Western

CS1027 - Computer Science Fundamentals II
Summer 2019

Assignment 2

Learning Outcomes

By completing this assignment, you will gain skills in:

- Using inheritance in Java,
- Solving problems through the use of stacks, and
- Designing algorithms in pseudocode and implementing them in Java.

Part I

A local community center is a public location where members of a community tend to gather for group activities, social support, public information, and other purposes. In this assignment, consider the task of developing a java program that keep tracking of the sales of tickets to a community center event. Each ticket has a unique number and a price. There are three types of tickets: walk-in, regular, and student tickets.

- Walk-in tickets price is $40, if it purchased in the same day of the event.
- Regular tickets purchased 10 or more days before the event cost $20, and 30$ if purchased fewer than 10 days before the event.
- Student tickets are always sold at 50% of the Regular tickets price (i.e. when Student tickets are purchased 10 or more days early, they are reduced to $10 per ticket, otherwise it costs $15 per ticket).

You are required to implement the following four classes.

The Ticket class

- Implement a class called Ticket that serves as the superclass for all three types of tickets. Define all common operations (getters and setters) in this class and specify all differing operations in such a way that every subclass may implement them (if any).
- No actual objects of type Ticket will be created: Each actual ticket will be an object of a subclass type.
- Define the following methods:
    - A constructor method that constructs a ticket by number.
    - A method that returns the ticket's price.
    - A method that prints a ticket object as a String. An example String would be "Number: 17, Price: $40.0".

The WalkinTicket class

- Implement a class called WalkinTicket to represent a walk-in event ticket.
- Walk-in tickets are also constructed by number, and they have a price of $40.

The RegularTicket class

- Implement a class called RegularTicket to represent regular tickets.
- A regular ticket is constructed with a ticket number and with the number of days in advance that the ticket was purchased.
- Hint: Regular tickets purchased 10 or more days before the event cost $20, otherwise it costs $30.

The StudentTicket class

- Implement a class called StudentTicket to represent tickets purchased by students.
- A Student ticket is constructed with a ticket number and with the number of days in advance that the ticket was purchased.
- Hint: Student tickets purchased 10 or more days before the event cost $10, and Student tickets purchased fewer than 10 days before the event cost $15 (half a price or a regular ticket).
- When a Student ticket is printed, the String should mention that the student must show his or her student ID (for example, "Number: 18, Price: 15.0 (ID required)").

The SalesReport Class

The SalesReport class stores the total volume of sales, the highest figure, the average sales, a list of tickets sold and the number of sold tickets for each type;
Further, the SalesReport Class should have the following public methods:

- A constructor method, SalesReport that creates an instance of the class (an object).
- An accessor method, getData that retrieves the sales data from a given file (sales.txt).
- A method, computeStats that computes the sales statistics (sample output is given below), and
- A method, displayResults that prints out the sales report.

The data for this class is stored in the file sales.txt.

The MyTickets.java program

Illustrate your implementation with a class called MyTickets that have a static method called ***main*** (Hint: the method ***main*** is used to run all the code)

The main function is given below:

```
public static void main (String [] args)
{ //Load sales data
 InStringFile salesReader = new InStringFile("src/sales.txt");
 // create sales report
 SalesReport ticketsSales = new SalesReport();
 ticketsSales.getData(salesReader);
 // sales analytics computations
 ticketsSales.computeStats();
 //Display the results
 ticketsSales.displayResults(); }
```

The MyTickets class will be used by the local community center stuff to generate sales report for tickets sold.

The following is a sample output screen for the program:

```
Sales status
Walk-in tickets' sales: 16
Regular tickets' sales: 61
Student tickets' sales: 59

Sales analytics
Total number of tickets sold: 121 tickets
Sales volume: $2185
Average sales is $18
largest sale (Regular): 61 tickets
```

Figure 1: Sample sales report

Further, the program should be able to print out a list of tickets sold. In practice, the list might have a single transaction like "Number: xxxxxx, Price: xx" for regular and walk-in tickets or like “Number: xxxxxx, Price: xx (ID required)" for student tickets.

The InStringFile.java program

As in assignment 1, you are supplied with InStringFile.java, which contains Java code to parse a file, reading one "token" at a time. The sales.txt file has three tokens per line. The first token is the ticket number, then the ticket type and the last token shows the # of days a ticket purchased. Main.java uses InStringFile class when parsing the input file.

Part II

John and his little brother Dave were attending one of the local community events. And, they were offered a plastic container of candies which can hold a maximum of ten candies at a time. The candies are stacked one over the other and Dave could only access the candies from its container’s top side. For this assignment, we are assuming that there are five colored candies with five different flavors, yellow (banana), purple (blackberry), red (strawberry), green (green apple) and pink (watermelon). Dave only likes the banana flavor, so he carefully takes out all the candies with banana flavor, one by one, eats them ones, and keeps the others in order minus the banana flavored candies of course. You can assume that the candies have the following initial order inside the container: red, purple, green, pink, yellow, pink, red, yellow, green, yellow.

- Write an algorithm in a pseudocode to simulate the process described above and to report back on how many banana candies Dave ate and how many candies are left in the plastic container. You may use any of the stack operations defined in the Stack ADT but may not assume any knowledge on how the stack is implemented (i.e. array, linked list, etc…).

- Implement your algorithm using the LLStack class that we discussed in the class. Your program must generate ten candies, show the initial content of the candy container, remove the banana's candies and then show the content of the container again.
- Illustrate your algorithm with a sample run to demonstrate the expected behavior of your implementation.

Non-functional Specifications

- All implemented programs have to perform exception handling if necessary.
- You are not allowed to use Java pre-build library for stack or queue in your code for this assignment.
- Include a comment identifying yourself (name and student number, uwo email address) and the assignment number, course and year at the beginning of all your class files. For example, the professor has:

```
//////////////////////////////////////////////////////////////////
// Ahmed Ibrahim 123456789                                      //
// ahmed@csd.uwo.ca or ahmed@uwo.ca                             //
// Assignment 2, CS1027, Spring 2019                            //
//////////////////////////////////////////////////////////////////
```

- Include comments describing key parts of your program well. Comments should be grammatically correct, concise, easy to understand and match the Javadoc syntax.
- Assignments are to be done **individually** and **must be your own work**. Group work is not tolerated. A software tool may be used to detect plagiarism.
- Use Java coding conventions and good programming techniques. For example:
    - Use meaningful variable names
    - Use Java conventions for naming variables and constants.
- Make sure your code runs using Eclipse's Java, even if you do not use Eclipse to write your code.
- **Zip** all the files used by your program **(project folder)**. Use your first and last names and student number and assignment number and year (with linking of the tokens in the zip filename done by underscores). For example, your professor's files would be in

  Ahmed_Ibrahim_123456789_Assignment_2_Spring_2019.zip.
- Assignment 2 due on Friday, July 12, at 5:00 pm. You need to submit all your .java files as a single zip file. And, it's your responsibility that the Zip file is not empty or damaged.
- You can submit your assignment more than once and up to three times through OWL. If you submit the assignment more than once, we assume that the latest submission is the one you want us to mark.
- It is your responsibility to submit your assignment **zip** file before the due date and in the dedicated drop-box for the assignment.
- No submission allowed through emails.

Good Luck