

Design Document

Project ATM Interface

Table of Contents

Design Document Project ATM Interface.....	1
Project Description.....	1
Application Introduction.....	2
Main Menu.....	2
Sign Up.....	2
Confirmation of entered name.....	3
Username and password.....	4
Sign In.....	4
Customer Menu.....	5
View Balance.....	5
Withdraw.....	5
Deposit.....	6
Transfer.....	7
Change Credentials.....	8
Sign Out.....	8
Program Persistence.....	8
Technologies used.....	8
Additional documentation.....	8

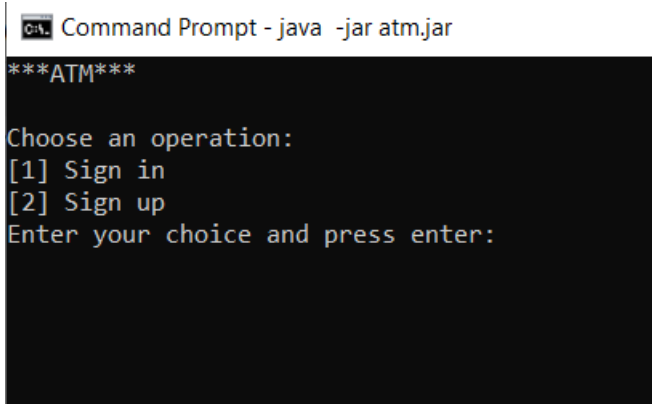
Project Description

This is a command line application that simulates an ATM interface. The application is intended to be used by a bank customer. The customer can sign up for a bank account by entering their first name and last name and by choosing a username and a password. The authenticated bank customer can then choose to view the balance of their bank account or to deposit, withdraw or transfer money.

Application Introduction

Main Menu

When the application runs, the first menu that is shown is the main menu. See picture below.



```
C:\> Command Prompt - java -jar atm.jar
***ATM***

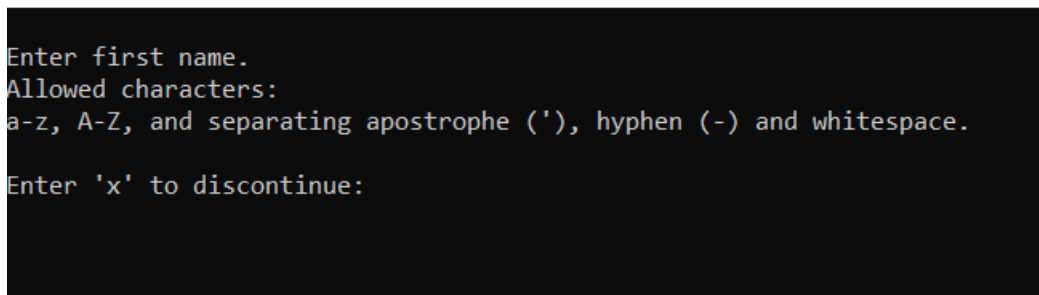
Choose an operation:
[1] Sign in
[2] Sign up
Enter your choice and press enter:
```

From this menu the customer can choose to either sign up for a bank account or to sign in if they already have an account.

Sign Up

When the customer chooses to sign up he or she is prompted to enter their first name and last name. When entering names, the app checks that **only valid letters and a specific set of valid characters** is entered, to avoid that an incorrect name is entered. If the customer enters invalid characters, the input is discarded and the customer is prompted to try again.

Names can contain hyphens and apostrophes, for example Bella-Rose O'Neil, or they can consist of several **white space separated names**, for example Muhammed Ali Pettersson Tahar. Therefore, hyphen, apostrophe and whitespace are allowed in names. However, they are **not allowed in the beginning of a name** and for the sake of user-friendliness the app checks that the customer doesn't enter these as the first character in a name. For the sake of additional user-friendliness and accuracy the program also removes any extra whitespace that the customer might mistakenly enter. The **customer can also choose the enter 'X' if they change their mind**. See screenshots below.



```
Enter first name.
Allowed characters:
a-z, A-Z, and separating apostrophe ('), hyphen (-) and whitespace.

Enter 'x' to discontinue:
```

```
Enter last name.  
Allowed characters:  
a-z, A-Z, and separating apostrophe ('), hyphen (-) and whitespace.  
  
Enter 'x' to discontinue:  
Borg
```

The app **accepts only letters A-Z** in names because the Windows command prompt doesn't handle presenting special Unicode characters without problems. It would have been better for user-friendliness if the app could accept all characters. For such an implementation, the following code could have been used to validate letters instead of the current one:

```
public static boolean validateLetters(String input) {  
  
    // Validate input using \p{L} which is a Unicode Character Property  
    // that matches any kind of letter from any language.  
  
    String allowedCharacters = "^[\p{L}\s'-]+$";  
  
    Pattern pattern = Pattern.compile(allowedCharacters, Pattern.CASE_INSENSITIVE);  
    Matcher matcher = pattern.matcher(input);  
    return matcher.find();  
}
```

This method works without problems in the IDE console.

Confirmation of entered name

Once the name has been entered, the app checks with the customer that the name is correct, because the customer won't be able to edit it later.

```
You entered the following name: Calle Palm  
  
You won't be able to edit this name after signing up.  
Is this name correct?  
  
Type 'Y' for yes or 'N' for no:
```

Username and password

Next the customer needs to choose a username and a password. Here the customer can enter any characters. Whitespace is not allowed for the sake of user-friendliness so that any confusion caused by mistakenly entered whitespace can be prevented. In this way the customer can't for example by accident enter a leading whitespace before the password. And the system can't for example have a customer with the username 'Calle' and another one with the username 'Calle ' (with trailing whitespace). The app also requires that the password has a reasonable length of 6-20 characters.

When entering the username the app checks that the username isn't already taken and prompts the customer to choose another one if it is so.

```
Choose a username.  
Required length 6 - 20 characters. White space not allowed.  
Enter 'x' to discontinue:
```

```
Choose a password.  
Required length 6 - 20 characters. White space not allowed.  
Enter 'x' to discontinue:
```

The password is saved using the SHA-256 hashing algorithm. In this way original passwords are not saved, but only their hashed versions.

Sign In

Once the customer has signed up he or she can sign in with their username and password. When signing in the app first checks that the username exists in the database. Next the app prompts the user for a password and hashes the entered password. For this, hashing salt is retrieved from the customer data. The customer's hashed saved password and entered password are compared and if they match the customer is let into the system.

Customer Menu

Once the customer has been authenticated through signing in, the app presents them with the customer menu seen below.

```
***Customer operations***  
  
Choose an operation:  
[1] View balance  
[2] Withdraw money  
[3] Deposit money  
[4] Transfer money  
[5] Change my customer credentials  
[6] Sign out  
Enter your choice and press enter:
```

View Balance

From here the customer can choose to view their account balance (option 1). This takes the customer to the following view:

```
***View Balance***  
  
Account owner: Calle Carlsson  
Account number: 8888-1  
Current balance: 0,00 kr  
  
Press enter to continue...
```

Here the balance is represented using Swedish currency formatting.

Withdraw

Option 2 instead takes the customer to the withdraw page where he or she can enter an amount of 1-20 000 kr. For the sake of user-friendliness the customer can use either comma or dot for decimal input. Only two decimals are allowed because that is the usual precision standard applied to representing currency values.

```
***Withdraw***  
  
Enter amount in digits.  
Allowed amount: 1 - 20000 kr.  
  
Use comma (,) or dot (.) for decimal numbers.  
  
Enter 'x' to discontinue:
```

If the customer tries to withdraw an amount that exceeds the account's balance the app tells the customer that the balance is insufficient. See below.

```
Insufficient balance. See current balance below.  
  
***View Balance***  
  
Account owner: Calle Carlsson  
Account number: 8888-1  
Current balance: 0,00 kr  
  
Press enter to continue...
```

The app also checks that the amount entered isn't negative and that it doesn't start with a leading zero as this could cause problems in the system.

Deposit

Option 3 from the customer menu takes the customer to the deposit page:

```
***Deposit***  
  
Enter amount in digits.  
Allowed amount: 1 - 20000 kr.  
  
Use comma (,) or dot (.) for decimal numbers.  
  
Enter 'x' to discontinue:
```

Input is validated again.

```
Deposit succeeded. See current balance below.
```

```
***View Balance***
```

```
Account owner: Calle Carlsson
```

```
Account number: 8888-1
```

```
Current balance: 550,50 kr
```

```
Press enter to continue...
```

View after a successful deposit.

Transfer

Option 4 in Customer Menu leads to the Transfer -page, where the customer is prompted to enter the recipient's account number:

```
***Transfer***
```

```
Enter the recipient's account number.  
Digits and hyphen allowed.
```

```
Enter 'x' to discontinue:
```

The entered account number is validated by checking whether it exists in the account database or not.

The app also checks whether the sending and recipient account are the same and prompts the customer if it is so.

Also if the account number is invalid, customer is prompted to try again.

A successful transfer is displayed to the customer as seen below.

```
Successfully transferred:
```

```
2 500,00 kr
```

```
Recipient account:
```

```
8888-1
```

```
See current balance below.
```

```
***View Balance***
```

```
Account owner: Pelle Herrman
```

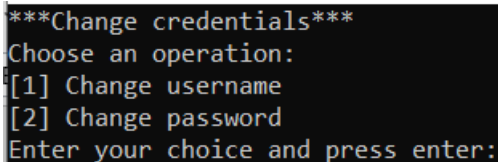
```
Account number: 8888-2
```

```
Current balance: 9 500,70 kr
```

```
Press enter to continue...
```

Change Credentials

Option 5 in Customer Menu takes the customer to the Change Credentials -page where the customer can change username or password. These are validated in the same manner as they are when signing up.



```
***Change credentials***  
Choose an operation:  
[1] Change username  
[2] Change password  
Enter your choice and press enter:
```

Sign Out

Option 6 (Sing out) leads back to the Main Menu.

Program Persistence

There is no option to quit the program because the ATM is intended to run continuously. The customer details are saved only within the program in the classes called 'CustomerDatabase' and 'AccountDatabase' that simulate real databases. The customer details are thus 'alive' only as long as the program runs as they are not saved externally.

Technologies used

The program has been developed using IntelliJ IDEA and Java SDK 17.0.2 (build 17.0.2 + 8) on a Windows operating system. The program has been tested only in Windows command prompt. Initial unit testing with JUnit has been initiated but due to the lack of time, substantial testing hasn't been carried out.

Additional documentation

The project's product backlog and UML diagrams can be found in the Project Management - directory of this project's GitHub repository. These document's can further clarify the technical implementation of this app.