Homework 5

Due Nov 17, 2020

12 pts each.

What should be turned in?

* This word doc
* Homework5 project in zip

# Problem 1: Flower Bouquet

(You need to use the decorator pattern to complete this problem, no credits will be given if decorator pattern is not used.)

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 need to use the builder pattern to complete this problem, no credits will be given if builder pattern is not used.)

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 s1 = new SelectStatement.SelectStatementBuilder("pnumber, pname, count(\*)", "Project join Works\_on on pnumber=pno").setGroupBy("pnumber, pname").setHaving("count(\*) > 2;").build(); |

|  |
| --- |
| 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 s2 = new SelectStatement.SelectStatementBuilder("fname, lname, address", "Employee joins Department on dnumber=dno").setWhere("dname=\'research\';").build(); |

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

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

|  |
| --- |
| SelectStatement s3 = new SelectStatement.SelectStatementBuilder("fname, lname, address", "Employee").build(); |

|  |
| --- |
| SELECT pnumber, pname, budget, count(\*) 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 s4 = new SelectStatement.SelectStatementBuilder("pnumber, pname, budget, count(\*)", "Project join Works\_on on pnumber=pno")  .setWhere("budget > 200000").setGroupBy("pnumber, pname, budget").setHaving("count(\*) > 2").setOrderBy("pnumber").build(); |

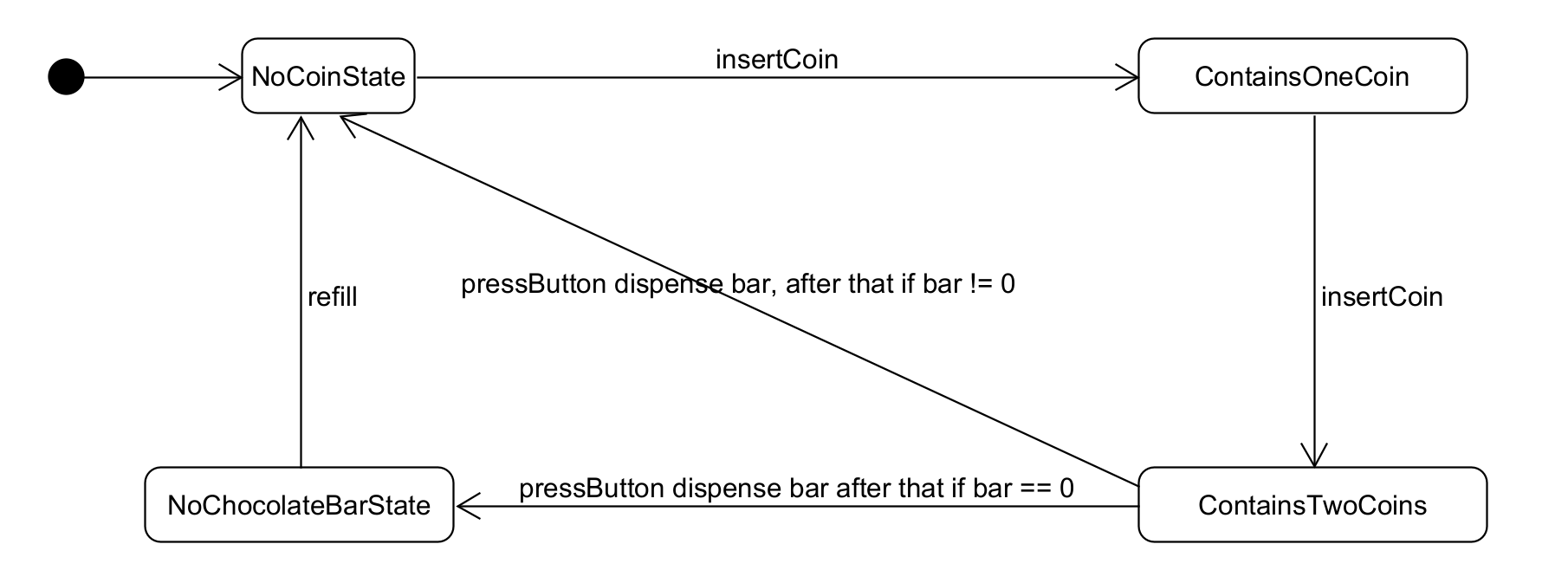
**Dang the Builder Design Pattern is cool! Being able to do this with just one public constructor of SelectStatement is amazing!!**

# Problem 3: Vending Machine

(You need to use the state pattern to complete this problem, no credits will be given if state pattern is not used.)

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 need to use the observer pattern with Guava EventBus to complete this problem, no credits will be given if observer pattern or Guava is not used.)

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

(You need to use the strategy pattern to complete this problem, no credits will be given if strategy pattern is not used.)

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?

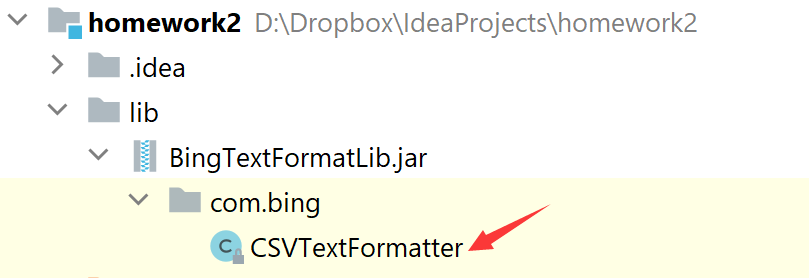
How would you modify your code? (Just write your thoughts here, no need to write code)

|  |
| --- |
| Create a new Strategy Class that implements the Discounter Interface |

# Problem 6: Text Formatter

(You need to use the adapter pattern to complete this problem, no credits will be given if adapter pattern is not used.)

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: Logger

(You need to use the proxy pattern to complete this problem, no credits will be given if adapter proxy is not used.)

Scenario 1: You are given some legacy Java code of a Tank game. Class Tank implements Movable interface. Now we want to log the execution time of invoking move() method to the console.

Code your solution here (use static proxy):

|  |
| --- |
| public class MoveableProxy implements Movable {   private static Movable *tank*;  @Override  public void move() {  if (*tank* == null) {  *tank* = new Tank();  }  long startTime = System.*nanoTime*();  *tank*.move();  long endTime = System.*nanoTime*();  long duration = (endTime - startTime); //divide by 1000000 to get milliseconds.  System.*out*.println(duration);  } } |

Scenario 2: what if you don’t have access to the source code of Tank class, in other words, this Tank class is from a JAR? In this case, you still can use it (e.g. instantiate this class using keyword new and invoke its methods like tank.move()), but you are not able to modify the source code like what you did in scenario 1. A good way to think about this is: can you modify the code in ArrayList<T> class? No, you can only use it, since it is part of JDK. Feel free to create other class under scenario2 package.

But we still want to log the execution time of invoking move() method to the console.

Code your solution here (use dynamic proxy):

|  |
| --- |
| public class TestTank {  public static void main(String[] args) { // // record move() execution time of a tank // TimerProxyFactory proxyFactory = new TimerProxyFactory(); // // // THIS NEEDS TO BE CAST TO MOVEABLE INTERFACE // Tank tank = (Tank) proxyFactory.createProxy(new Tank()); // // //THIS WILL NOT WORK CLASS CAST EXCEPTION // tank.move();    //EXTERNAL SOLUTION MADE USING JAVASSIST LIBRARY  ExternalSolution externalSolution = new ExternalSolution();  Tank tank = null;  try {  tank = (Tank) externalSolution.createObject();  } catch (InvocationTargetException e) {  e.printStackTrace();  } catch (NoSuchMethodException e) {  e.printStackTrace();  } catch (InstantiationException e) {  e.printStackTrace();  } catch (IllegalAccessException e) {  e.printStackTrace();  }  tank.move();  } |

# Problem 8: Refactoring Code

Your project needs to process a file including comma separated integers. Your colleague, John Smith, already wrote a program to sort those number in ascending order. John is very good at data structures, algorithms, and big data processing, so he considers different situations about this given file.

If the given file size is small, say less than 6GB, the program will read the file into RAM (e.g. the RAM of the server is 8GB) and perform quick sorting; if the file is large and cannot fit into RAM at once, he uses external sorting algorithm (a very common interview question! In case you are interested, <https://en.wikipedia.org/wiki/External_sorting>); if the file is super large, say, larger than 100GB, he uses concurrent external sort (sorting on multiple servers); if the file is super super super large, he uses map reduce sorting (<https://en.wikipedia.org/wiki/MapReduce>).

You can find his code here.

|  |
| --- |
| package problem7;  import java.io.File;  public class Sorter {  private static final long *GB* = 1000 \* 1000 \* 1000;   public void sortFile(String filePath) {  File file = new File(filePath);  long fileSize = file.length();  if (fileSize < 6 \* *GB*) { *// [0, 6GB)* quickSort(filePath);  } else if (fileSize < 10 \* *GB*) { *// [6GB, 10GB)* externalSort(filePath);  } else if (fileSize < 100 \* *GB*) { *// [10GB, 100GB)* concurrentExternalSort(filePath);  } else { *// [100GB, ~)* mapreduceSort(filePath);  }  }   private void quickSort(String filePath) {  *// ...algorithm omitted* }   private void externalSort(String filePath) {  *// ...algorithm omitted* }   private void concurrentExternalSort(String filePath) {  *// ...algorithm omitted* }   private void mapreduceSort(String filePath) {  *// ...algorithm omitted* } }  public class SortingTool {  public static void main(String[] args) {  Sorter sorter = new Sorter();  sorter.sortFile(args[0]); *// args[0] stores the file path*  } } |

John created a Pull Request on GitHub and appoint you as the code reviewer. How would you help refactor the code?

Please provide your refactored code in IntelliJ. Remember, your code should be SRP, OCP.

Hint: what if new sorting algorithm is being added or old algorithm being removed?