupních řetězců. Metody jako parseJmenoPrijmeni1 a parseJmenoPrijmeni2 slouží k extrakci jmen a příjmení z řetězců, přičemž druhá metoda provádí dodatečnou úpravu vstupu. Další metody, jako parseLogin, parseCislaInt a parseCislaDouble, se specializují na zpracování řetězců obsahujících loginy, celá čísla a desetinná čísla, přičemž výsledky jsou buď vypisovány nebo vraceny jako pole.

V souboru Main jsou tyto metody volány s konkrétními vstupy pro demonstrování jejich funkcionality. To zahrnuje také výpis výsledků na konzoli. Celkově jde o demonstrační účely a testování implementovaných funkcí pro zpracování různých typů vstupů.

## Úkol 8 část 2

V kódovém projektu jsou definovány třídy pro reprezentaci matematických funkcí - Funkce, Funkce\_Kvadraticka a Funkce\_Primka. Abstraktní třída Funkce obsahuje abstraktní metodu pro výpočet hodnoty y pro zadané x. Třídy Funkce\_Kvadraticka a Funkce\_Primka jsou odvozeny od této abstraktní třídy a implementují konkrétní matematické funkce, kvadratickou rovnici a přímku.

V Main.java je vytvořeno několik instancí těchto funkcí, a to přímky a kvadratické rovnice. Ty jsou uloženy v kolekci a následně vypsány na konzoli. Je také demonstrováno filtrování funkcí podle typu, kdy jsou tisknuty pouze přímkové nebo pouze kvadratické funkce.

Celkově je kód organizovaný, a představuje jednoduchý model matematických funkcí s možností práce s různými typy funkcí a ukládáním do kolekce.

## Úkol 8 část 3

Poskytnutý kód obsahuje implementaci hierarchie tříd pro hraní šachů. Třída HraSachovnice je abstraktní a definuje metodu execute(), která slouží k provádění her. Třídy HraSachovniceRandom a HraSachovniceManual jsou odvozeny od této abstraktní třídy a implementují konkrétní strategie pro hraní šachů.

V třídě HraSachovniceRandom je implementována metoda execute(), která simuluje automatizovaný náhodný pohyb na šachovnici. V tomto případě jsou generovány náhodné tahy pomocí třídy Random.

Třída HraSachovniceManual implementuje metodu execute(), která představuje ruční ovládání šachovnice. Uživatel je vyzván k zadání tahu prostřednictvím konzole, a zadaný tah je poté zpracován.

V souboru Main jsou tyto dvě strategie hry použity. V metodě main() jsou vytvořeny instance těchto strategií a následně jsou spuštěny voláním metody hra() s příslušnou instancí. Celkově jde o strukturu programu, který umožňuje provádět šachové hry s různými strategiemi.

## Úkol 8 část 4

Poskytnutý kód zahrnuje třídy reprezentující kvadratickou funkci (KvadratickaFunkce) a přímku (Primka). Dále existuje třída Ukol4, která obsahuje hlavní metodu pro demonstraci funkcí těchto tříd.

V metodě main třídy Ukol4 je vytvořena generická kolekce (ArrayList<Object> kolekce), do které jsou přidány instance objektů typu Primka a KvadratickaFunkce. Následně jsou všechny objekty v kolekci vytisknuty. Dále jsou zvlášť vytisknuty přímky a kvadratické funkce pomocí operátoru instanceof.

Třídy KvadratickaFunkce a Primka obsahují atributy a konstruktory pro inicializaci těchto atributů. Výpis informací o objektech (metoda toString), metoda equals a další metody mohou být součástí těchto tříd, ačkoliv ve zdrojovém kódu nejsou zobrazeny.

Celkově lze třídy použít pro reprezentaci a manipulaci s kvadratickými funkcemi a přímkami, a třída Ukol4 demonstruje jejich ukládání do kolekce a selektivní výpis podle typu.

# Úkol 9 (Téma 7)

Poskytnutý kód obsahuje několik tříd pro práci s daty, soubory a serializací. Třída **Datum** reprezentuje datum s atributy den, měsíc a rok a implementuje rozhraní Serializable pro možnost serializace objektů. Třída **IOFileTool** obsahuje statické metody pro práci se soubory, včetně funkcí pro počítání znaků v textovém souboru, kopírování obsahu mezi soubory s odstraněním bílých znaků a serializaci/deserializaci objektů třídy Osoba. Třída **Osoba** představuje osobu s atributy jméno, příjmení, věk a datum narození. Hlavní třída **Main** slouží k testování funkcionalit těchto tříd, včetně vytváření a manipulace s objekty Datum a Osoba, a provádění operací souborů a serializace.

# Úkol 10 (Téma 5)

Hlavní třída **Main** obsahuje vstupní bod programu. Nejprve hráč zadává velikost hracího pole, přičemž je ošetřeno vstupními výjimkami pro zajištění správného zadání. Poté se vytvoří instance třídy **MinovePole** a hráč může postupně zadávat souřadnice políček k odhalení. Vstup je opět kontrolován na neplatné hodnoty.

Třída **MinovePole** představuje samotnou hru. V konstruktoru se vytvoří hrací pole a umístí miny. Metoda **createVisual** vytváří vizuální reprezentaci hracího pole, která je vypsána na konzoli po každém kroku hráče. Metoda **checkValue** slouží k ověření hodnoty na daných souřadnicích a zajišťuje správné chování hry v závislosti na odhaleném poli.