Homework 2

Due Sep 23, 2020

15 pts each except Problem 7.

What should be turned in?

* This word doc
* Homework2 project in zip

# Problem 1: Flower Bouquet

Let us say you run a flower shop close to TCU. In your store, there are only two kinds of flowers: rose bouquet ($20) and daisy bouquet ($15). Every bouquet has a description and a price (see code in IntelliJ). There are also decorations for extra charge: paper wrap, card, balloon and. E.g. A rose bouquet with one paper wrap and a card, a rose bouquet with double paper wraps and triple glitters, or a daisy bouquet with no decoration at all. Decoration has a price and description too. E.g. paper wrap is $5 each, card is $6 each, balloon is &7 each and glitters is $8 each.

Please write code to implement the above description in Java.

In your test case, please create

1. a rose bouquet with a paper wrapping and a card,
2. a rose bouquet with double paper wrappings and a glitter
3. a daisy bouquet with no decoration at all.

Once they are created, calculate and print their total prices (don’t write any calculation logic in TestFlowerBouquet class).

Your printout must look like this:

|  |
| --- |
| Rose Bouquet Paper wrap ... Card ... $31.0  -----------------------------------  Rose Bouquet Paper wrap ... Paper wrap ... Glitter ... $38.0  -----------------------------------  Daisy Bouquet $15.0 |

After running your business for a while, you decide to add lily bouquet and ornamental leaves decoration. Please modify your code accommodate the change.

# Problem 2: SQL Statements

You are working in a team that delivers a Java framework to work with relational databases, e.g. MySQL. You are assigned the task to model SQL select statement. Since everything is an object, you create a class and name it SelectStatement.

Let’s briefly review the components of a SQL select statement:

|  |
| --- |
| SELECT *content*  FROM *content*  [WHERE …] *content*  [GROUP BY …] *content*  [HAVING…] *content*  [ORDER BY …] *content*; |

*Everything in [] is optional for a select statement.*

Since you need to model this SQL select, in your SelectStatement class, please define private fields for SELECT, FROM, WHERE, GROUP BY, HAVING, and ORDER BY. Their types are all String, so you can store the content in the corresponding field. You may also want to define some constructors for statement instance creation.

Next, let’s create some SelectStatement instances using constructors.

|  |
| --- |
| SELECT pnumber, pname, count(\*)  FROM Project join Works\_on on pnumber=pno  GROUP BY pnumber, pname  HAVING count(\*) > 2; |

Copy your code for creating the above SQL statement instance here:

|  |
| --- |
| SelectStatement select1 = new SelectStatement("pnumber, pname, count(\*)", "Project join Works\_on on pnumber=pno", "pnumber, pname", "count(\*) > 2;"); |

|  |
| --- |
| SELECT fname, lname, address  FROM Employee join Department on dnumber=dno  WHERE dname='research'; |

Copy your code for creating the above SQL statement instance here:

|  |
| --- |
| SelectStatement select2 = new SelectStatement("fname, lname, address", "Employee join Department on dnumber=dno", "dname='research';"); |

|  |
| --- |
| SELECT fname, lname, address  FROM Employee; |

Copy your code for creating the above SQL statement instance here:

|  |
| --- |
| SelectStatement select3 = new SelectStatement("fname, lname, address", "Employee;"); |

|  |
| --- |
| SELECT “ FROM Project join Works\_on on pnumber=pno  WHERE budget > 200000 GROUP BY pnumber, pname, budget HAVING count(\*) > 2  ORDER BY pnumber; |

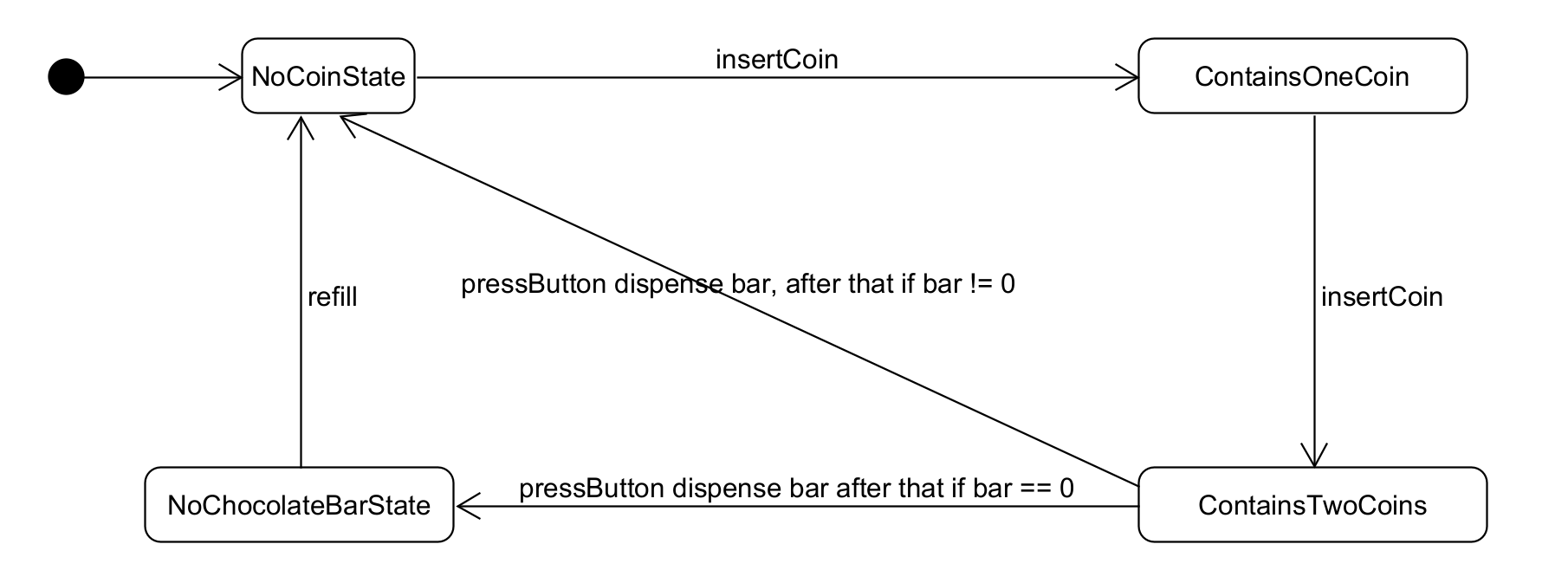
Copy your code for creating the above SQL statement instance here:

|  |
| --- |
| SelectStatement select4 = new SelectStatement("pnumber, pname, budget, count(\*)", "Project join Works\_on on pnumber=pno ", "budget > 200000", "pnumber, pname, budget", "count(\*) > 2", "pnumber;"); |

# Problem 3: Vending Machine

You are responsible for designing the behavior of a vending machine. When a customer inserts two coins into the machine and presses “Buy” button, the vending machine is to roll out a chocolate bar. Note, coins need to be inserted one by one. It is not possible to insert 2 at the same time.

The machine will then have four states: “No Coin”, “Contains One Coin”, “Contains Two Coins”, and “No Chocolate Bar”. If the bars are out, vending machine goes to “No Chocolate Bar” state. After refill, vending machine goes from “No Chocolate Bar” back to “No Coin”. These states represent different behaviors of the machine. State transitions will move the machine from one state to another. As an example, if the current state of the machine is “No Coin”, and then a customer enters a coin, a state transition will move the machine to the “Contains One Coin” state. Take a look at the state diagram.



The dot on the left means the initial state, it automatically transitions to NoCoinState. The sentence on the arrow means event, if insertCoin event occurs while the machine is at NoCoinState, the machine transitions to ContainsOneCoin state. In this example, a method call to the insertCoin() method of the vending machine can be viewed as the insertCoin event.

Please implement the behavior of this vending machine in Java.

Once you are done, please copy the following into the main method of TestVendingMachine class. Assume there are 2 bars in the vending machine initially.

|  |
| --- |
| VendingMachine vendingMachine = new VendingMachine(2);  vendingMachine.insertCoin();  vendingMachine.pressButton();  vendingMachine.insertCoin();  vvendingMachine.pressButton();  vvendingMachine.insertCoin();  vendingMachine.insertCoin();  vendingMachine.pressButton();  vendingMachine.insertCoin();  vendingMachine.insertCoin();  vendingMachine.pressButton();  vendingMachine.refill(5);  vendingMachine.insertCoin();  vendingMachine.insertCoin();  vendingMachine.pressButton(); |

When you launch the program, you should get:

|  |
| --- |
| One coin inserted...  Please insert one more coin...  Two coins inserted...please press Buy button...  Buy button pressed...start dispensing  One coin inserted...  Two coins inserted...please press Buy button...  Buy button pressed...start dispensing  Sorry, we are out...  Sorry, we are out...  Sorry, we are out...  One coin inserted...  Two coins inserted...please press Buy button...  Buy button pressed...start dispensing |

# Problem 4: Online auction

You are a developer of an online auction software. For online auction, there are many products waiting to be bid on, there are also many users (bidders) ready to bid on products. It is very obvious we need two classes.

* Every product has a name and initial bidding price.
* Every bidder has a name.

First, please create one product and three bidders in TestAuction file.

|  |
| --- |
| new Product("iPhone 12",new BigDecimal(999));  new Bidder("Sanchez");  new Bidder("Wei");  new Bidder("Scherger"); |

A bidder would opt to subscribe a product, meaning, once there is a higher bid, this user will receive a message (Email or SMS message, but here, we simply print a message to the console.)

Now write code to make all three bidders subscribe this iPhone 12.

Next in your program, please simulate Sanchez bidding $1200 and Scherger, bidding $1250.

Your program MUST print something like this:

|  |
| --- |
| -----------------New bid placed----------------  Hello Sanchez! New bid of amount 1200 has been placed on iPhone 12 by you  Hello Wei! New bid of amount 1200 has been placed on iPhone 12 by Sanchez  Hello Scherger! New bid of amount 1200 has been placed on iPhone 12 by Sanchez  -----------------New bid placed----------------  Hello Sanchez! New bid of amount 1250 has been placed on iPhone 12 by Scherger  Hello Wei! New bid of amount 1250 has been placed on iPhone 12 by Scherger  Hello Scherger! New bid of amount 1250 has been placed on iPhone 12 by you |

As you can see, when Sanchez made a bid, Sanchez receives a message, “Hello Sanchez! New bid of amount 1200 has been placed on iPhone 12 by you”, Wei receives, “Hello Wei! New bid of amount 1200 has been placed on iPhone 12 by Sanchez” and Scherger receives “Hello Scherger! New bid of amount 1200 has been placed on iPhone 12 by Sanchez”

When Scherger made a bid, Sanchez receives a message, “Hello Sanchez! New bid of amount 1250 has been placed on iPhone 12 by Scherger”, Wei receives, “Hello Wei! New bid of amount 1250 has been placed on iPhone 12 by Scherger” and Scherger receives “Hello Scherger! New bid of amount 1250 has been placed on iPhone 12 by you”

# Problem 5: Promotion Strategies

Assume you are a developer at Amazon. Holiday sales is around the corner, and you are asked to implement several mutually exclusive strategies for sales promotion.

* The first strategy is price deal, 10% off to all products.
* The second strategy is to refund $100 if the total spending is greater than or equal to $500.
* The third strategy is to give away an Amazon Echo Dot if the total cost is greater than or equal to $300.

The marketing team will active one of the above strategies on different dates in the holiday season based on feedback of the sales.

In TestShoppingCart class, I have already defined a shopping cart with three items in it.

When a user checks out their shopping cart, the active promotion strategy will be applied to the products in the cart. Please implement the three strategies.

When you are done, in TestShoppingCart class, please turn on the first strategy and checkout the cart. Your program must print:

|  |
| --- |
| $981.0 |

Turn on the second strategy and checkout the same cart. Your program must print:

|  |
| --- |
| $990.0 |

Turn on the second strategy and checkout the same cart. Your program must print:

|  |
| --- |
| Add a free Amazon Echo Dot in shopping cart  $1090.0 |

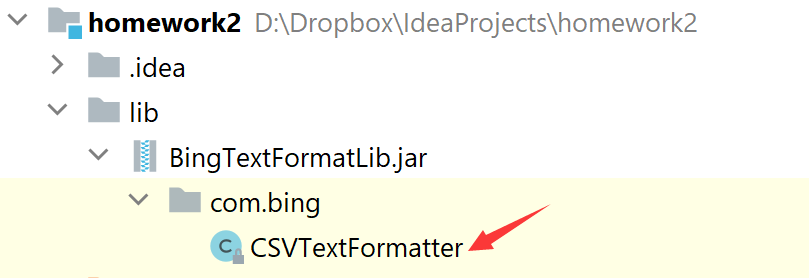
Now think about this, what if the marketing team is asking you to implement a fourth strategy?

How would you modify your code? (Just write your thoughts here, no need to write code)

|  |
| --- |
|  |

# Problem 6: Text Formatter

Assume you are developing a text formatting software. Your team has already finished a NewLineFormatter class, which implements the Formatter interface. NewLineFormatter can replace periods (.) with newline (\n) characters. In the next iteration of the software development, you plan to work on CommaFormatter, that is, replace periods (.) with commas (,). However, you are lucky to find an external vendor who provides this feature in CSVTextFormatter (The JAR is under directory “lib”).



However, this CSVTextFormatter class doesn’t implement the interface your team designed previously. Since you practice “Program to an interface,” all the existing code in TestFormatter is programed to Formatter interface. How can use make use of this external class?

When you finish this program, please format the same sentence using two formatters in TestFormatter class.

The given string is “Bing.is.a.good.man.”

|  |
| --- |
| // using NewLineFormatter  Bing  is  a  good  man  // using vendor’s comma formatter  Bing,is,a,good,man, |

Warning!

In TestFormatter class, you must program to Formatter interface.

DO NOT write this:

|  |
| --- |
| CSVTextFormatter formatter1 = new CSVTextFormatter();  System.out.println(formatter1.formatCSVText("Bing.is.a.good.man.")); |

This violates “Program to an interface” No points will be given.

# Problem 7: TankWar (10 pts)

Read the source code and answer the following questions.

What does PaintThread do in this program? Or Why do we need this class?

|  |
| --- |
| PaintThread continuously repaints the program to ensure that the view is updated every 50 milliseconds. |

What does PropertyMgr do in this program? Or Why do we need this class?

|  |
| --- |
| The PropertyMgr class loads the predefined properties in confing/tank.properties. This allows us to predefine values such as number of initial tanks and respawn count outside of the source code so it is not hard coded in. |

In the paint method in TankWarGame class, what is the purpose of each for loop?

|  |
| --- |
| The purpose of each for loop in the paint method of TankWarGame is to draw all the tanks, missiles, and explosions contained in each list. |

When an enemy tank is hit by a missile from our tank? What is the logic from there? You can use your own word to describe.

|  |
| --- |
| When an enemy tank is hit, it is checked if the missile is alive and the enemy tank is alive. If the tank is good, remove 20 health points from the tank and check if it is still alive. If it is not still alive, change live to false, else do nothing. If the tank hit is not good, change live to false. Then set live of the missile to false and generate a new explosion, returning true. If the initial check failed, return false. |

How does the Explosion work?

|  |
| --- |
| Whenever Explosion.draw is called, the explosions are initialized (if not already) and checked to be alive and to see if we have stepped through all the images of the explosion. While we have not stepped through all the images, keep painting a new image generating an explosion like image “slideshow”. |