STŘEDNÍ PRŮMYSLOVÁ ŠKOLA  
MLADÁ BOLESLAV

**ROČNÍKOVÁ PRÁCE**

Mattias Justin Demjan

Mladá Boleslav 2022

STŘEDNÍ PRŮMYSLOVÁ ŠKOLA  
MLADÁ BOLESLAV

**ROČNÍKOVÁ PRÁCE**

**Autor: Mattias Justin Demjan  
Studijní obor: 18-20-M/01 Informační technologie  
Vedoucí práce: Ing. Lukáš Mázl**

Mladá Boleslav 2022

# Obsah

[Obsah 3](#_Toc102338201)

[1 Úvod 6](#_Toc102338202)

[1.1 Použité programy 6](#_Toc102338203)

[2 Obsah práce 7](#_Toc102338204)

[2.1 Koncept programu 7](#_Toc102338205)

[2.2 Design programu 7](#_Toc102338206)

[2.3 Třídy 7](#_Toc102338207)

[2.3.1 Account 7](#_Toc102338208)

[2.3.2 Password 8](#_Toc102338209)

[2.3.3 Dates 9](#_Toc102338210)

[2.3.4 Ostatní třídy 10](#_Toc102338211)

[3 Závěr 12](#_Toc102338212)

[4 Přílohy 12](#_Toc102338213)

[4.1 Obrázky 12](#_Toc102338214)

[4.2 Zdroje 12](#_Toc102338215)

**Prohlášení**

Prohlašuji, že jsem svou ročníkovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti zpřístupňování této ročníkové práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

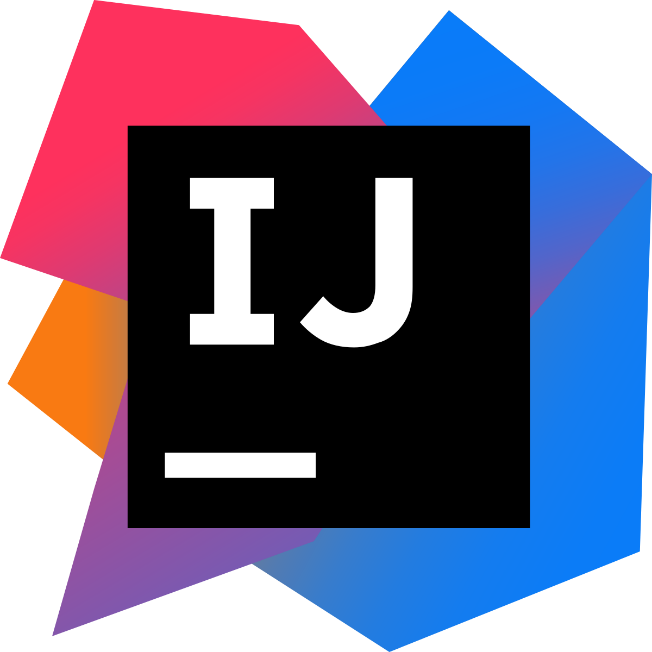
V Mladé Boleslavi dne podpis:

# Úvod

Zadání ročníkové práce znělo vytvořit simulátor bankomatu. Vývoj aplikací v programovacím jazyce Java mě baví, a vidím v něm svoji budoucí kariéru. I to byl jeden z důvodů, proč jsem si vybral programování této okenní aplikace.

## Použité programy

K programování této aplikace bylo použito vývojové prostředí IntelliJ IDEA v jazyce JavaFX verze 17 a s rozšířením Maven. FXML soubory byly upravovány pomocí Scene Builderu. K vytvoření obrázku pozadí byl použit freeware program Paint.net.

Obrázek 1 : logo IntelliJ [https://commons.wikimedia.org/wiki/File:IntelliJ\_IDEA\_Icon.svg]

# Obsah práce

## Koncept programu

Nejdříve bylo potřeba promyslet, jak by takový program vůbec fungoval, jak by pracoval a co všechno by obsahoval. Tento postup se vykonává při vytváření každého sofistikovanějšího programu. Program byl vytvořen pomocí OOP postupu a byly i využity metody z nové verze Javy.

## Design programu

Jako první věc bylo potřeba vytvořit samotný základ programu. Toho bylo docíleno pomocí Scene Builderu v JavaFX. Jako základ posloužil obrázek daný do pozadí programu v podobě bankomatu. Ten byl vytvořen v programu Paint.net podle předloh fotografií skutečných bankomatů.

Obrázek 2 : obrázek pozadí banomatu 

## Třídy

### Account

Třída Account má na starosti udržení hodnoty prostředků na účtě a pomocí ArrayListu držení dat z třídy Dates. Taktéž třída Accounts obsahuje metody na zápis informací do souboru, a následné čtení.

#### Čtení a zápis dat

Čtení i zápis dat je prováděn metodami getAccountBalance a setAccountBalance. Obojí pomocí BufferedReaderu a nebo BufferedWriteru. V obou metodách se jedná o textový dokument s názve account.txt, který je přímo uložený ve složce resources v programu. Metody getAccountBalance a setAccountBalance se používají mimo zápis a čtení dat také k nastavení či získání dat, k čemuž jsou metody set a get vždy zaměřeny.

#### Zjištění zůstatku

V tříde Account je taktéž metoda na zjištění zůstatku. Ve vstupu je požadováno číslo, avšak ve Stringu, které je následně porovnáno se zůstatkem na účtě pomocí metody getAccountBalance. Je-li na účtě dostatek pěnez, metoda vrátí konstanci true. Pokud není, vrátí false. Tato metoda velice ulehčila a zjednodušila způsob jakým lze zjistit zda-li má účet dostatek prostředků. Jedná se o ukázku OOP.

### Password

Třída password, jak již název napovídá, se zaobírá heslem, neboli spíše PINem, kterým se uživatel autentizuje pro vstup na účet do bankomatu. V první instanci je heslo uloženo pouze jako String, avšak v následujícíh metodách je zahashováno.

#### Hashovací algoritmus

K hashování hesla, a hashování vstupu od uživatele byl použit MD5 algoritmus. Tento algoritmus byl vyvinutý již v roce 1992 a používá funkci MessageDigest. Tento algoritmus je ovšem velmi jednoduše zlomitelný, a proto jeho užití v reálném světe je riskantní. Pro demonstrativní účely tohoto programu byl vybrán pro svoji jednoduchou implementaci a rychlost.

MessageDigest md = MessageDigest.getInstance("MD5");

md.update(neuPass.getBytes());

byte[] bytes = md.digest();

StringBuilder sb = new StringBuilder();

for (int i = 0; i < bytes.length; i++) {

sb.append(**Integer**.toString((bytes[i] & 0xff) + 0x100, 16).substring(1));

}

password = neuPass;

#### Zápis a čtení hesla

Jelikož je možnost pro uživatele heslo měnit, je heslo ukládano a čteno z textového dokumentu password.txt, který je ukládán v resources složce. Pro zápis a čtení byly znovu využity funkce BufferedReader a BufferedWriter. K těmto účelům mohl být taktéž použit FileWriter a FileReader. Rozdíl mezi nimi je ten, že File by měly být použity při zápisu a čtení méně dat. Buffer díky vnitřímu vyrování je rychlejší a méně zatěžující.

#### Autentifikace hesla

Metoda checkPass má za úkol zjištění, zda-li se heslo zadané uživatelem shoduje s heslem účtu. Na vstupu metody je String pass, neboli uživatelův vstup. Uvnitř metody dojde k zahashování vstupu od uživatele pomocí stejného algoritmu a pomocí stejného klíče. Následně se oba zahashované výsledky porovnají, a je-li přesná shoda metoda vrátí true. Není-li shoda metoda vrací false. S těmito hodnotami se pak dále v programu pracuje.

#### Změna hesla

Vyžádá-li si uživatel změnu hesla dojde k zavolání meody setPassword. Ve vstupu musí být zadáno nynější heslo pro kontrolu, a nové heslo, které chce uživatel nastavit. Přímo v metodě je podmínka kdy je zavolána metoda checkPass, do které se vloží vstup hesla od uživatele. Neshoduje-li se, vrátí se false. Pokud se shoduje, je nové heslo String neuPass zahashováno pomocí stejného algoritmu i klíče, a následně pomocí BufferedWriter zapsáno do password.txt. Taktéž metoda nastaví nové heslo do proměnné password uvnitř programu. Pokud se vše povede vrátí metoda true, se kterou se dá dále v programu pracovat.

### Dates

Třída Dates má za úkol uchovávat datum jejího dne a taktéž platební operaci ve Stringu. Je zde taktéž hlavní proměnná LocalDate pojmenovaná date a proměnná amount, neboli počet peněz.

#### Konstruktor

Pomocí kontruktoru se nastavují hodnoty do proměnných date, amount a operation. Při vyvolání třídy Dates je nutné uvést do kontruktoru tyto proměnné.

#### Vypsání

Metoda toString, je metoda která vrací date, amount a operation tak, aby vracela “hezky upravený“ Stringový text. Za proměnnou amount je uvedeno “CZK“ aby uživatel věděl o jakou hodnotu pěnez se jedná.

### Ostatní třídy

Zbytek tříd funguje jako prosté Controllery pro FXML soubory. V drtivé většině je implementováno Initializable, neboli metoda která provede věci ihned při načtení. To se využije například při zapsání dat nebo hodnot pro uživatele na výběr. Třídy jsou si pak například podobné stejnými metodami. Ve všech se tak setkáme s metodami pro buttony, jelikož rozložení bankomatu se nemění. Buttony jsou zde pro zadávání čísel a pro potvrzení či zamítnutí. Pokud ve třídě dochází k zápisu přes buttony či klávesnici od uživatele, je zamítací tlačítko použito k smazání znaků. Pokud iž v žádném poli znaky nejsou, a nebo nebyly vůbec uživatelem zadávány, program se ukončí, stejně jako reálný bankomat. Ve všech třídách je taktéž metoda backButton, která vrátí uživatele na hlavní stránku programu, odkud si vybírá své kroky. Ve všech třídách taktéž najdeme Labely, které slouží k informování uživatele, co má kam napsat, popřípadně že například špatně zadal heslo. V třídách se to dále různí na zápisu informací od uživatele, ale nejčastěji se setkáme s TextField a při výběru i s ChoiceBox. Většina tříd funguje tak, že uživatel něco udělá, například zadá informace nebo si něco vybere a daná akce se provede až po zmáčknutí tlačítka acceptButton čímž se provede metoda acceptButtonClick.

#### MainController

V hlavním controlleru, do kterého se uživatel dostane po úspěšné autentifikace, má uživatel na výběr pomocí buttonů “menu“, neboli “odkazy“ do dalších sekcí. Je zde možnost zkontrolovat zůstatek, vybrání hotovosti, dobití kreditu, změna PIN, poslání pěnez a vložit hotovost.

#### CheckBalanceController

Uživateli se po objeví zůstatek v hodnotě CZK, a pokud provedl uživatel jakoukoliv transakci v tomto bankomatu bez odhlášení se zde zobrazí i všechny transakce. To je docíleno pomocí Dates a Account třídy. Tam je pak vypsáno datum, typ transakce a počet peněz.

**public** void buttonOneClick(ActionEvent actionEvent) {

amountField.setText(amountField.getText()+**String**.valueOf(1));

}

**public** void buttonTwoClick(ActionEvent actionEvent) {

amountField.setText(amountField.getText()+**String**.valueOf(2));

}

**public** void buttonThreeClick(ActionEvent actionEvent) {

amountField.setText(amountField.getText()+**String**.valueOf(3));

}

#### CashOutController

V této třídě si může uživatel vybrat částku a to buď tak, že si vybere z ChoiceBox nebo napíše do TextField vlastní. Poté co tak učiní a zmáčkně acceptButton, dojde k ověření zda-li je na účtě dostatek prostředků pomocí checkBalance metody z Account třídy. Není-li na účtě dostatek prostředků, je to uživateli vypsáno, aby změnil hodnotu na menší. Pokud na účtě dostatek prostředků je, pak v hodnotě uživatelského výběru “vyjede“ z bankomatu bankovka, kterou si uživatel vezme pomocí kliknutí na ni.

#### CreditPhoneController

V této třídě může uživatel dobít svůj mobilní telefon. Po zadání požadovaných hodnot do TextFieldu a kliknutí na acceptButton se znovu provede metoda checkBalance a taktéž se zkontroluje zda-li je telefonní číslo zadáno správně, podle počtu číslic.

#### ChangePinController

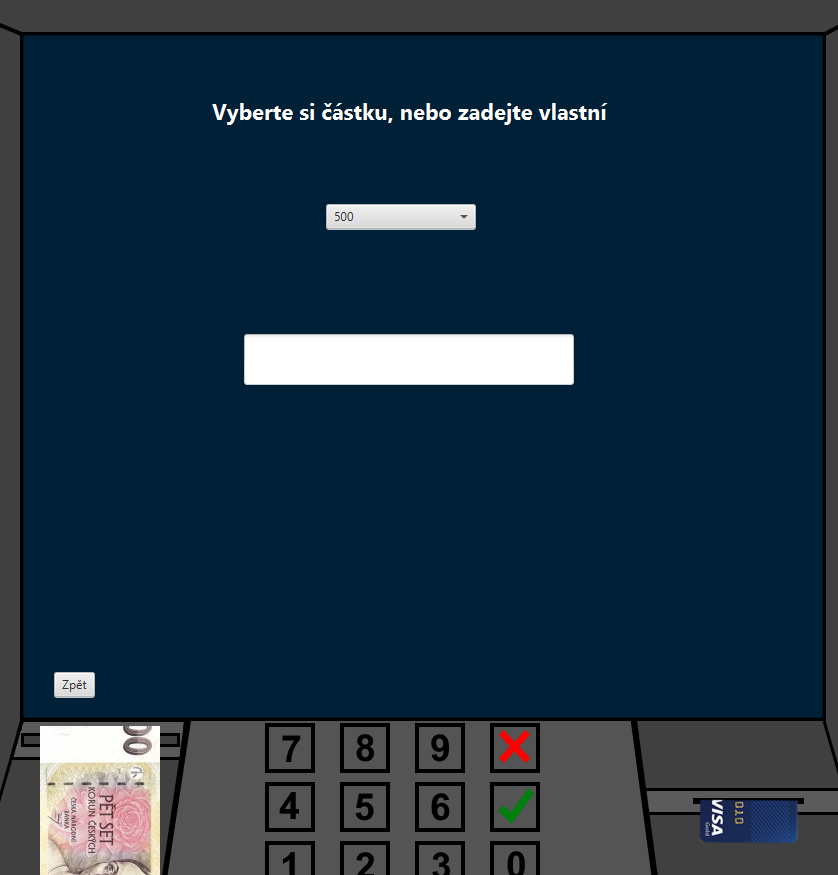
Zde dochází pouze k zavolání setPassword metody ze třídy Password, která se o vše postará. Zadá-li uživatel svoje heslo špatně, vypíše se mu chyba do Labelu.

#### SendMoneyController

V tomto controlleru uživatel zadá číslo účtu včetně čísla banky a částky. Znovu dojde ke zkontrolování hodnoty účtu a následnému odtržení částky a zapsání do dates ArrayListu.

#### CashInController

V tomto controlleru se ihned při načtění pomocí Initializable načtou obrázky bankovek jakožto ImageView. Po kliknutí na nějakou bankovku se uživateli přidá na účet pomocí metody v Account třídě a je zároveň uživateli potvrzeno že se skutečně bankovka vložila.

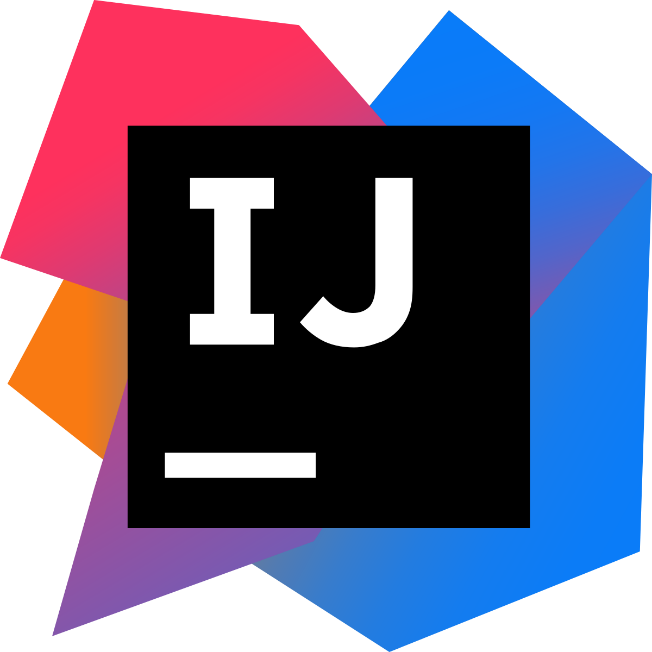
 Obrázek 3 : výběr 500 z bankomatu

# Závěr

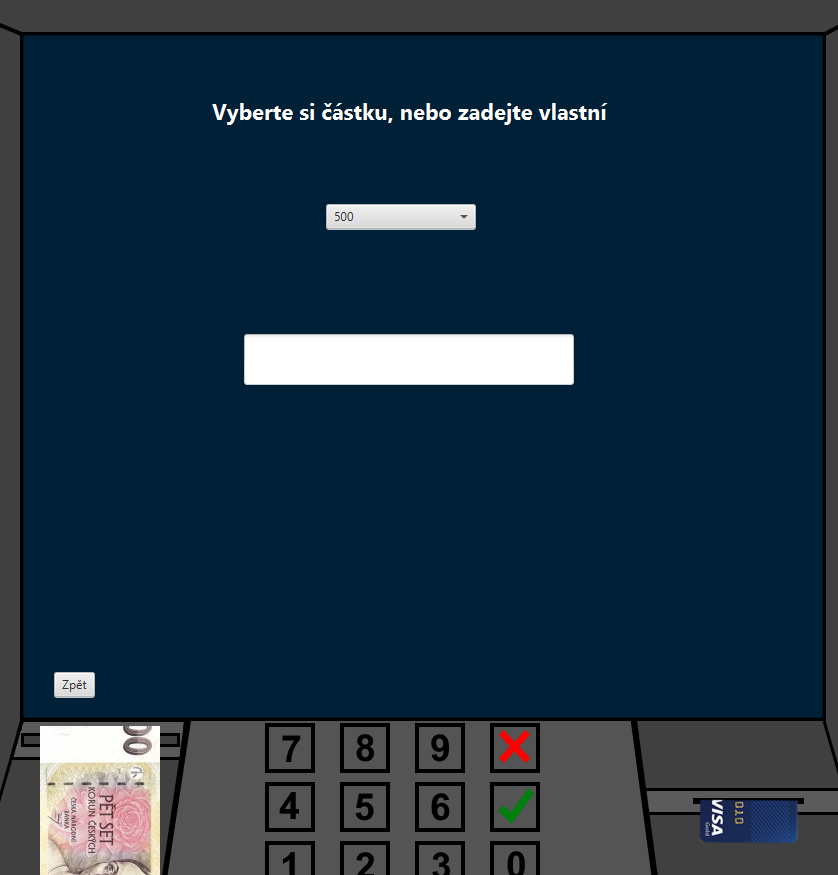
Cílem bylo vytvořit okenní aplikaci v Jave simulátor bankomatu. Myslím si že se mi podařilo tento úkol poměrně slušně a správnými metodami splnit. Vytváření takového projektu mi kromě procvičení základní Javy přineslo i potvrzení pocitu, že mě programování opravdu baví a že bych se tím rád v budoucnu živil.

# Přílohy

## Obrázky

[Obrázek 1 : logo IntelliJ [https://commons.wikimedia.org/wiki/File:IntelliJ\_IDEA\_Icon.svg] 6](#_Toc102341437)

[Obrázek 2 : obrázek pozadí banomatu  7](#_Toc102341438)

[ Obrázek 3 : výběr 500 z bankomatu 13](#_Toc102341439)

## Zdroje

<https://www.baeldung.com/java-password-hashing>

<https://en.wikipedia.org/wiki/Hash_function>

https://cs.wikipedia.org/wiki/JavaFX