CEGEP VANIER COLLEGE
CENTRE FOR CONTINUING EDUCATION
Programming Algorithms and Patterns
420-930-VA

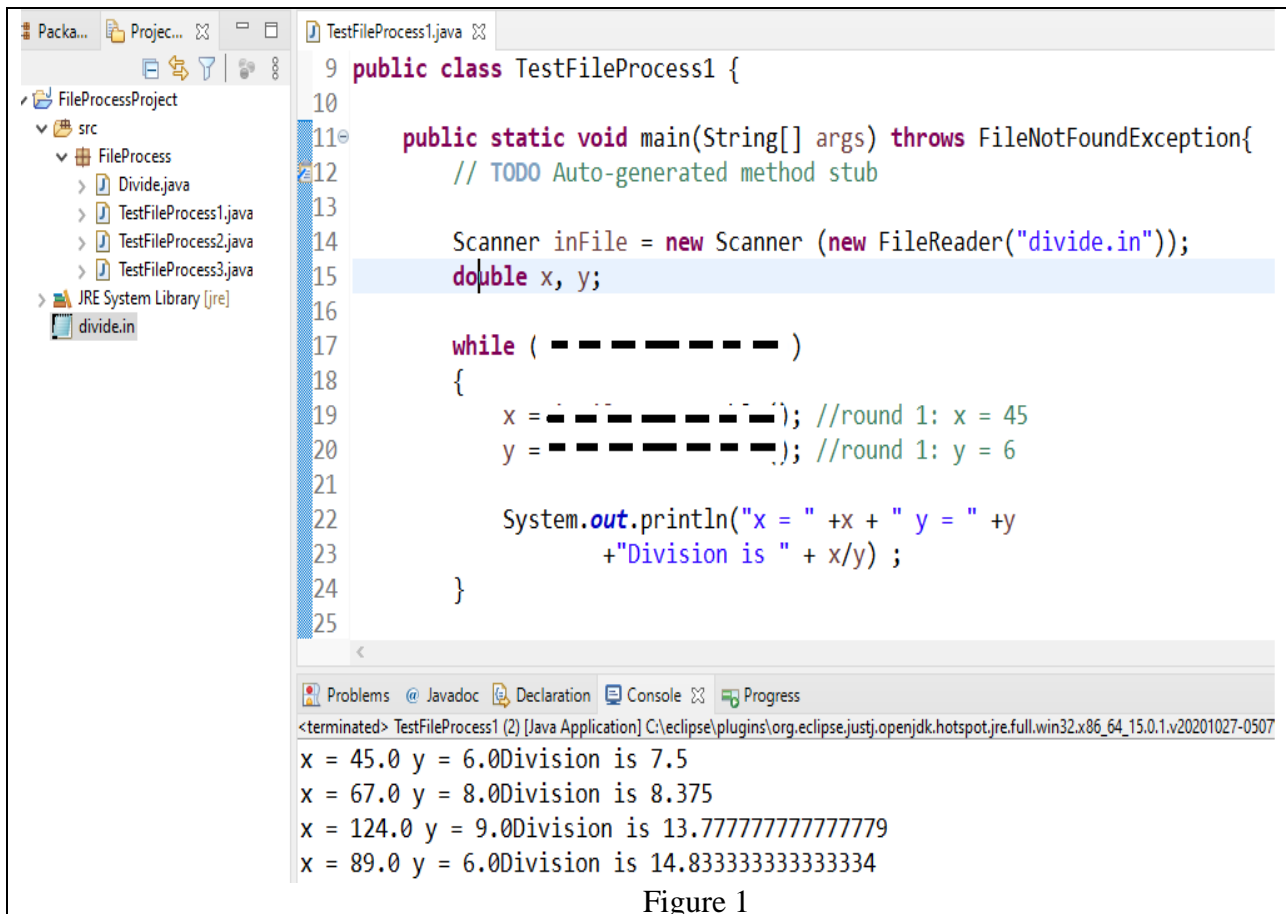**Teacher: Samir Chebbine**   **Lab 1**   **Jul 14, 2023**

Lab 1: Arrays, Arrays of objects and ArrayList

Complete all these following programs as explained during **Zoom Synchronous classes.** All *missing coding statements* were provided there with explanation. Create and Submit a Word file *Lab1OOPProgramminAlgorithmsYourName.docs* which includes output screenshots for every Java Project. Submit the Java projects too.

1. **Data Structure**
a) **Reading Data From Text File without Array**

Create a Java Project to be named ***FileProcessProject*** using Eclipse IDE for reading input file *divide.in*. Create *TestFileProcess1.java* to read from input file as shown in Figure 1.

Figure 1

b) **Reading Data From Text File and Storing it into Parallel Arrays**

Create *TestFileProcess2.java* to read from input file and storing its content into parallel arrays as shown in Figure 2.

```
10  public class TestFileProcess2 {
11    public static void main(String[] args) throws FileNotFoundException{
12        Scanner inFile = new Scanner (new FileReader("divide.in"));
13        double[] ▬ ▬ ▬ ▬ ▬ ▬ ▬ ▬    //index x[0] x[1] x[2] x[3]
14        double[] ▬ ▬ ▬ ▬ ▬ ▬ ▬ ▬
15        int index=0;
16
17        while ( ▬ ▬ ▬ ▬ ▬ ▬ ▬)
18        {
19            x[index] = ▬ ▬ ▬ ▬ ▬ ▬ ▬); //round 1: x[0] = 45
20            y[index] = ▬ ▬ ▬ ▬ ▬ ▬ ▬); //round 1: y[0] = 6
21
22            index++;    //round 2: index = 1  x[1] = 67
23        }
24        inFile.close();//Like Closing db and free space memory
25
26        //Print all array x and y. You can apply processing algorithms
27        System.out.println("Printing Parallel Arrays");
28        for (index=0; index< x.length; index++)
29            System.out.println("x = " +x[index] + " y = " +y[index]
30                    +" Division is " + x[index]/y[index]);
```

divide.in file contents:
```
45 6
67 8
124 9
89 6
```

Console output:
```
<terminated> TestFileProcess2 (1) [Java Application] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.1.v20201027-050(
Printing Parallel Arrays
x = 45.0 y = 6.0 Division is 7.5
y = 67.0 y = 8.0 Division is 8.375
```
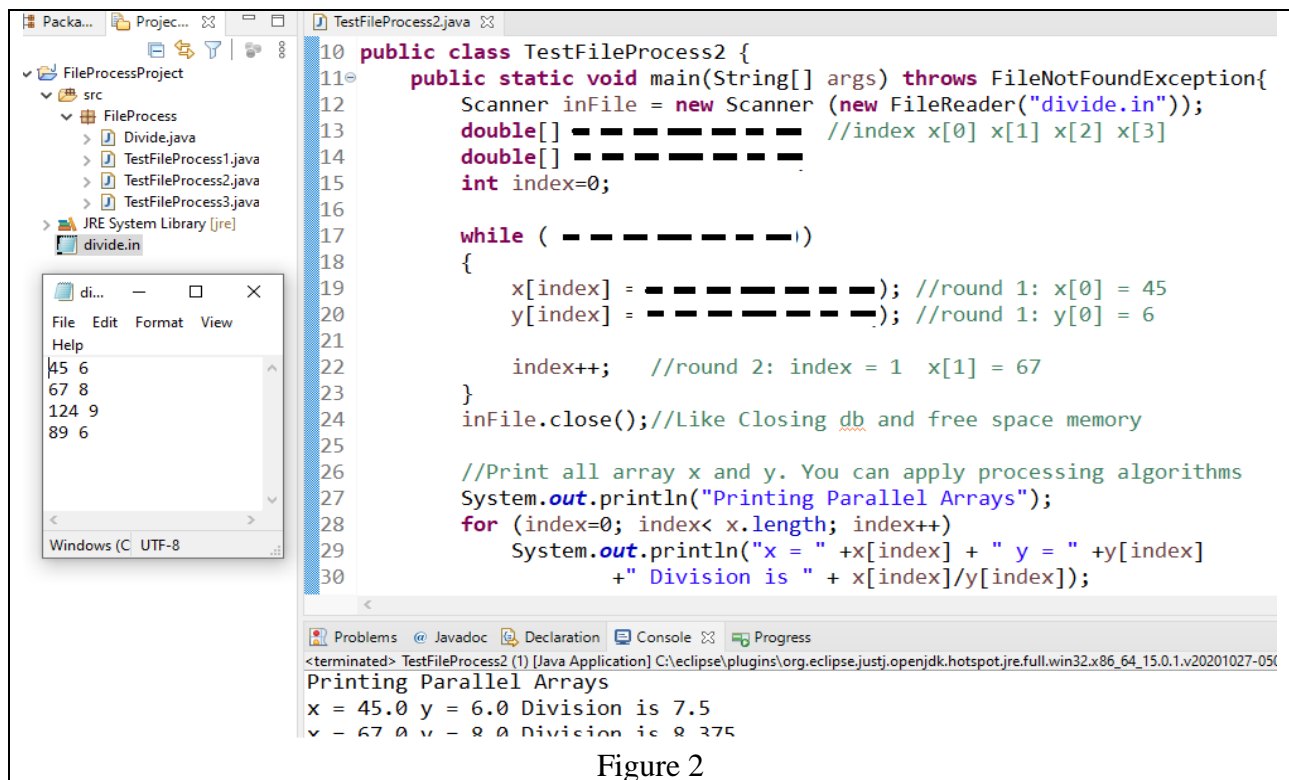
Figure 2

c) Reading Data From Text File and Storing it into Object of Arrays

Create *TestFileProcess3.java* to read from input file and storing its content into object of array as shown in Figure 3.
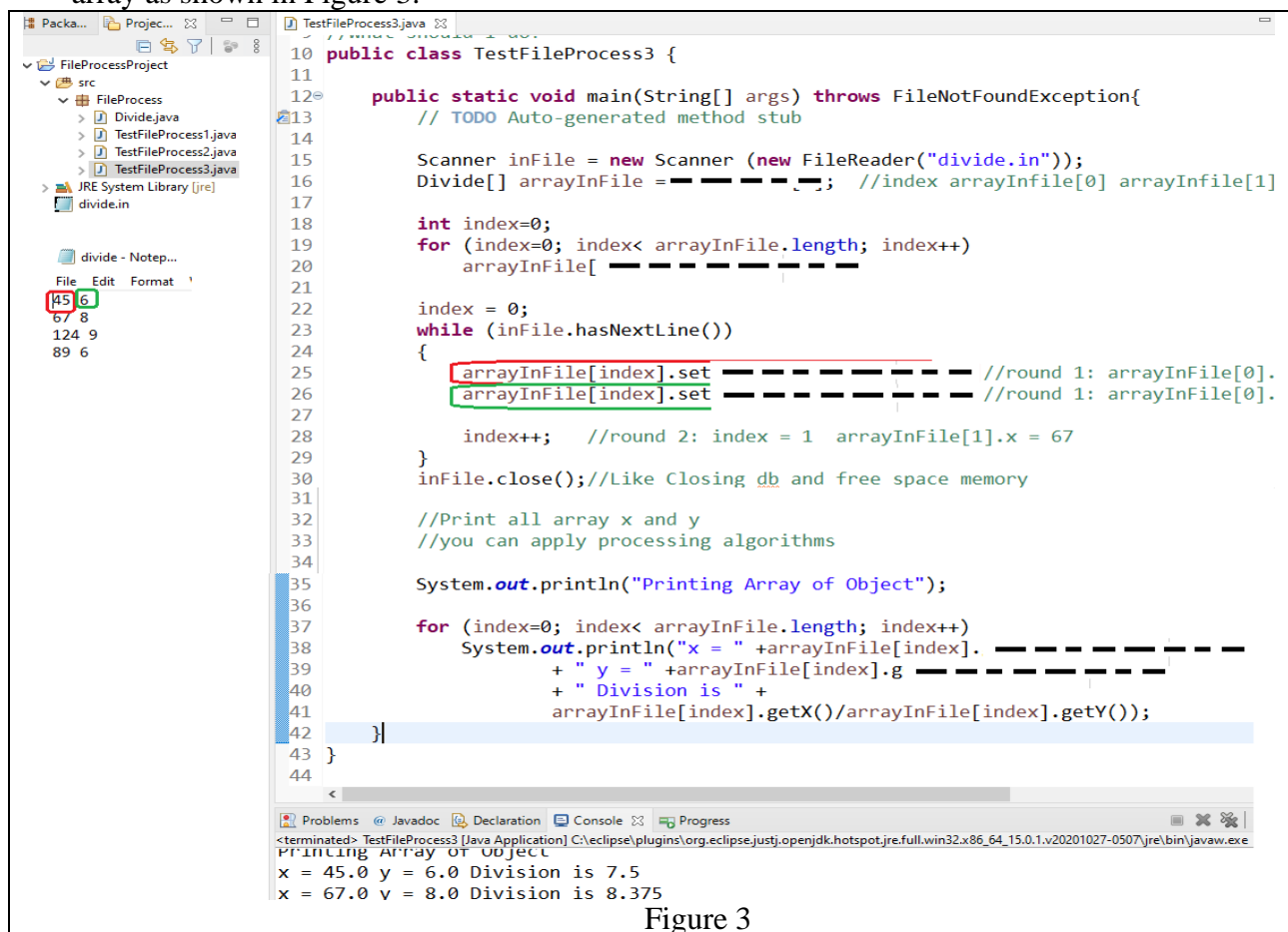


```
10  public class TestFileProcess3 {
11
12    public static void main(String[] args) throws FileNotFoundException{
13        // TODO Auto-generated method stub
14
15        Scanner inFile = new Scanner (new FileReader("divide.in"));
16        Divide[] arrayInFile =▬ ▬ ▬ ▬ ▬ ;  //index arrayInfile[0] arrayInfile[1]
17
18        int index=0;
19        for (index=0; index< arrayInFile.length; index++)
20            arrayInFile[ ▬ ▬ ▬ ▬ ▬
21
22        index = 0;
23        while (inFile.hasNextLine())
24        {
25            arrayInFile[index].set ▬ ▬ ▬ ▬ ▬ ▬ ▬ ▬ //round 1: arrayInFile[0].
26            arrayInFile[index].set ▬ ▬ ▬ ▬ ▬ ▬ //round 1: arrayInFile[0].
27
28            index++;    //round 2: index = 1  arrayInFile[1].x = 67
29        }
30        inFile.close();//Like Closing db and free space memory
31
32        //Print all array x and y
33        //you can apply processing algorithms
34
35        System.out.println("Printing Array of Object");
36
37        for (index=0; index< arrayInFile.length; index++)
38            System.out.println("x = " +arrayInFile[index]. ▬ ▬ ▬ ▬ ▬ ▬ ▬ ▬
39                    + " y = " +arrayInFile[index].g ▬ ▬ ▬ ▬ ▬ ▬ ▬
40                    + " Division is " +
41                    arrayInFile[index].getX()/arrayInFile[index].getY());
42    }
43  }
44
```

Console output:
```
<terminated> TestFileProcess3 [Java Application] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.1.v20201027-0507\jre\bin\javaw.exe
Printing Array of Object
x = 45.0 y = 6.0 Division is 7.5
x = 67.0 y = 8.0 Division is 8.375
```

Figure 3

2. ArrayLists

a) Create *ArrayListProcessProject* using Eclipse IDE for reading input file *divide.in*. Create *TestArrayList1.java* to to create ArrayList from List interface as shown in Figure 4.
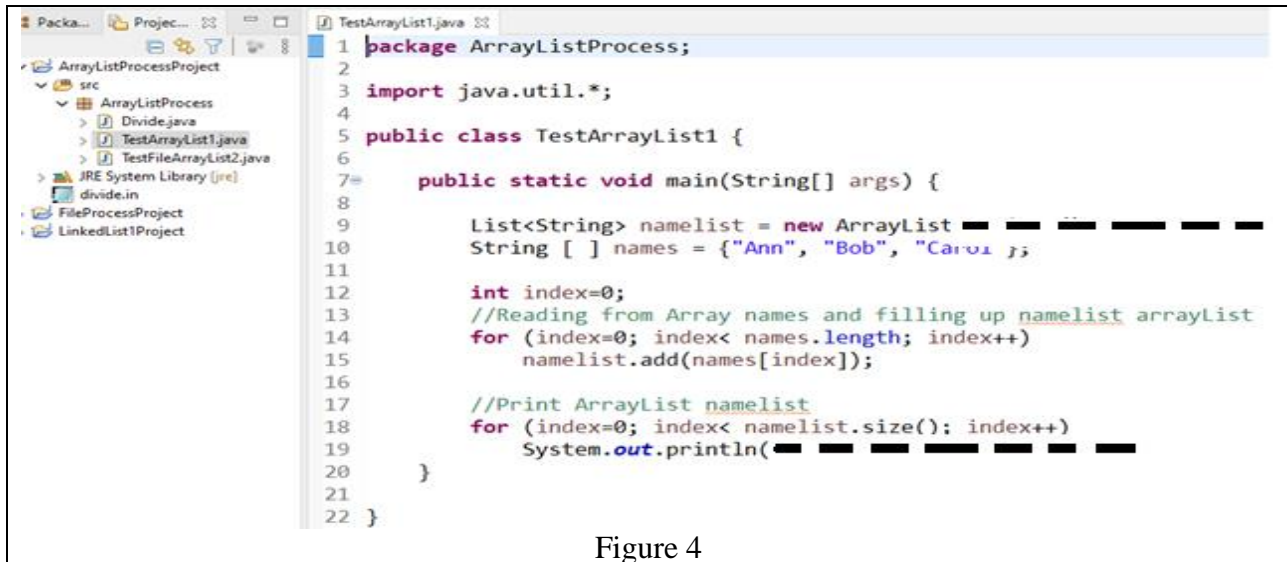

Figure 4

b) Reading Data from Text File and Storing it into ArrayList Data Structure
Create *TestFileArrayList2.java* to read from input file and storing its content into ArrayList as shown in Figure 5.
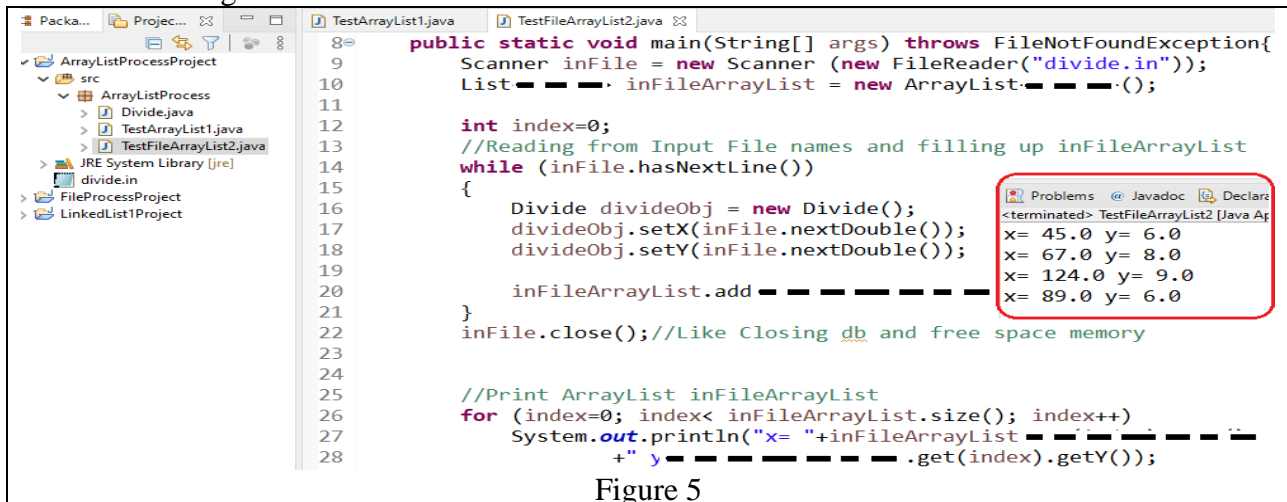

Figure 5

3. Using Array of Objects in GradeProject

a) Create a Java Project to be named *GradeProject* using Eclipse IDE to define a new data structure type, called *Grade*.

b) Create a class to define data structure type, called *Grade*, which is designed to group data and functions into a single unit that *represents* a template of the fields used in `Grade.in` as shown in the following Figure.

c) Each line within *Grade.in* represents a Grade's record with the following fields: `student_id (int), student_lname (string), student_fname (string), s_grade_Assignment1 (int), s_grade_Assignment2(int), s_grade_Assignment3(int), s_grade_Mid_Term (int), s_grade_Final_Term (int)`.

d) Add **default constructor** `(student_id=0, student_fname="", student_lname="", s_grade_Assignment1=0,s_grade_Assignment2=0,` s_grade_Assignment3=0,

s_grade_Mid_Term=0, s_grade_Final_Term=0) and **constructor with parameters** (get the same name as the name of the class) within the *Grade* class in order to initialize the data members

(student_id,student_fname,student_lname,s_grade_Assignment1,s_grade_Assignment2,

s_grade_Assignment3, s_grade_Mid_Term, s_grade_Final_Term) of every object.

e) Add public **Mutator** (**setter**) methods in *Grade* class to modify the values of private members.

f) Add public **Accessor** (**getter**) methods in *Grade* class to read the values of private members.

g) Add a method called public String toString() in *Grade* class to print the *Publisher* information in the form of "S Id :"+ student_id +"S LName :"+ student_lname + "S FName :" + student_fname + "S Ass1:" + s_grade_Assignment1 + "S Ass2:" + s_grade_Assignment2 + "S Ass3:" + s_grade_Assignment3 + "S MT:" + s_grade_Mid_Term + "S FT:" + s_grade_Final_Term.

h) Add a method within *Grade* class called (Calculate_GradeAverage ()) that calculates and returns the average score for each student according to the following mark distribution:

   1) 40 % for all assignments (*s_grade_Ass1* for Assignment 1), (*s_grade_Ass2* for Assignment 2), (*s_grade_Ass3* for Assignment 3)
   2) 30 % for Mid Term Exam (*s_grade_Mt*)
   3) 30 % for Final Exam (*s_grade_Ft*)

i) You need to *test* the implemented class Grade in the **main program** *TestGradeProject*.

j) Instantiate an array object of Grade class type to be referenced by (all_sgrades) of Grade class type) using the **default constructor**. Then read records from the input file Grade.in as shown in Figure 4 and assign the instance data of every *record* of the input File into its corresponding array object component using the implemented **setter** methods.

k) Invoke the method Calculate_GradeAverage () related to every array object component, and display the student information using the **getter** methods, the grade average of every student, and the total grade average of all students as shown in the following Figure.
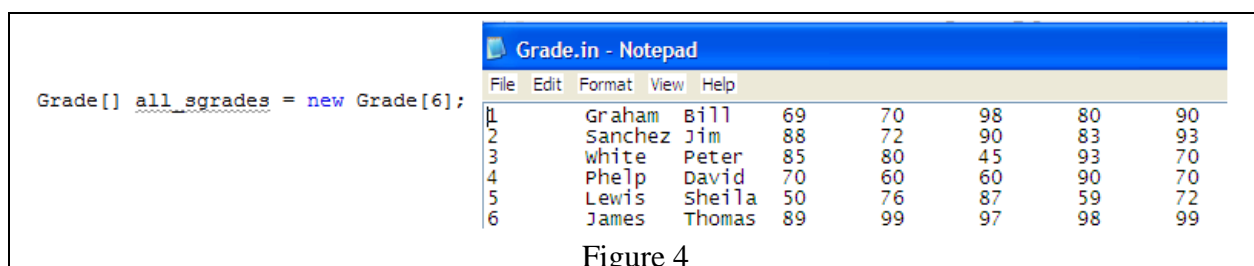

Figure 4

l) (**Parallel Arrays**) Add more Java statements in the main program to do the following:
   - Store the first column of the file *Grade.in* which represents the field `student_id` into *one-dimensional array* of *int* type, and print its components.
   - Store the second and third columns of the file *Grade.in* which represents the fields `student_lname (string)`, `student_fname (string)` into *parallel one-dimensional array* of *String* type each, and print its components.
   - Store all column grades `s_grade_Assignment1`, `s_grade_Assignment2`, `s_grade_Assignment3`, `s_grade_Mid_Term`, `s_grade_Final_Term` of the file *Grade.in* into *two dimensional array* of *double* type each, and print its components.

4. **Using ArrayList Data Structure in ArrayListBillingProject**
   Create a Java Project **ArrayListBillingProject** using NetBeans IDE that allows end user to issue a billing by populating data file Billing.in into an **ArrayList** data structure of type Billing class type.

a) You need to design a **Java class** called `Billing`, which takes the price and quantity as two **private** non static members called respectively (`prd_Price`) and (`prd_Qty`). The variables called `Fed_Tax`, `Prv_Tax` as **public** and static data members.
   1) Add **default constructor**, setters, getters, and toString()
   2) Add a method called `CalculateBilling()` in `Billing` class to calculate the total of billing
      `T_Billing = (prd_Price* prd_Qty) + (prd_Price*prd_Qty)* Fed_Tax + (prd_Price*prd_Qty)* Prv_Tax`

b) Create *TestArrayListBilling.java* where you populate an ArrayList data structure of Billing class type to be referenced by (billingArrList) from input file Billing.in. Set every component using the implemented setter methods (setPrd_Price(), setPrd_Qty()).

Calculate the total of billing of all ArrayList components. Display the total as shown above.