CSE 102 Programming Assignment 2

Due:

Sunday, 16-April-2023 by 23:59

Deliverables:

The following Java file should be submitted to MS Teams by the due date and time specified above. Submissions received after the deadline will be subject to the late policy described in the syllabus.

- Assignment02_{StudentNumber}.java

Specifications:

Overview: You will continue the program this semester to maintain the inventory for a store. Do not forget your headers with @author and @since information.

Requirements: Write and modify the following set of classes (All constructors should take parameters in the order given and initialize empty collections)::

1. ~~Product~~
    a. ~~Attributes~~
        i. ~~Id: Long~~
        ii. ~~No change to other attributes~~
    b. Methods – all methods from Assignment 01 with following modifications
        i. setPrice(price: double): None
            1. throws InvalidPriceException if price is negative
                a. Also do this in the constructor
        ii. addToInventory(amount: int): int
            1. throws InvalidAmountException if amount is negative
                a. Also do this in the constructor
        iii. purchase(amount: int): double
            1. if amount is negative, throws InvalidAmountException with only amount as the parameter
            2. if amount greater than quantity, throws InvalidAmountException with both the amount requested and remaining quantity
            3. Otherwise, decrease count by amount and return the total price (amount X price)
2. FoodProduct – a child of Product
    a. Attributes – no change
    b. Methods
        i. setCalories(calories: int): None
            1. throws InvalidAmountException if calories is negative
                a. Also do this in the constructor
3. CleaningProduct – no change
4. Customer
    a. Attributes – no change (hint: additional attributes will make this class easier to implement)
    b. Methods

i. addToCart(product: Product, count: int): None
    1. Adds the passed Product and number to the customer cart
    2. For this assignment, it will mean calling the purchase() method of the Product passed and adding to the receipt String and totalDue if the purchase is successful
    3. If the purchase() method throws an InvalidAmountException, this method catches it and displays a message to the screen beginning with "ERROR: "
ii. receipt(): String
    1. returns each Product in the cart on a separate line in the format below
    {Product Name} – {Product Price} X {count} … {total for Product}
    Total Due – {Total amount}
iii. getTotalDue(): double – returns the total amount due
iv. pay(amount: double): double
    1. If amount is greater than or equal to total due, displays a "Thank you" message to the screen, clear the cart, and returns the amount that should be given as change
    2. If amount is less than total due, raises a InsufficientFundsException

5. ClubCustomer – a child of Customer
    a. Attributes – no change
    b. Methods
        i. pay(amount: double, boolean: usePoints): double
            1. Performs the same task as the pay method from the Customer class
            2. If the customer has points available and usePoints is true, will subtract 0.01 from the total for every point the customer has.
            3. If usePoints is true and the total after using the points would be 0, reduces the customer's points by the number of points used (i.e. points = points - total * 100)
            4. Otherwise, if usePoints is true and payment is successful, reduces the customer's points to 0.
            5. Also, if payment is successful, adds points to the customer total using the calculation of 1 point for every whole 1 TL spent (i.e. the total after any reduction from points)
6. Store
    a. Attributes
        i. A collection of ClubCustomer objects
        ii. A collection of Product objects
            1. Products will no longer be stored by index
    b. Methods

i. getName() and setName(name: String) – no change
ii. getWebsite() and setWebsite(website: String) – no change
iii. getInventorySize() – no change
iv. addProduct(product: Product): None
    1. Adds the passed product to the collection
v. getProduct(ID: Long): Product
    1. returns the Product with the ID passed
    2. if the ID is not found in the collection, throws ProductNotFoundException
vi. getProduct(name: String): Product
    1. returns the Product with the name passed
    2. if the name is not found in the collection, throws ProductNotFoundException
vii. addCustomer(customer: ClubCustomer): None
    1. Adds the passed Customer to the set
viii. getCustomer(phone: String): ClubCustomer
    1. returns the ClubCustomer with the phone number passed
    2. if the phone number is not found in the set, throws CustomerNotFoundException
ix. removeProduct(ID: Long): None
    1. removes the Product with the ID passed from the set
    2. if the ID is not found in the set, throws a ProductNotFoundException
x. removeProduct(name: String): None
    1. removes the Product with the name passed from the set
    2. if the name is not found in the set, throws a ProductNotFoundException
xi. removeCustomer(phone: String): None
    1. removes the ClubCustomer with the phone number passed from the set
    2. if the phone number is not found in the set, throws a CustomerNotFoundException

7. ~~Custom Exceptions~~
    a. ~~CustomerNotFoundException – instance of **IllegalArgumentException**~~
        i. ~~Additional Attribute – phone: String~~
        ii. ~~toString() "CustomerNotFoundException: " + phone~~
    b. ~~InsufficientFundsException – instance of **RuntimeException**~~
        i. ~~Additional Attributes – total: double, payment: double~~
        ii. ~~toString() "InsufficientFundsException: " + total + " due, but only " + payment + " given"~~
    c. ~~InvalidAmountException – instance of **RuntimeException**~~
        i. ~~Additional Attribute – amount: int, quantity: int~~
        ii. ~~Constructor that takes only amount parameter~~

iii. Constructor that takes both amount and quantity
iv. toString()
1. if only constructor with only amount was used, "InvalidAmountException: " + amount
2. if constructor with both parameters was used, "InvalidAmountException: " + amount + " was requested, but only " + quantity + " remaining"

d. ~~InvalidPriceException – instance of **RuntimeException**~~
~~i. Additional Attribute – price: double~~
~~ii. toString() "InvalidPriceException: " + price~~

e. ~~ProductNotFoundException – instance of **IllegalArgumentException**~~
~~i. Additional Attribute – ID: Long, name: String~~
~~ii. Constructor that takes only a Long parameter – sets the name attribute to null~~
~~iii. Constructor that takes only a String parameter – sets the Long to any value~~
~~iv. toString()~~
~~1. if name is null, "ProductNotFoundException: ID - " + ID~~
~~2. Otherwise, "ProductNotFoundException: Name - " + name~~

Design: Your program does not require a main method. You are only responsible for creating the six (6) classes and five (5) Exceptions described above.
Code: The file you submit will be named Assignment02_{StudentNumber}. You should put all java classes for this assignment inside of this file as discussed in class.
Test: You are responsible for testing your program. It is important to not rely solely on the examples presented in this Assignment description. It would be a very good idea to write your own test cases for this assignment.

Grading:
MS Teams Submission: If anything is ambiguous, it is your responsibility to ask questions. It is also your responsibility to complete this assignment in a timely manner. Questions regarding this assignment will likely not be answered if received after 17:00 on the last weekday before the due date of the assignment.

```java
public class Assignment02_123456789 {
   public static void main(String[] args) {
      Store s = new Store("Migros", "www.migros.com.tr");

      Customer c = new Customer("CSE 102");

      ClubCustomer cc = new ClubCustomer("Club CSE 102", "05551234567");
      s.addCustomer(c); // compiler error because c is Customer not ClubCustomer
      s.addCustomer(cc);

      Product p = new Product(123456L, "Computer", 20, 1000.00);
      FoodProduct fp = new FoodProduct(456798L, "Snickers", 100, 2, 250, true, true, true, false);
      CleaningProduct cp = new CleaningProduct(31654L, "Mop", 28, 99, false, "Multi-room");

      s.addProduct(p);
      s.addProduct(fp);
      s.addProduct(cp);

      System.out.println(s.getInventorySize());
      System.out.println(s.getProduct("shoes"));  // Exception due to product not being in store

      System.out.println(cp.purchase(2));
      s.getProduct("Computer").addToInventory(3);
      System.out.println(fp.purchase(200));  // results in Exception

      c.addToCart(p, 2);
      c.addToCart(s.getProduct("snickers"), -2); // NOTE: This does not stop the program because the Exception is caught
      c.addToCart(s.getProduct("snickers"), 1);
      System.out.println("Total due - " + c.getTotalDue());
      System.out.println("\n\nReceipt:\n" + c.receipt());

      System.out.println("After paying: " + c.pay(2000)); // results in Exception

      System.out.println("After paying: " + c.pay(2020));

      System.out.println("Total due - " + c.getTotalDue());
      System.out.println("\n\nReceipt 1:\n" + c.receipt());

      Customer c2 = s.getCustomer("05551234568");  // Exception
      cc.addToCart(s.getProduct("snickers"), 2);
      cc.addToCart(s.getProduct("snickers"), 1);
      System.out.println("\n\nReceipt 2:\n" + cc.receipt());

      Customer c3 = s.getCustomer("05551234567");
      c3.addToCart(s.getProduct("snickers"), 10);
      System.out.println("\n\nReceipt 3:\n" + cc.receipt());

      System.out.println(((ClubCustomer)c3).pay(26, false));

      c3.addToCart(s.getProduct(31654L), 3);
      System.out.println(c3.getTotalDue());
      System.out.println(c3.receipt());
      System.out.println(cc.pay(3*99, false));

      c3.addToCart(s.getProduct(31654L), 3);
      System.out.println(c3.getTotalDue());
      System.out.println(c3.receipt());
      System.out.println(cc.pay(3*99, true));
   }
```

An example main() method.

```
Assignment02_123456789.java:8: error: incompatible types: Customer cannot be converted to ClubCustomer
      s.addCustomer(c); // compiler error because c is Customer not ClubCustomer
                    ^
```

Compiler error

```
3
Product Shoes has 28 remaining
178.0
Exception in thread "main" InvalidPurchaseException: 200 requested, 100 remaining
    at Product.purchase(Assignment02_123456789.java:79)
    at Assignment02_123456789.main(Assignment02_123456789.java:26)
```

Commenting out compiler error gets first Runtime error (“`s.getProduct("shoes")`”)

```
3
Exception in thread "main" ProductNotFoundException: ID - 0
    at Store.getProduct(Assignment02_123456789.java:333)
    at Assignment02_123456789.main(Assignment02_123456789.java:20)
```

```
3
198.0
Exception in thread "main" InvalidAmountException: 200 was requested, but only 100 remaining
   at Product.purchase(Assignment02_123456789.java:113)
   at Assignment02_123456789.main(Assignment02_123456789.java:24)
```

Commenting out first Exception line gets second Exception (“`fp.purchase(200)`”)

```
3
198.0
ERROR: InvalidAmountException: -2
Total due - 2002.0


Receipt:
Computer - 1000.0 X 2 = 2000.0
Snickers - 2.0 X 1 = 2.0

-------------------------------------

Total Due = 2002.0
Exception in thread "main" NotEnoughPaymentException: 2002.0 due, but only 2000.0 given
   at Customer.pay(Assignment02_123456789.java:239)
   at Assignment02_123456789.main(Assignment02_123456789.java:32)
```

Commenting out second Exception line gets third Exception (“c.pay(2000)”)

```
Total Due = 2002.0
Thank you for shopping with us.
After paying: 18.0
Total due - 0.0


Receipt 1:

-------------------------------------

Total Due = 0.0
Exception in thread "main" CustomerNotFoundException: 05551234568
   at Store.getCustomer(Assignment02_123456789.java:344)
   at Assignment02_123456789.main(Assignment02_123456789.java:39)
```

Commenting the third Exception line gets fourth Exception
("`s.getCustomer("05551234568")`")

Final output after commenting out all Exception lines

```
 ----jGRASP exec: java Assignment02_123456789
3
198.0
ERROR: InvalidAmountException: -2
Total due - 2002.0


Receipt:
Computer - 1000.0 X 2 = 2000.0
Snickers - 2.0 X 1 = 2.0

-------------------------------------

Total Due = 2002.0
Thank you for shopping with us.
After paying: 18.0
Total due - 0.0


Receipt 1:

-------------------------------------

Total Due = 0.0


Receipt 2:
Snickers - 2.0 X 2 = 4.0
```

```
Snickers - 2.0 X 1 = 2.0

-------------------------------------

Total Due = 6.0


Receipt 3:
Snickers - 2.0 X 2 = 4.0
Snickers - 2.0 X 1 = 2.0
Snickers - 2.0 X 10 = 20.0

-------------------------------------

Total Due = 26.0
Thank you for shopping with us.
0.0
297.0
Mop - 99.0 X 3 = 297.0

-------------------------------------

Total Due = 297.0
Thank you for shopping with us.
0.0
297.0
Mop - 99.0 X 3 = 297.0

-------------------------------------

Total Due = 297.0
Thank you for shopping with us.
3.230000000000018

 ----jGRASP: operation complete.
```