Program 2

Due 11:59 pm, May 26

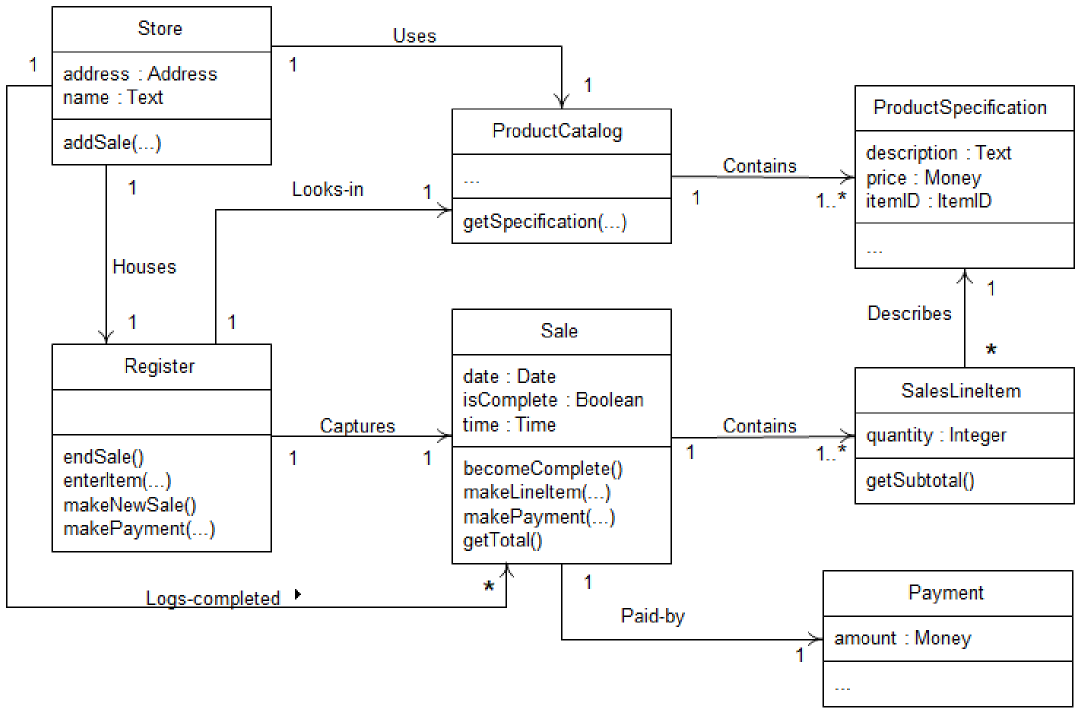
In this program, you are improving the POST system that you worked with in hw1. Follow the contents provided in OOP-Post-CaseStudy.ppt.

You still need to work on the same item data in hw1, but now the item code is changed as shown:

item code	item name	unit price
A001	corn_dog	1.25
B002	candy	0.98
B003	chocolate	1.79
B004	gum	0.45
C005	soda	1.29
C006	orange_juice	1.39
C007	bottle_water	0.89
A008	popcorn	1.50
A009	pretzel	2.35

Provide the item data in a file and read them into your program. The item code type is String.

Refer to the following class diagram covered in the lecture.



Your program must have the following classes: Register (this is the main class, like Post), Sale, SalesLineItem, ProductCatalog, and ProductSpecification. The Payment and Store classes are optional. You can add more classes if you want.

Also, you are required to use the code provided in the note for each of those classes. (The code is not complete, and it might have some minor syntax errors too.) Here shows the codes covered in the lecture. Provide separate files for each class.

```
class Register {
   private Sale sale;
   private ProductCatalog catalog;
   public void makeNewSale() {
      sale = new Sale();
   }
   public void enterItem(String id, int quantity) {
      ProductSpecification spec = catalog.getSpecification(id);
      sale.makeLineItem(spec, quantity);
   }
}
class Sale {
   private List<SalesLineItem> saleLineItems = new ArrayList<>();
   private Date date = new Date();
   private boolean isComplete = false;         //may not be needed

   public void makeLineItem(ProductSpecification spec, int quantity) {
      saleLineItems.add(new SalesLineItem(spec, quantity));
   }
   public boolean becomeComplete() { isComplete = true; }

   public float getTotal() {
      float total = 0;
      Iterator it = saleLineItems.iterator();
      while (it.hasNext()) {
         SalesLineItem sli = (SalesLineItem) it.next();
         total.add(sli.getSubtotal());
      }
      return total;
   }
}
public class ProductCatalog {
   private Map<String, ProductSpecification> productSpecs = new HashMap<>();
   public ProductCatalog() { } // build productSpec (by reading from the item file)
   public ProductSpecification getSpecification(String id) {
      return productSpecs.get(id);
   }
}
class ProductSpecification {
   private float price;
   private String description;

   public ProductSpecification(float price, String description) {
      this.price = price;
      this.description = description;
   }
   public float getPrice() { return price; }
   public String getDescription() { return description; }
}
class SalesLineItem {
   ...
}
```

Here shows changes to be made in this homework:

- The Item class used in hw1 is no longer needed in this homework. Instead, you need to implement the ProductCatalog and ProductSpecification classes.
- The description field in the ProductSpecification is for the item name, the price field is for the unit price.
- You do not need itemID in the ProductSpecification: itemID is for the item code and it is used as the key for the HashMap productSpecs in the ProductCatalog class.
- The Register class works like the Post class in hw1.
- Provide the main method in a class called Main.
- A good portion of the codes done in hw1 can be reused. However, in this program you make your program more object oriented so that it can be maintained well for future change.

Your program should work like hw1, with one more feature: showing the item description. Here shows the working of the program.

```
$ javac Main.java
$ java Main
Welcome to POST system!

Beginning a new sale (Y/N) y            ;both the upper and lower case should work
--------------------
Enter product code: B002                ;the user input stays right to the message
         item name: candy
Enter quantity:     2
        item total: $   1.96            ;the output should be aligned properly

Enter Product code: B004
         item name: gum
Enter quantity:     1
        item total: $   0.45

Enter product code: -1
----------------------------
Items list:
    2 candy         $   1.96
    1 gum           $   0.45
Subtotal            $   2.41
Total with Tax (6%) $   2.56
Tendered amount     $   5               ;the output should be properly aligned
Change              $   2.44            ;if less amount is entered, ask again
----------------------------


Beginning a new sale (Y/N) Y
--------------------
Enter product code: B003
         item name: chocolate
```

```
Enter quantity:     2
        item total: $   3.58

Enter Product code: A008
         item name: popcorn
Enter quantity:     4
        item total: $   6.00

Enter product code: 3
         item name: chocolate
Enter quantity:     1
        item total: $   1.79

Enter product code: 0000                ; you enter this value to show the items

item code     item name     unit price
A001          corn_dog      1.25
B002          candy         0.98
B003          chocolate     1.79
B004          gum           0.45
C005          soda          1.29
C006          orange_juice  1.39
C007          bottle_water  0.89
A008          popcorn       1.50
A009          pretzel       2.35

Enter product code: C007
         item name: bottle_water
Enter quantity:     3
        item total: $   2.67

Enter Product code: A008
         item name: popcorn
Enter quantity:     2
        item total: $   3.00

Enter product code: -1
----------------------------
Items list:
    3 chocolate     $   5.37      ;note that chocolate is listed once
    3 water         $   2.67
    6 popcorn       $   9.00
Subtotal            $  17.04
Total with Tax (6%) $  18.06
Tendered amount     $  20
Change              $   1.94

Beginning a new sale (Y/N) N

===================
The total sale for the day is  $  20.62

Thanks for using POST system. Goodbye.
```

In the sample run, the bold face letters are output from the POST system and non-bold italicized letters are input from the user. The user enters '-1' to terminate each current sale. The POST system is terminated if the user enters 'N' on **`Beginning a new sale (Y/N).`** Then, the POST system prints out the total sale amount of the day.

Your program should compile and run in the command line environment. All user input needs to be checked properly to avoid exceptions.

Provide two sample runs: one for your own that has three sales and one that is shown above – follow the exact same input sequence. Provide output files “hw1-out1.txt” and “hw2-out2.txt” for the two sample runs. The program output on the screen can be copied and pasted into a text file. Do not submit the screen image.

To get a full point

- Properly document/comment your program. Your program header comment for PostTest.java should be like

  ```
  // Homework 2: POST Program
  // Course: CS 216
  // Due date: xxx
  // Name: Jon Doe
  // Instructor: Alexander Antonio
  // Program description: ...

  public class Main
  {
     ...
  }

  ...
  ```
- As noted, do not provide program comments with the Javadoc format.
- Properly layout and indent your code with Javadoc comment format.
- Format your output properly. (You can use the printf function).
- Avoid exceptions due to wrong input type.
- Run Javadoc on your code
- Zip your program (all Java files), the Javadoc generated files, the input file (that contains the item data), and the output files with two sample runs (hw2-out1.txt and hw2-out2.txt) to as lastName-hw2.zip and submit to Canvas.