Lab 4: Presto Simulator (Arrays)

Object Oriented Programming 1 – Fred Stiebler

Program Description

In this lab you will write a Java console app that will **simulate the Presto card system.** Our Presto system will allow the user to **add** and **remove** cards from an **ArrayList of Presto cards**. **Each card has its own name and balance**. The user can **tap** and **top-up each card individually**. **Fare is fixed to $2.50**.

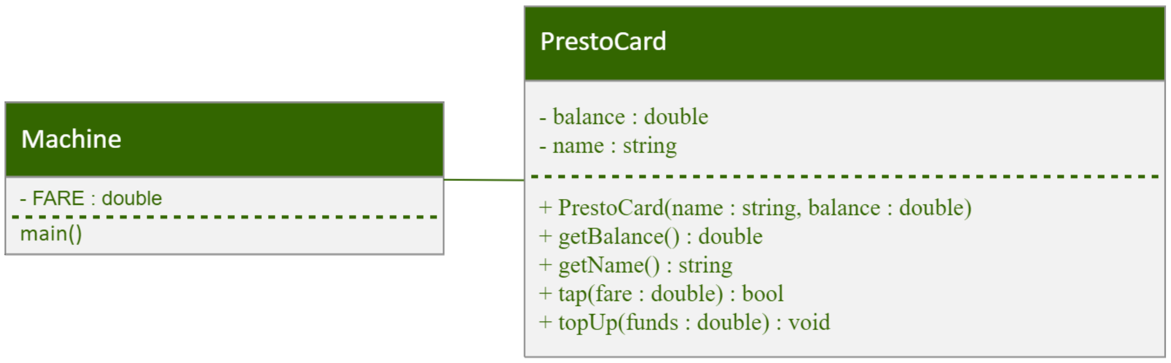
Presto Card

If you are not familiar with Presto card, you can learn more about it at: prestocard.ca

Note that **we are not simulating all available Presto card features, nor the actual prices!** These vary according to the transit commission (GO, VIVA, TTC, DRT, YRT, etc) and vehicle type that you use in your commute. **The only required features are described below in the required classes and methods section.**

Required Classes and Methods

You **must** implement the **classes** and **methods** described below:



Lab 4: Presto Simulator (Arrays)

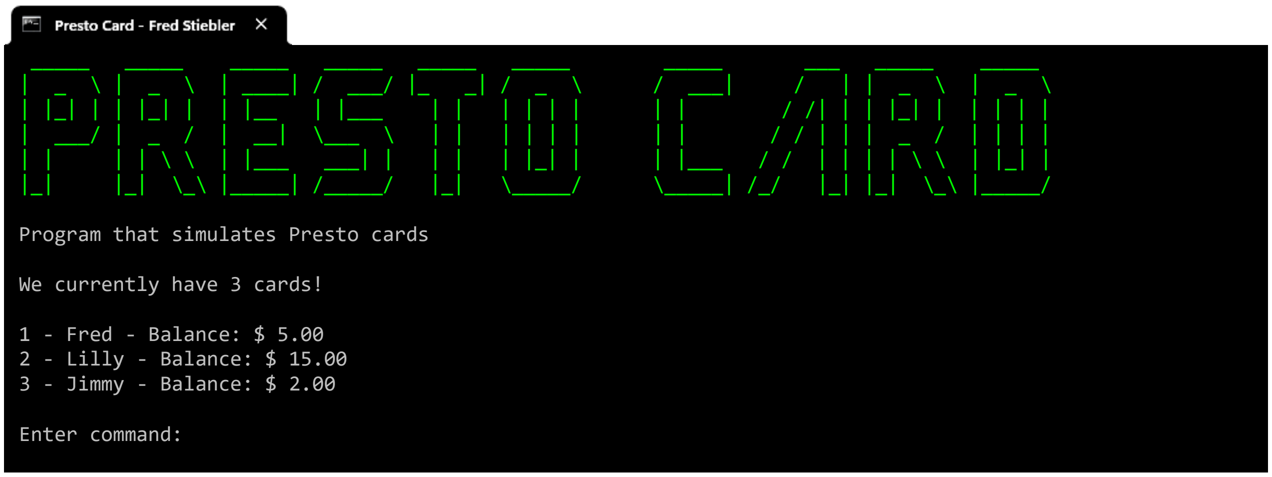
Object Oriented Programming 1 – Fred Stiebler

Simulation Screen

1. Clear the console window
2. Print the banner and description
3. Print **how many elements** (Presto cards) are in the ArrayList
4. Print the contents of the ArrayList
    - Print each card's **number**, **name**, and its **balance**
5. Prompt for a command, the available commands are:
    - **quit** Example: `quit`
        - Closes the app. **DO NOT FORCE CLOSE**. Use a Boolean to stop the loop!
    - **add** CardName InitialBalance
        - Example: `add Fred 20.00`
        - **Add** a new Presto card **object** at the end of the ArrayList
    - **del** CardIndex
        - Example: `del 2`
        - **Delete** an **existing** card from the ArrayList
    - **tap** CardIndex
        - Example: `tap 1`
        - **Tap** an existing card. I.e. **Take money from a card's balance**.
        - **Deducts a fixed fare (constant) of $2.50 from the card's balance**
        - ⚠ Make sure to **check if enough funds are available!**
        - `tap()` **returns** `true` if the card has enough cash to tap, otherwise, **returns** `false`
    - **topup** CardIndex Amount
        - `topup 2 20.50`
        - **Add funds** to the **balance** of an existing card. I.e. **Adds money to a card's balance**.
        - Example below adds $20.50 to **card** number **2**
6. **Print the appropriate error messages for invalid user input**
7. Loop back to step 1

Example simulation screen

Try to match this as close as possible (stylised banner and colours are **optional**):



```
Program that simulates Presto cards

We currently have 3 cards!

1 - Fred - Balance: $ 5.00
2 - Lilly - Balance: $ 15.00
3 - Jimmy - Balance: $ 2.00

Enter command:
```

⭐ Bonus Marks Opportunities (choose one, up to 5%)

Show the added functionality in the report (and mention which one you chose) to make sure I see it!
You can do both if you want, but the bonus will be capped to 5%.

If you choose option1:

Add the ability to **del, tap,** and **topup** by typing the **CardName instead of index.**

Examples:

```
topup Fred 20.00
tap Fred
del Fred
```

Write the `static` method `searchName(cards:ArrayList, name:string):int` in the `PrestoCard` class. This method will **iterate through all the elements using a for loop** to **compare** if the **name you are searching for** is **equals** to the **name at the current index**. **Return the card's index** if their **names are equal**, or **-1 if card is not found** with that name. If an index is returned, then you can delete the card in that index.

If you choose option 2:

Add the **history** command that prints the **entire balance history** for a given card.
You can choose to make the **history** command accept either the **card's index** or **CardName** (or both!).

Examples:

```
history 1
```

```
history Fred
```

```
Fred's Card History:

tap   - new balance: $2.00
topup - new balance: $12.00
tap   - new balance: $9.50

Press [enter] to return to Main Menu:
```

Write the `public` method `printHistory():void` in the `PrestoCard` class. This method will **print the balance history** of a `PrestoCard` **object**.

Style Guide and Documentation

To be eligible for full marks on this or any lab in this course your application must conform to the requirements as outlined above and the course Style Guide, in this case making sure to include:

- Your code follows our Java style guide.
- **Appropriately declared data types for all possible variables and constants.**
- Appropriate and complete **program documentation** (code comments).