CS 251
Project 2
Payment Verification

Due 09-25-22

Joseph Haugh

Project Description

You are tasked with creating a command line program that can verify two types of payments, credit card and cash. For credit cards you will be using Luhn's algorithm to verify the check digit for a given card number. For cash you will be verifying that the given serial number is in the following range [1_000_000, 10_000_000].

This project emphasizes the following concepts:

- Exceptions
- Inheritance

Project Specification

Your program should read in input from the command line until given the string "EXIT". Each line of input will have the following structure:

PAYMENTTYPE PAYMENTNUMBER PAYMENTAMOUNT\n

PAYMENTTYPE: "CREDITCARD" | "CASH"

PAYMENTNUMBER: Integer

PAYMENTAMOUNT: Double

You must handle the following exceptions:

- NumberFormatException: print out "$PAYMENTNUMBER is not a valid card/serial number."
- IllegalArgumentException: print out the message within the exception. See below for details.
- InvalidAmountException: print out "$PAYMENTAMOUNT is not a valid amount."

Credit Card Class

Your constructor should check for the following invariants in the specified order:

First you must verify that the given int[] only contains ints in the following range: [0, 9]. If this does not hold then you need to throw an IllegalArgumentException with the following message:

"The card number must consist of numbers in the following range: [0,9]"

Then check that the given credit card number has exactly 6 digits. If it does not, you need to throw an IllegalArgumentException with the following message:

"The card number must be exactly 6 digits"

Lastly you need to check the check digit. The last digit is the check digit. This digit verifies that the other 5 digits were entered correctly. This digit is computed using Luhn's algorithm. You must implement this algorithm in order to verify if the given credit card number has a valid check digit. If the check digit is invalid then you must throw an IllegalArgumentException with the toString value of the PaymentVerification enum.

The credit card class must verify these properties in the constructor. Next you must override the equals and toString methods. The equals method should return true if the card numbers have the same contents in the same order. The toString method should print out the following:

Valid Credit Card Number: $CARDNUMBER, amount: $amount

Where $CARDNUMBER is the card number printed out. Where $amount is the field in the Payment abstract class. For example if the credit card number was 123455 and the amount was 40.25 then the print out would be:

Valid Credit Card Number: 123455, amount: 40.25

Cash Class

Cash serial numbers must in the following range [1_000_000, 10_000_000] The Cash class must verify this property in the constructor and throw an IllegalArgumentException with the toString value of the PaymentVerification enum if it does not hold. Next you must override the equals and toString methods. The equals method should return true if the serial numbers are equal. The toString method should print out the following:

Valid Cash Serial Number: $SERIALNUMBER, amount: $amount

Where $SERIALNUMBER is the serial number on the cash. Where $amount is the field in the Payment abstract class. For example if the serial number is 1000123 and the amount was 40.25 then the print out would be:

Valid Cash Serial Number: 1000123, amount: 40.25

Project Starter

I have uploaded a project starter alongside this PDF on canvas. You **cannot** change any type signatures! That means any function that throws an exception must remain that way. Do not listen to Intellij if it tells you it doesn't throw an exception that is only because you have implemented it yet. You must complete all the TODO:'s in order to complete the assignment.

Submitting Your Project

You must submit a zip file on canvas of your src folder. This means it should only include .java source files.