Lab 8

Requirements:

- Create a Java project named yourStudentId_OOP_Lab8
- Read instructions and create classes needed. You are supposed to add 3 classes (*Fruit, Register, and Tester*) to the project. The *Fruit* class has the same content as last week.
- All instance variables are private. Please use public interfaces to access private variables.

1. Create *Fruit* class (Same as Lab7)

Fruit	
Modifier and type	**Method (or Variable) and description**
Instance variable	
String	name The name of the fruit.
int	price The price of the fruit.
int[]	sale The individual sale of the fruit.
int	totalSales The total sales of the fruit.
Constructor	
Fruit(String name, int price) Enable to construct a *Fruit* object with given name, price and an empty array of sale that can store 3 records. Meanwhile, initializes the *totalSales* as 0.	
Instance methods	
-	3 getter for 3 attributes (getName(),getPrice(), and getTotalSales ()). 2 setter for 2 attributes (setName(...), setPrice(...)).
void	updateTotalSales(int amount) Accumulate all sales from different carts. a. Add the value of *amount* to *sale* array. b. Accumulate the *amount* into the *totalSales* attribute.
String	getInfo() a. Return a String contains name, price, individual sale, and total sale of the *Fruit*. b. Individual sale should sort out the sale array from small to large. c. Use *"for-each"* concept to print out the content of sale array. d. You should follow the following formatted layout:

2. Create *Register* class

Register	
Modifier and type	**Method (or Variable) and description**
Instance variable	
int	totalRevenue The sum of all bills spent.
ArrayList<Integer>	bills Store each bill spent.
Constructor	
Register() Initialize *totalRevenue* as 0 and the ArrayList named *bills*.	
Instance methods	
-	2 getter for 2 attributes (gettotalRevenue(), getBill(int id)). a. The *getBill()* function must use the input *id* to find out from the ArrayList.
void	calctotalCost(int id, int num, int price) Calculate all expenses of a single order. a. Use the *"try-catch"* concept to determine whether the ArrayList already has a value. If an *"IndexOutOfBoundsException"* occurs, it means that there is no such space in the ArrayList, so you need to use *Array.add()* to add the expense to the ArrayList. Otherwise, use *Array.set()* to update the value to the corresponding index position. b. Whenever you add or update the record, you need to call *this.calctotalRevenue()*.
void	calctotalRevenue(int cost) a. Calculate the current store's total revenue and update the result to *totalRevenue*.

String	getInfo()
	a. The returned *String* object contains all of the expense and the final value of *totalRevenue* in the following format (must include all the contents of the sample output).
	b. Use the *"for-each"* concept to obtain each bill spent in the ArrayList.
	c. At the same time, the final value of *totalRevenue* should be returned.
	Sample output:
	`My register info:`
	`Bill`
	`192`
	`172`
	`152`
	`188`
	`251`
	`132`
	`117`

```
283
241
296
Total Revenue:2024
```

3. The *main* method in *Tester* class

a. Construct the Fruit objects named *Apple, Banana, and Orange*, and their corresponding prices are $10, 12, and 15. At the same time, store these objects in the Array named *fruits*.

b. You must construct the following types of objects and follow the naming convention:

- The Register object named *register* is to record all the information.
- The File object named *myObj* to open the file. The file contains ten records of each fruit purchased. (**Note: This file has been uploaded to Moodle, please download it to the path under this Project.)**
- The Scanner object named *reader* to read the data of the file.
- The FileWriter named *myWriter* writes the return value of *Register.getInfo()* into the *"register_info.txt"* file.

c. Use *Scanner.hasNetInt()* to determine whether the file has an *Int* type value. If so, please pass the cost per unit and amount purchased of the fruit in each order to *Register.calctotalCost()*.

d. Call *FileWriter.write()* to write out the return value of *Register.getInfo()*.

e. Check and import needed exceptions handling packages in the method. If an exception occurs, make sure to print an error message.

f. The Scanner and FileWriter objects need to be closed regardless of whether there is an exception.

g. You should choose a better way to use finally clause in a *try/catch* block, e.g., two nested try clauses to control the flow.

Submission: Submit your project as ".zip file" via Moodle. No other submissions will be graded.

Reminder: Please zip the whole project

Deadline: Tomorrow's midnight (for both Mon56 and Tue23)