CSC 151 Assignment 1

Objectives:

- Revisiting java and OOP principles
- Introducing Eclipse and unit testing

Definition

In this assignment you will simulate the shopping cost calculation. You will have two classes called **Product** and **Receipt**. A product is anything we can buy. Each product can have a unit price and quantity as an integer. The total cost of a product is calculated by *unit price x quantity*. While you are shopping you can buy any number(not really!!!) of products and your receipt will be prepared accordingly. In the receipt the products will be kept in an array and receipt will give you the total cost by adding the costs of all products you bought.

Considering the specifications provided above, please write the following classes.

Class Product

- Name of your file will be **Product.java** (case-sensitive)
- It will be in **package assignment** (it is required otherwise **Gradescope** cannot run the tests)
- It will have **two private integer instance variables** *unitPrice* and *quantity* and **one private String variable** *pName*(for product name)
- *unitPrice* can be an integer between **0 and 100** (use constant values for max and min values and verify these whenever you need to modify this value)
- *quantity* can be an integer between **0 and 10** (use constant values for max and min values and verify these whenever you need to modify this value)
- *pName* is a String and will have the value **"NO NAME"** if a name is not provided
- Provide two **public constructors**
    - One with no parameter and initialize integers variables to **0 (zero)** and String variable to **"NO NAME"**. Use chaining or set methods inside.
    - One with 3 parameters for *unitPrice*, *quantity* and *pName* and initialize the instance variables with the given values. Use set methods inside.

- Provide **public set/get** methods and do not forget to check restrictions for price and quantity
- Override **equals** method. Two Product objects will be equal if they have the same *unitPrice*, same *quantity* and same *pName*
- Override the **toString** method. When a Product object with *unitPrice*=35 and *quantity*=5 and *pName*=”Candy” is converted to a String it will be “Candy:35 x 5 = 175”
    - *pName*
    - followed by :
    - followed by *unitPrice*(no space before or after :)
    - followed by a single space
    - followed by *quantity*
    - followed by a single space
    - followed by x
    - followed by a single space
    - followed by =
    - followed by a single space
    - followed by the product of unitPrice and quantity and
    - no new line at the end.
- Provide a method named **total** which will calculate and return the total cost of this Product as *unitPrice* x *quantity*
- Test your Product class in the main method.
- Test your classes by JUnit

<u>**Class Receipt**</u>

- Name of your file will be **Receipt.java** (case-sensitive)
- It will be in **package assignment** (it is required otherwise **Gradescope** cannot run the tests)
- It will have **one private integer instance variable** *itemCount* and one Product array variable *receipt*(for keeping all products).
- *receipt* array can keep Product objects between **0 and 100** (use constant value for max and verify it whenever you need to modify the array)

- Provide a **public constructor** with no parameters and initialize the array
- Provide an **addItem** method that has a Product parameter and returns the *itemCount* after adding this Product. If the array is full, do not add the item and return the current *itemCount*
- Override the **toString** method. Return the String representation of each Product item in the receipt followed by a new line.
- Provide a method named **calcTotal** which will calculate and return the total cost of the Product items in the array by calling the **total** method of Product.
- Test your Receipt class in the **main** method

<u>**General Submission Instructions**</u>

- You may have groups for the assignments **if it is allowed.** If you have a group state it **explicitly in the honor code** by writing all group members' names. Please name the pdf files accordingly. **However, all group members should submit individually.**
- All submissions require a package called **assignment** and your source codes (.java files) will be under it.
- You should submit
  - Your java files to **Gradescope** as required and
  - Your java files and sample outputs for various scenarios as a **single pdf** file named as **YourFullName(s)_AssignmentN.pdf** with the **honor code** at the top as a comment to Nexus under the appropriate link. Replace N in AssignmentN with the appropriate value. (e.g., if I submit Assignment1 I will submit a pdf file whose name is **ZeynepOrhan_Assignment1.pdf.** If I have a group whose members are me and Jane Doe, then the file name will be **JaneDoeZeynepOrhanAssignment1.pdf** and the honor code will be modified). Use the template .docx file attached and modify it accordingly.
- **If you use any resource to get help, please state it in the honor code. Otherwise, it will be an honor code violation!!!**
- **Gradescope provides a feature to check code similarity by comparing all submissions. If the percentage of your code's similarity is above a threshold, then this will be an honor code violation!!!**

- Make sure that you have no compiler errors.
- When you have compiler error free codes and submit appropriately to Gradescope, your code will be tested automatically and graded. You do not need to know the test cases, but you will see the test results. Informative messages will be provided for success and failure cases.
- You can resubmit your files any number of times if you have failing tests until the due date.
- Your final submission or the one you selected will be graded after the due date of the assignment.
- Please start as early as possible so that you may have time to come and ask if you have any problems.
- Read the late submission policy in the syllabus.
- Please be aware that the correctness, obeying the OOP design and submission rules are equally important
- Gradescope testing will generally be 100% and sometimes manual grading will be done. Please be aware that **your grade can be reduced if you do not follow the instructions and apply good programming practices**.
- Use the starter code if provided.
- Do not use any predefined Collection classes of Java unless stated otherwise.

Assignment specific instructions

- General submission instructions apply.
- **Group work: NOT ALLOWED!!!**
- **Due date:** January 9, 2021 until 11:50 PM
- **Nexus:** Submit the code files (**Product.java, Receipt.java**) as a single pdf file named as **YourFullName_Assignment1.pdf** with the honor code at the top as a comment under Assignment 1 link.
- **Gradescope:** Submit the java files (**Product.java, Receipt.java**) under the Assignment 1
- **Grading:**
  - Gradescope: Yes
  - Manual: No

- **Starter code:** Use the code given below

```
/*
* Honor Code
*/
package assignment;
/**
 * Product class for your receipts.
 * Each Product has a name, unit price and quantity
 * Total cost is calculated as unit price x quantity
 *
 * @author
 * @version
 */
public class Product {
        //Limit values as constants




        /**
         * Instance variables for unit price, quantity, product name
         */





        /**
    * Constructor with no parameter
    * int instance variables are set to 0
    * String instance variable is set to NO NAME
    *
    *
    */





        /**
    * Constructor with 3 parameters
    *
    * @param unitPrice initial unit price
    * @param quantity initial quantity
    * @param pName initial name of the product
    *
    *
    */
```

```
        /**
    * get method
    * @return unit price as integer
    */

        /**
      * set method
           * @param unitPrice to set
           */

        /**
    * get method
    * @return quantity as integer
    */

        /**
      * set method
           * @param quantity to set
           */

        /**
    * get method
    * @return product name as String
    */

        /**
```

```
 * set method
	 * @param pName the pName to set
	 */





/* Override equals method
*/
	@Override







/* total method for calculating total
*/





/* Override toString method
*/
	@Override



/* main method for testing, change the one below completely
*/
	public static void main(String[] args) {

		Product p1=new Product();
		Product p2=new Product(10,20,"Candy");
		int total=p1.total()+p2.total();

		System.out.println(p1);
		System.out.println(p2);
		System.out.println(total);

		p1.setUnitPrice(35);
		p1.setQuantity(125);
```

```
                System.out.println(p1);
                System.out.println(p2);
                total=p1.total()+p2.total();
                System.out.println(total);

                p1.setQuantity(5);
                System.out.println(p1);
                System.out.println(p2);
                total=p1.total()+p2.total();
                System.out.println(total);

        }


}
```

```
package assignment;

/**
 * Receipt class for your receipts of Products.
 * Each Receipt has a collection of Product objects kept as array
 * Total cost is calculated as unit price x quantity for all
 * Product instances in the array
 *
 * @author
 * @version
 */
public class Receipt {
    /**
         * Limit value for max items as a constant
         */

        public static final int MAX_ITEMS=100;

    /**
         * private instance variables for receipt as array and item count
         */






        /**
         * Constructor with no parameter
         */






     /**
         * addItem method for adding a Product to the array
         */







         /**
         * calcTotal method for calculating the total cost of items
         */







         /**
         * Override toString method
         */

         /**
         * main method for testing, change the one below completely
         */

         public static void main(String[] args) {
                 // TODO Auto-generated method stub
                 Receipt mR=new Receipt();
                 for(int i=0; i<3;i++)
                         mR.addItem(new Product(i+1,(i+1)*3,"p"+i));
                 System.out.println(mR+" = "+ mR.calcTotal());

         }

 }
```