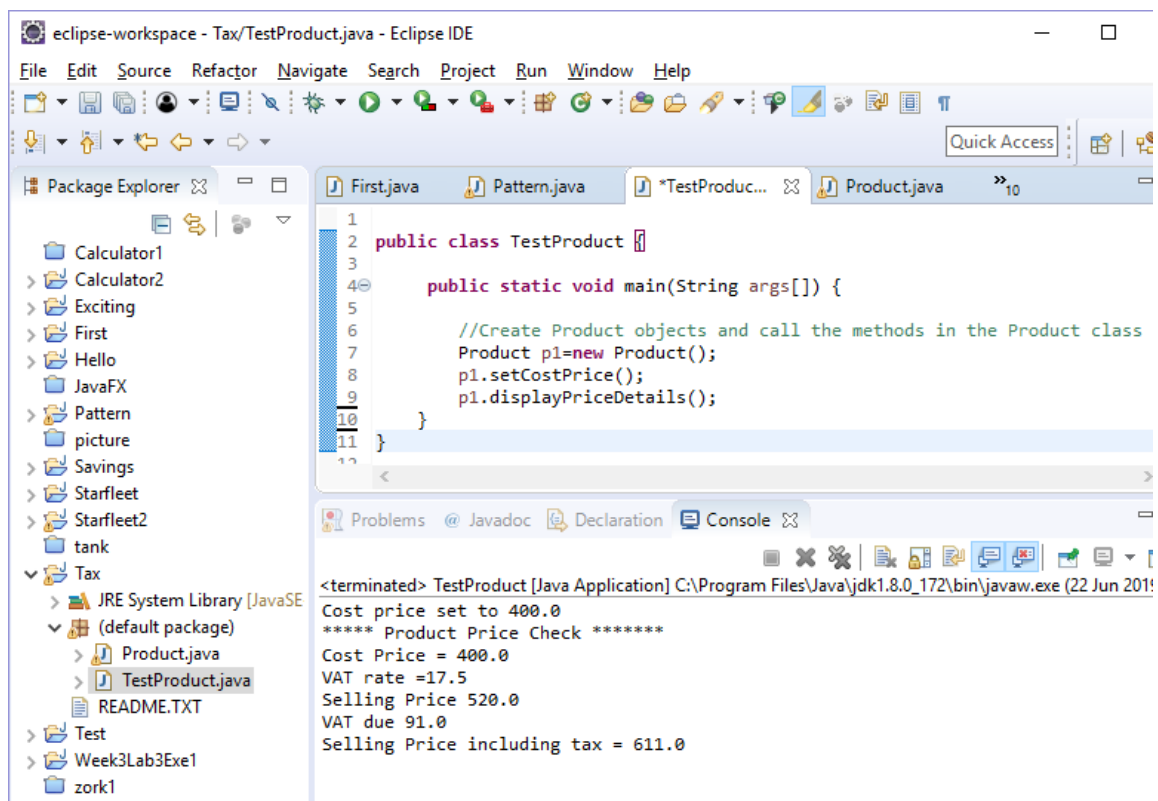


Week 2 In-Lesson Tasks Feedback

2.1.1 - Writing Methods

Task: Open the Tax project, in your Week 2, Lab 1 folder. Within the TestProduct class, create an object and test the **setCostPrice()** and **displayPriceDetails()** methods.

Here it is in the snapshot below:



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project named 'Tax' with a sub-package 'Tax' containing 'Product.java' and 'TestProduct.java'. The main editor window shows the 'TestProduct.java' file with the following code:

```
1 public class TestProduct {
2
3     public static void main(String args[]) {
4
5         //Create Product objects and call the methods in the Product class
6         Product p1=new Product();
7         p1.setCostPrice();
8         p1.displayPriceDetails();
9     }
10 }
11
```

The Console window at the bottom shows the output of the program:

```
<terminated> TestProduct [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (22 Jun 2019)
Cost price set to 400.0
***** Product Price Check *****
Cost Price = 400.0
VAT rate =17.5
Selling Price 520.0
VAT due 91.0
Selling Price including tax = 611.0
```

2.1.2 - Methods Can Return Values

Task: Open the Tax project and test the above method within the TestProduct class.

Here it is as in the snapshot below.

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project named 'Tax' with a sub-package 'JRE System Library [JavaSE]' and a 'default package' containing 'Product.java', 'TestProduct.java', and 'README.TXT'. The main editor window shows the 'TestProduct.java' file with the following code:

```
1
2 public class TestProduct {
3
4     public static void main(String args[]) {
5
6         //Create Product objects and call the methods in the Product class
7         Product p1=new Product();
8         p1.setCostPrice();
9         p1.displayPriceDetails();
10
11         //2.1.2 task
12         Product p2=new Product();
13         p2.displayPriceDetails();
14     }
15 }
16
```

The Console window at the bottom shows the output of the program:

```
<terminated> TestProduct [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (22 Jun 2019)
Cost price set to 400.0
***** Product Price Check *****
Cost Price = 400.0
VAT rate =17.5
Selling Price 520.0
VAT due 91.0
Selling Price including tax = 611.0
***** Product Price Check *****
Cost Price = 0.0
VAT rate =17.5
Selling Price 0.0
VAT due 0.0
Selling Price including tax = 0.0
```

2.1.3 - Passing Information to Methods

Task: Open the picture project and review the moveLeft, moveRight, moveVertical and moveHorizontal method in the Circle class. Take a particular look at the methods with parameters.

Here are the code extracts below. Please note that the moveVertical and moveHorizontal methods each takes a parameter. Within the moveRight and moveLeft method, the value 20 and -20 are passed to the moveHorizontal method.

```
/**
 * Move the circle a few pixels to the right.
 */
public void moveRight()
{
    moveHorizontal(20);
}

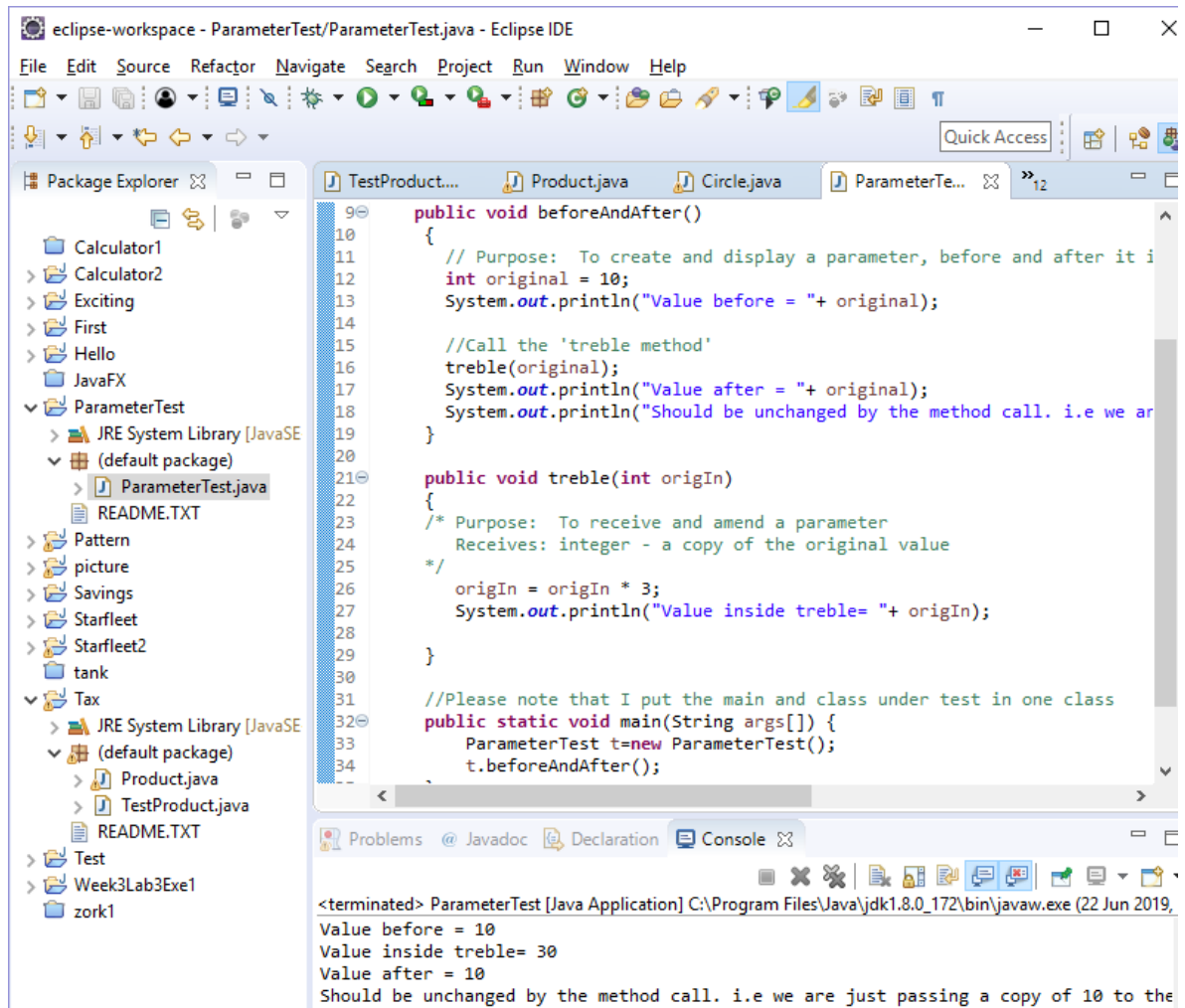
/**
 * Move the circle a few pixels to the left.
 */
public void moveLeft()
{
    moveHorizontal(-20);
}

/**
 * Move the circle horizontally by 'distance' pixels.
 */
public void moveHorizontal(int distance)
{
    erase();
    xPosition += distance;
    draw();
}

/**
 * Move the circle vertically by 'distance' pixels.
 */
public void moveVertical(int distance)
{
    erase();
    yPosition += distance;
    draw();
}
```

Task: Watch the video below and then run the beforeAndAfter method yourself by going to your Week 2, Lab 1 folder and finding the ParameterTest project folder.

The snapshot below illustrates this.

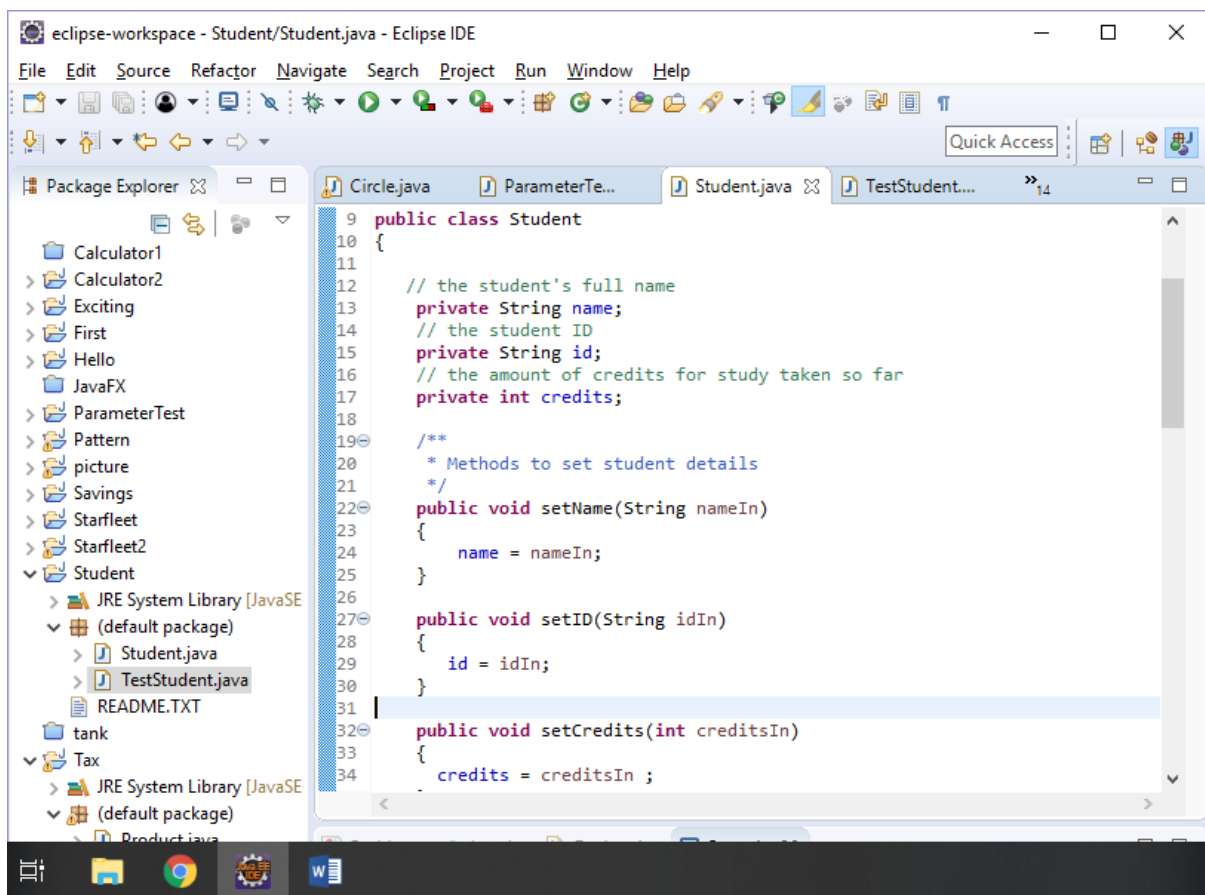


2.1.4 - Using Strings

Task: Study the student project.

Notice how we pass **Strings as parameters** in the 'setName(String nameIn)' method, to set the students' name. As you can see the method specifies one formal parameter - nameIn.

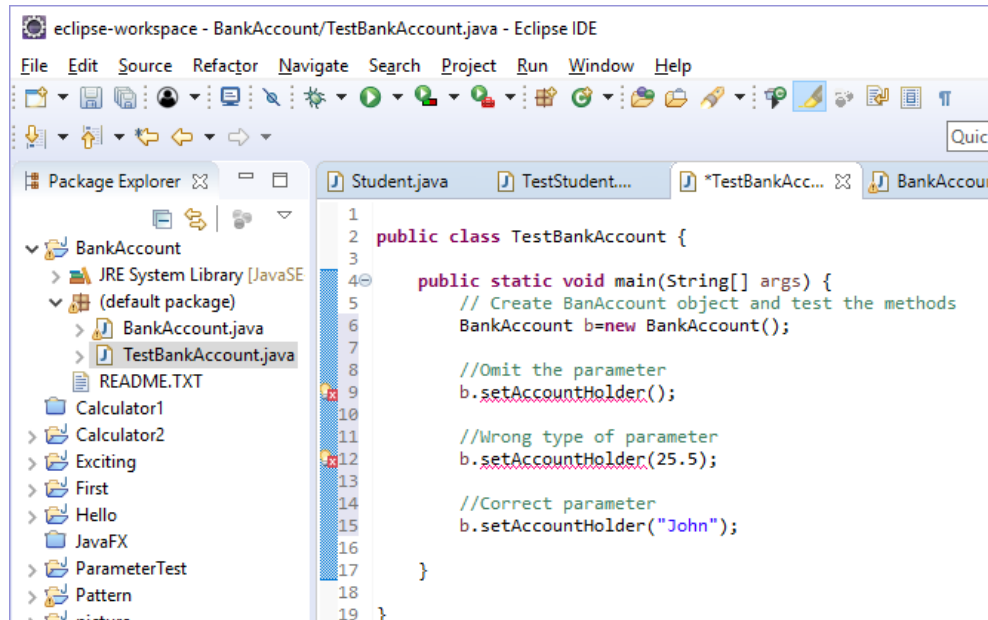
The snapshot below demonstrates the use of String *nameIn* and *idIn* as parameter. Their values are passed onto *name* and *id* respectively.



```
eclipse-workspace - Student/Student.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
Calculator1
Calculator2
Exciting
First
Hello
JavaFX
ParameterTest
Pattern
picture
Savings
Starfleet
Starfleet2
Student
  JRE System Library [JavaSE]
  (default package)
    Student.java
    TestStudent.java
    README.TXT
  tank
Tax
  JRE System Library [JavaSE]
  (default package)
  Product.java
Circle.java
ParameterTe...
Student.java
TestStudent...
14
9 public class Student
10 {
11
12     // the student's full name
13     private String name;
14     // the student ID
15     private String id;
16     // the amount of credits for study taken so far
17     private int credits;
18
19     /**
20      * Methods to set student details
21      */
22     public void setName(String nameIn)
23     {
24         name = nameIn;
25     }
26
27     public void setID(String idIn)
28     {
29         id = idIn;
30     }
31
32     public void setCredits(int creditsIn)
33     {
34         credits = creditsIn ;
35     }
36 }
```

Task: Open the BankAccount project, within the TestBankAccount class, pass incorrect parameters to methods and view errors.

Here are some examples in the snapshot below. Incorrect ones are highlighted in red.



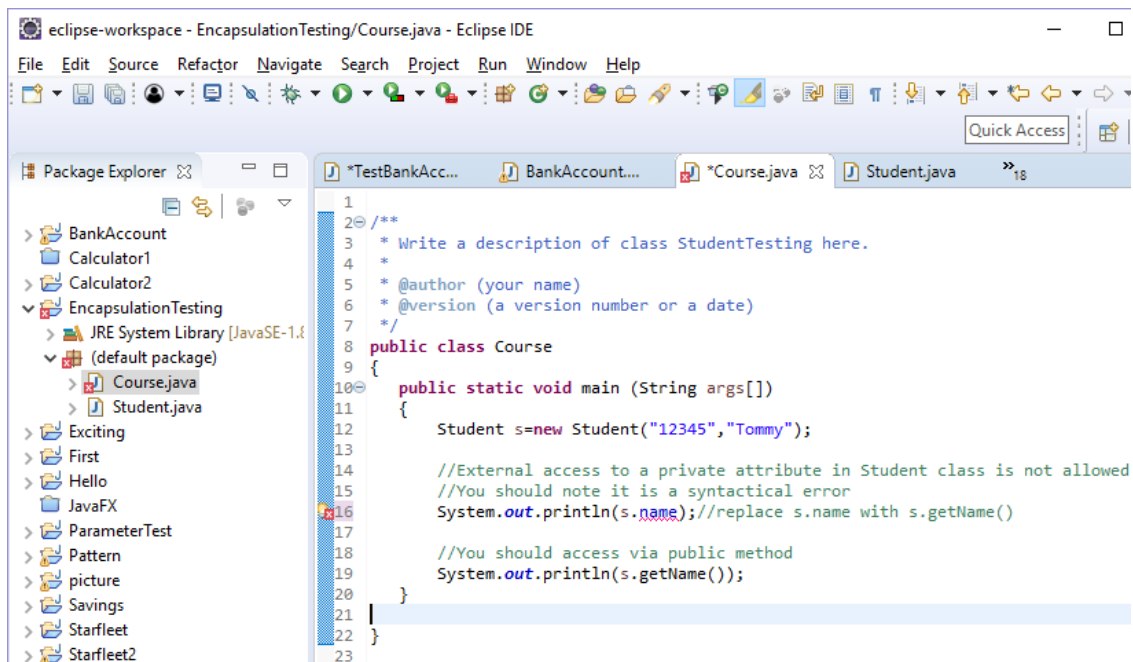
The screenshot shows the Eclipse IDE interface. The Package Explorer on the left shows the project structure with 'BankAccount' containing 'TestBankAccount.java'. The main editor displays the code for 'TestBankAccount.java'. The code includes a 'main' method that creates a 'BankAccount' object and calls 'setAccountHolder' with three different parameters: no parameter, a double value (25.5), and a string value ('John'). The first two calls are marked with red error icons and red squiggly lines, while the third call is correct.

```
1
2 public class TestBankAccount {
3
4     public static void main(String[] args) {
5         // Create BanAccount object and test the methods
6         BankAccount b=new BankAccount();
7
8         //Omit the parameter
9         b.setAccountHolder();
10
11        //Wrong type of parameter
12        b.setAccountHolder(25.5);
13
14        //Correct parameter
15        b.setAccountHolder("John");
16
17    }
18
19 }
```

2.2.2 - Encapsulation

Task: Open the EncapsulationTesting project. Investigate and correct the syntax error in the Course class.

Here it is the snapshot that illustrates this.



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure, with 'EncapsulationTesting' expanded to show 'Course.java' and 'Student.java'. The main editor window shows the code for 'Course.java'. The code includes a class declaration, a main method, and a call to 'Student.getName()'. A syntax error is highlighted on line 16, where 's.name' is used instead of 's.getName()'. The error message states: '//External access to a private attribute in Student class is not allowed //You should note it is a syntactical error //You should access via public method'. The code is as follows:

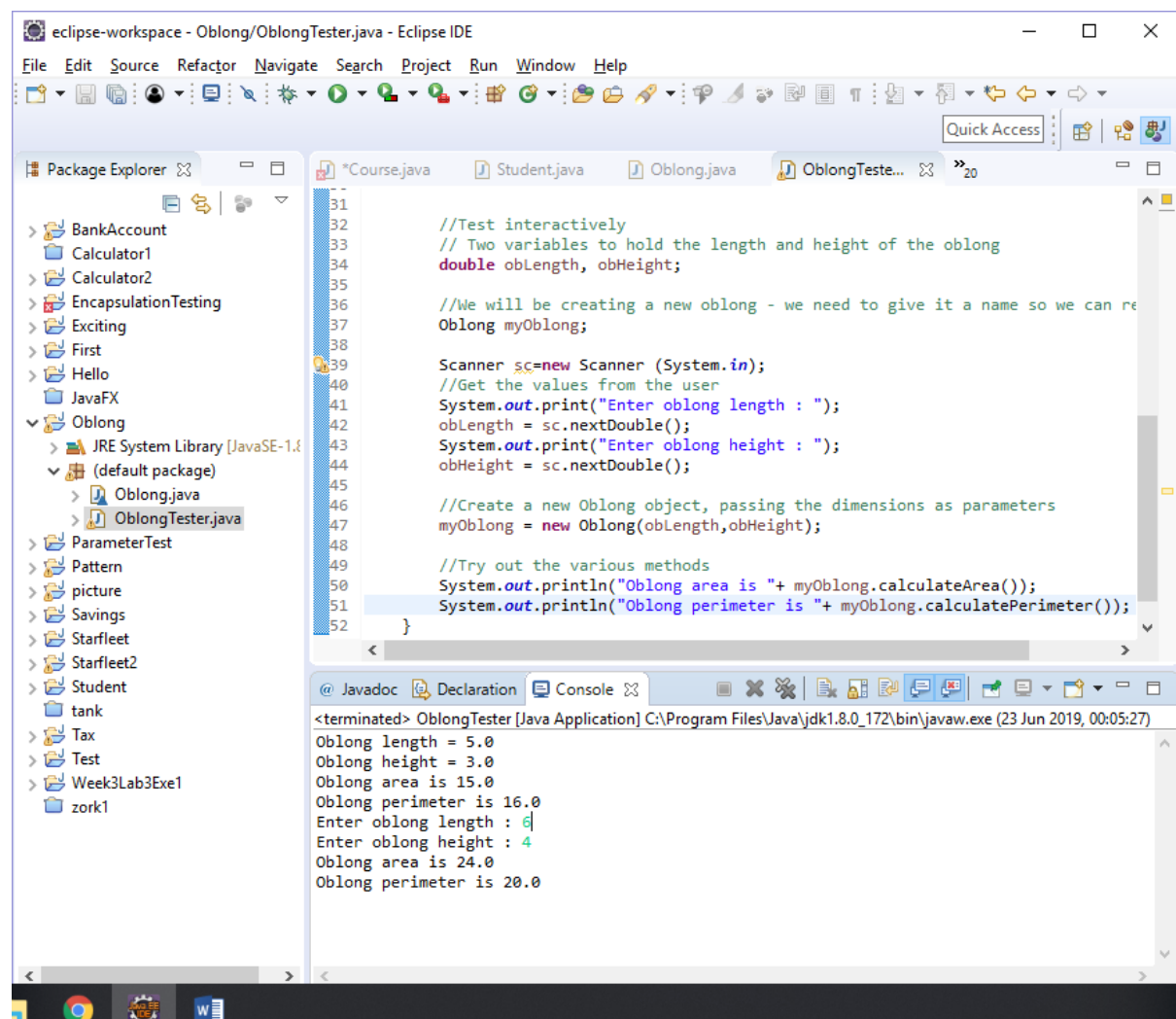
```
1
2 /**
3  * Write a description of class StudentTesting here.
4  *
5  * @author (your name)
6  * @version (a version number or a date)
7  */
8 public class Course
9 {
10     public static void main (String args[])
11     {
12         Student s=new Student("12345","Tommy");
13
14         //External access to a private attribute in Student class is not allowed
15         //You should note it is a syntactical error
16         System.out.println(s.name); //replace s.name with s.getName()
17
18         //You should access via public method
19         System.out.println(s.getName());
20     }
21 }
22
23
```

2.2.3 - Classes as data types

Task:

Open the Oblong project in Eclipse. Review the code in *Oblong* and *OblongTester* class. Run the project and interact with the program by entering the *length* and *height*, then observing the output. Try to match the code with the program behaviour.

Please see the snapshot below. Code line 26-29 produce the first four lines of the output. Code line 41 onwards produce the next four lines of the output.



The screenshot shows the Eclipse IDE with the `OblongTester.java` file open. The code in the editor is as follows:

```
31
32 //Test interactively
33 // Two variables to hold the length and height of the oblong
34 double obLength, obHeight;
35
36 //We will be creating a new oblong - we need to give it a name so we can re
37 Oblong myOblong;
38
39 Scanner sc=new Scanner (System.in);
40 //Get the values from the user
41 System.out.print("Enter oblong length : ");
42 obLength = sc.nextDouble();
43 System.out.print("Enter oblong height : ");
44 obHeight = sc.nextDouble();
45
46 //Create a new Oblong object, passing the dimensions as parameters
47 myOblong = new Oblong(obLength,obHeight);
48
49 //Try out the various methods
50 System.out.println("Oblong area is "+ myOblong.calculateArea());
51 System.out.println("Oblong perimeter is "+ myOblong.calculatePerimeter());
52 }
```

The console output at the bottom shows the program's execution:

```
<terminated> OblongTester [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (23 Jun 2019, 00:05:27)
Oblong length = 5.0
Oblong height = 3.0
Oblong area is 15.0
Oblong perimeter is 16.0
Enter oblong length : 6
Enter oblong height : 4
Oblong area is 24.0
Oblong perimeter is 20.0
```


2.2.4 - The String Class

Task: Open the String1 project. Review the code in the *StringTest* class and make a note of the String methods that have been invoked. Run and interact with the program.

The following String methods have been invoked:

String constructor

length

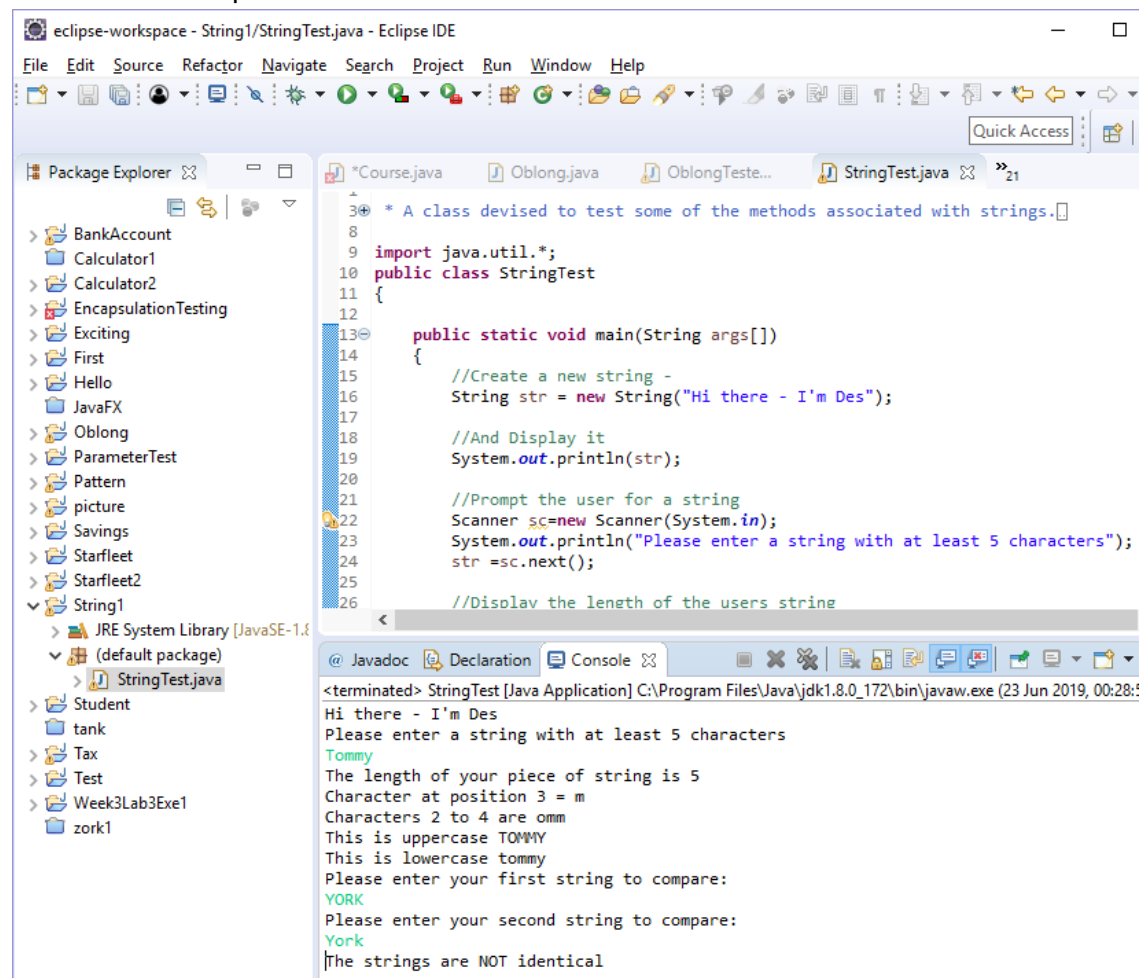
charAt

substring

toUpperCase

toLowerCase

Here it is the snapshot with interactions:



The screenshot shows the Eclipse IDE with the *StringTest.java* file open. The code in the file is as follows:

```
* A class devised to test some of the methods associated with strings.
8
9 import java.util.*;
10 public class StringTest
11 {
12
13     public static void main(String args[])
14     {
15         //Create a new string -
16         String str = new String("Hi there - I'm Des");
17
18         //And Display it
19         System.out.println(str);
20
21         //Prompt the user for a string
22         Scanner sc=new Scanner(System.in);
23         System.out.println("Please enter a string with at least 5 characters");
24         str =sc.next();
25
26         //Dislay the length of the users string
```

The console output shows the following interactions:

```
<terminated> StringTest [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (23 Jun 2019, 00:28:
Hi there - I'm Des
Please enter a string with at least 5 characters
Tommy
The length of your piece of string is 5
Character at position 3 = m
Characters 2 to 4 are omm
This is uppercase TOMMY
This is lowercase tommy
Please enter your first string to compare:
YORK
Please enter your second string to compare:
York
The strings are NOT identical
```

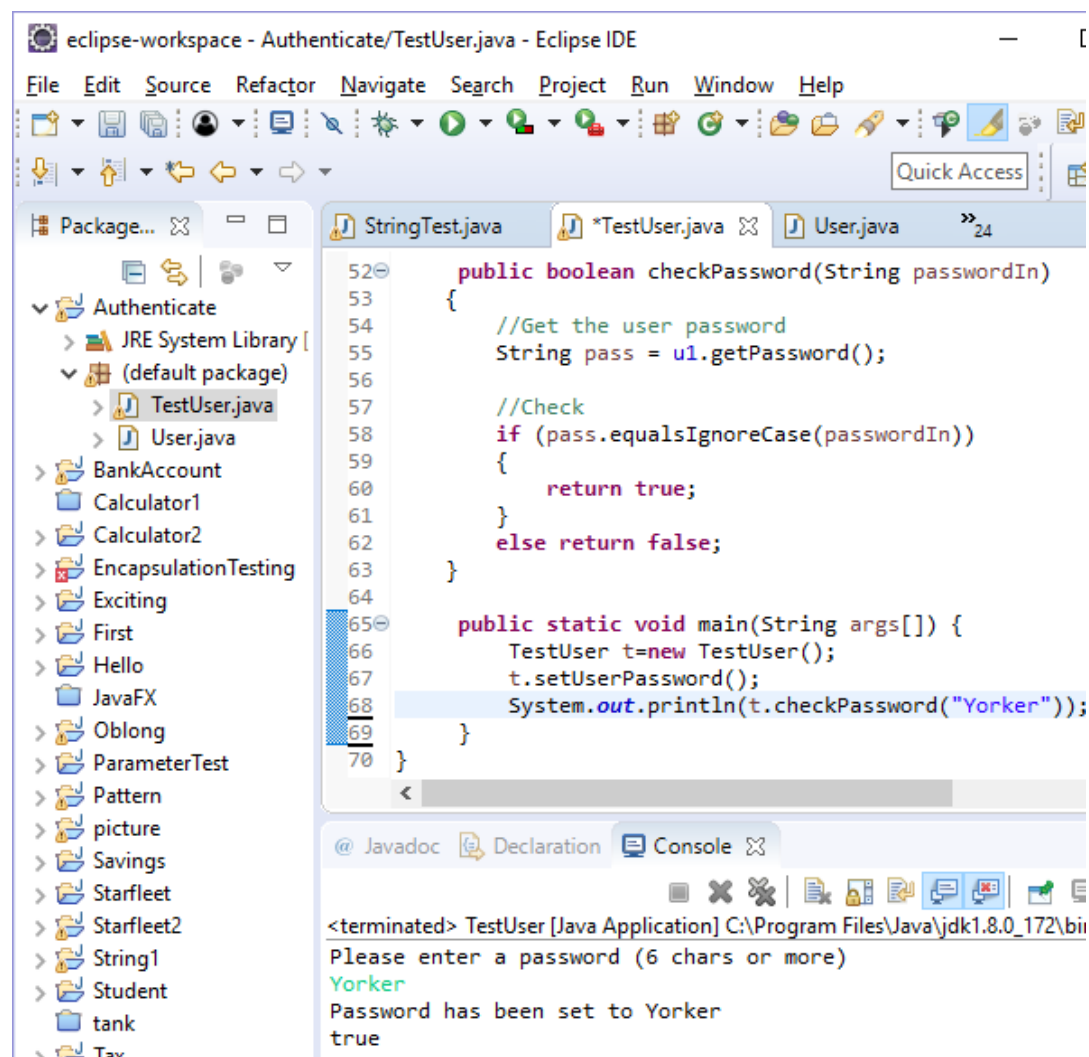
2.2.5 - Objects as Parameters

Task: Open the Authenticate project and review the code in the *User* and *TestUser* class. Make a note of the parameters for *setPassword* and *checkPassword* method. Run and interact with the program by entering at least a bad and a good password.

Parameters:

- setPassword(String passwordIn)
- checkPassword(String passwordIn)

Here is the snapshot for a good password.

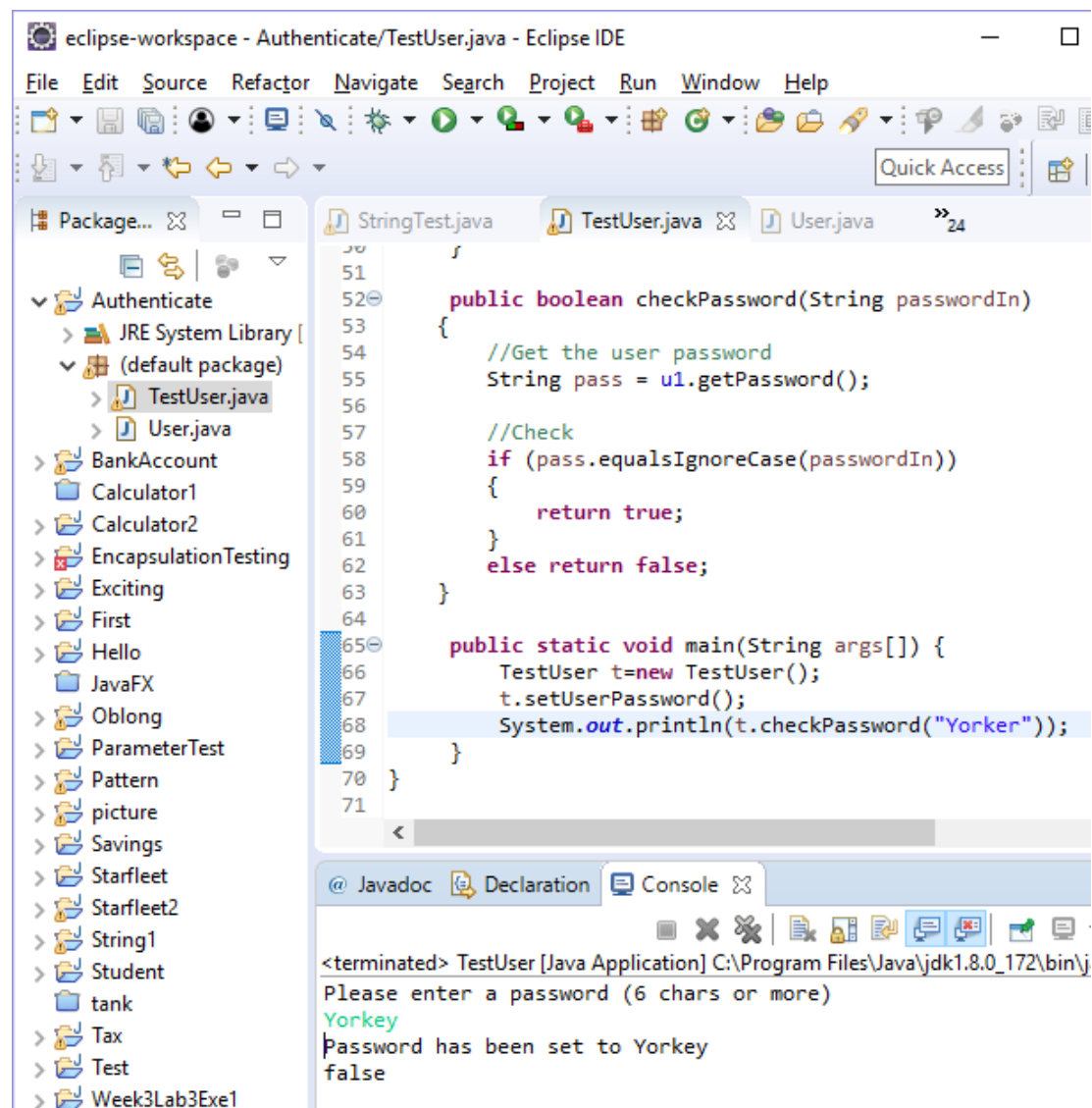


```
eclipse-workspace - Authenticate/TestUser.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package... StringTest.java *TestUser.java User.java »24
Authenticate
  JRE System Library
  (default package)
    TestUser.java
    User.java
  BankAccount
  Calculator1
  Calculator2
  EncapsulationTesting
  Exciting
  First
  Hello
  JavaFX
  Oblong
  ParameterTest
  Pattern
  picture
  Savings
  Starfleet
  Starfleet2
  String1
  Student
  tank
  Tav

52 public boolean checkPassword(String passwordIn)
53 {
54     //Get the user password
55     String pass = u1.getPassword();
56
57     //Check
58     if (pass.equalsIgnoreCase(passwordIn))
59     {
60         return true;
61     }
62     else return false;
63 }
64
65 public static void main(String args[]) {
66     TestUser t=new TestUser();
67     t.setUserPassword();
68     System.out.println(t.checkPassword("Yorker"));
69 }
70 }

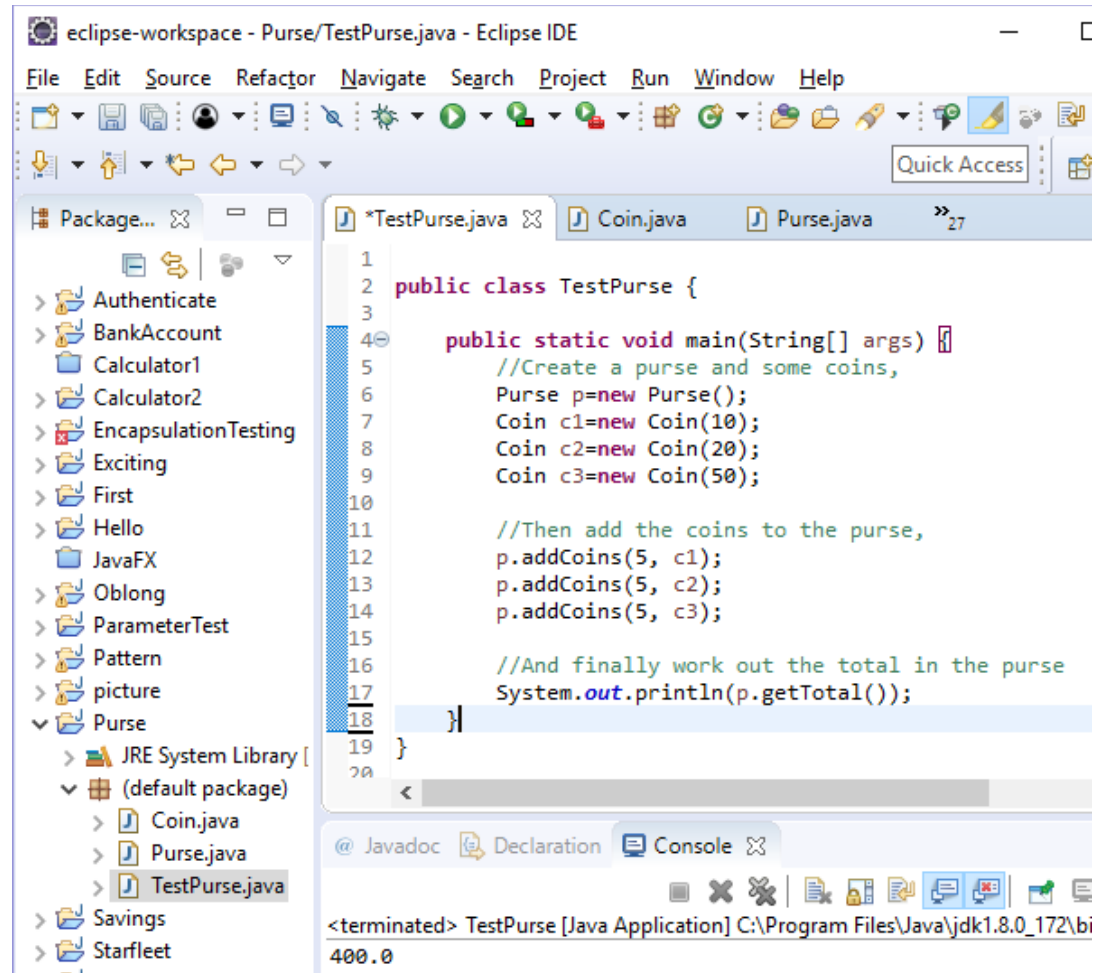
@ Javadoc Declaration Console
<terminated> TestUser [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin
Please enter a password (6 chars or more)
Yorker
Password has been set to Yorker
true
```

And here is the snapshot for a bad password.



Task: Investigate the Purse project. Use the TestPurse class to create some coins, and add them to the purse. Then find the total in the purse.

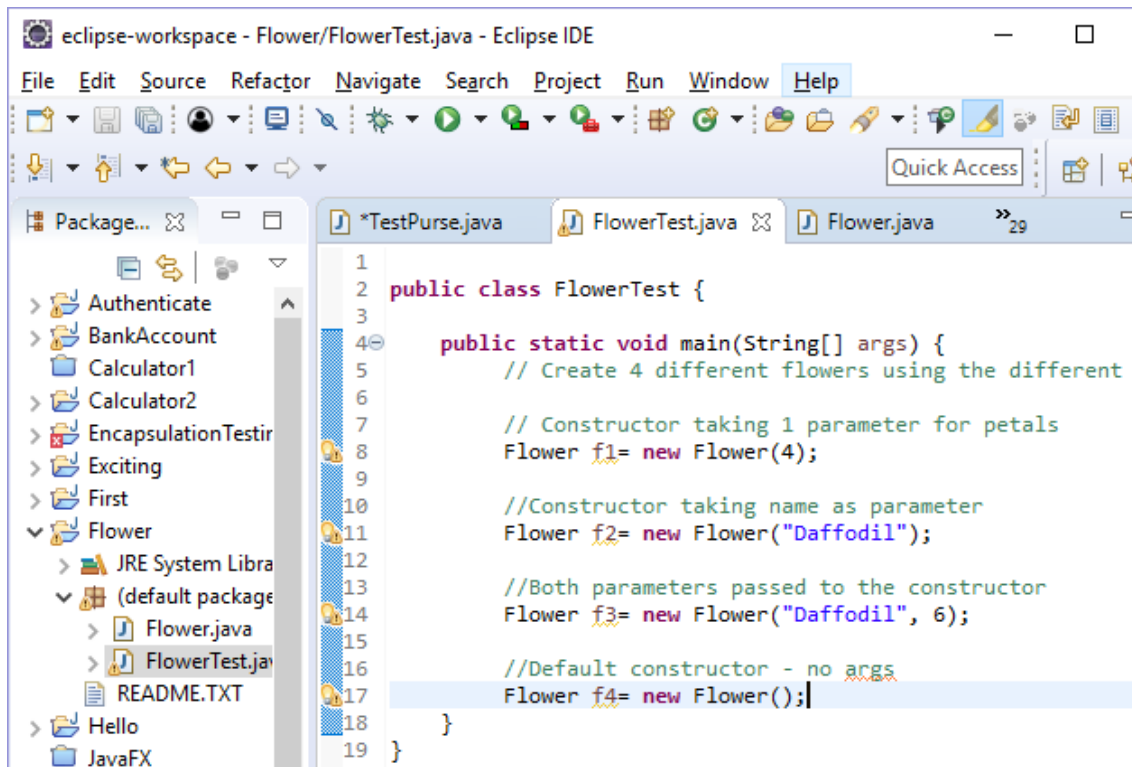
The snapshot below illustrates this.



2.3.1 - The Oblong Class

Task: Investigate *Flower* Project with its constructors. Within *FlowerTest* class, create 4 different flowers using the different constructors.

Here it is as in the snapshot below:



```
1 public class FlowerTest {
2
3     public static void main(String[] args) {
4         // Create 4 different flowers using the different
5
6         // Constructor taking 1 parameter for petals
7         Flower f1= new Flower(4);
8
9
10        //Constructor taking name as parameter
11        Flower f2= new Flower("Daffodil");
12
13        //Both parameters passed to the constructor
14        Flower f3= new Flower("Daffodil", 6);
15
16        //Default constructor - no args
17        Flower f4= new Flower();
18    }
19 }
```

2.3.2 - More Class Diagram to Implementation Examples

Task: Inspect the code for the BankAccount class within BankAccount1 project, and identify where the class attributes are defined. Notice accountNumber and accountName are defined as Strings. As you know, Strings are objects in their own right. ***It is perfectly normal for the attributes of one class to be the objects of another class.***

```
// the class attributes  
private String accountNumber;  
private String accountName;  
private double balance;
```

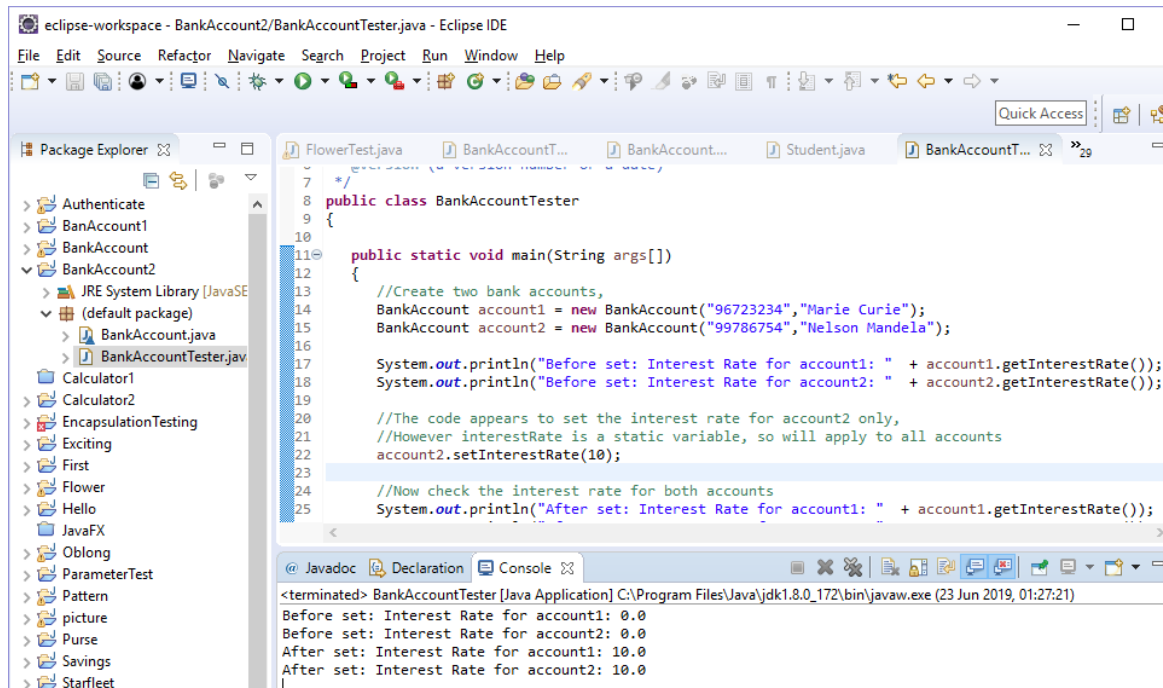
Task: View the Student Class and match the code to the class diagram.

Yes, the code exactly matches the class diagram with three attributes, two constructors, two mutator and four accessor methods.

2.3.3 - The Static Keyword

Task: Open the bankAccount2 project, view the BankaccountTester code, and run it. Notice the change of the interestRate of one object, applies to others as well.

Here it is in the snapshot below. Please note line 22 sets account2 interest to 10 and this applies to account1.



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure, including the BankAccount2 package. The main editor window shows the BankAccountTester.java file with the following code:

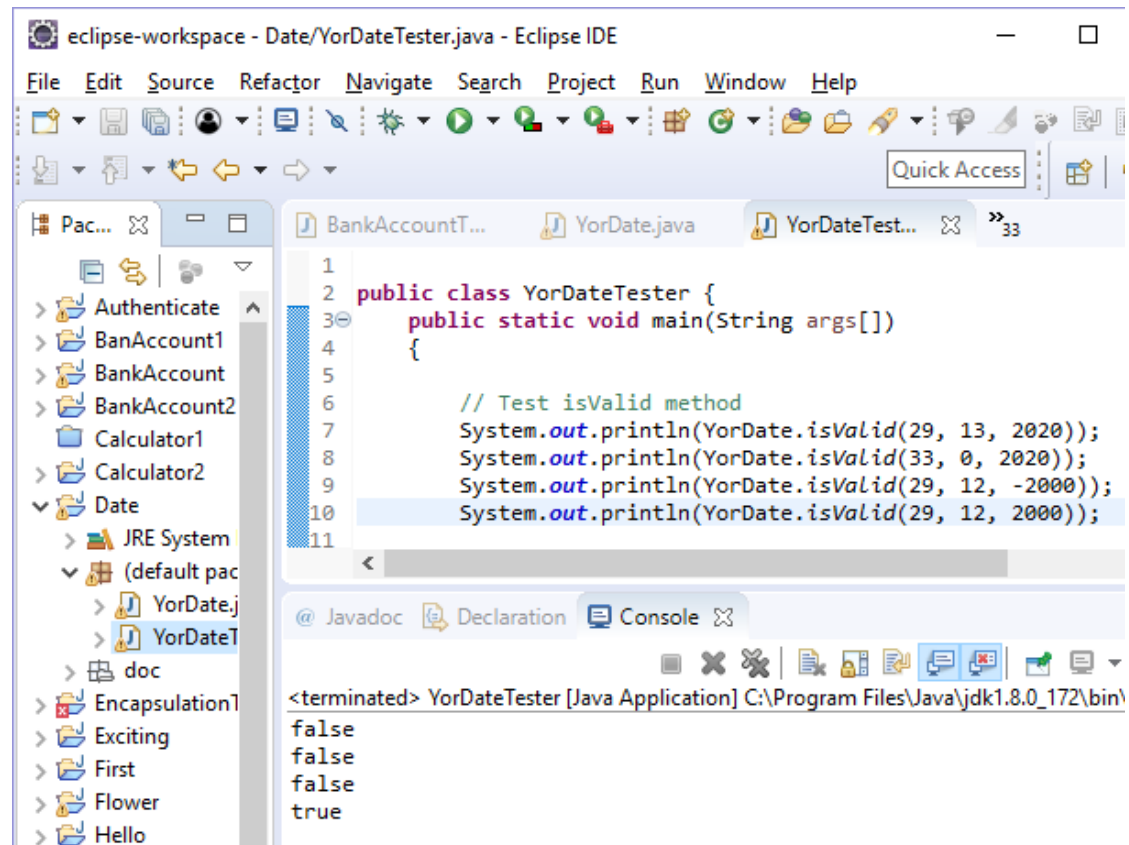
```
7  /*
8  public class BankAccountTester
9  {
10
11     public static void main(String args[])
12     {
13         //Create two bank accounts,
14         BankAccount account1 = new BankAccount("96723234","Marie Curie");
15         BankAccount account2 = new BankAccount("99786754","Nelson Mandela");
16
17         System.out.println("Before set: Interest Rate for account1: " + account1.getInterestRate());
18         System.out.println("Before set: Interest Rate for account2: " + account2.getInterestRate());
19
20         //The code appears to set the interest rate for account2 only,
21         //However interestRate is a static variable, so will apply to all accounts
22         account2.setInterestRate(10);
23
24         //Now check the interest rate for both accounts
25         System.out.println("After set: Interest Rate for account1: " + account1.getInterestRate());
```

The Console window at the bottom shows the output of the program:

```
<terminated> BankAccountTester [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (23 Jun 2019, 01:27:21)
Before set: Interest Rate for account1: 0.0
Before set: Interest Rate for account2: 0.0
After set: Interest Rate for account1: 10.0
After set: Interest Rate for account2: 10.0
```

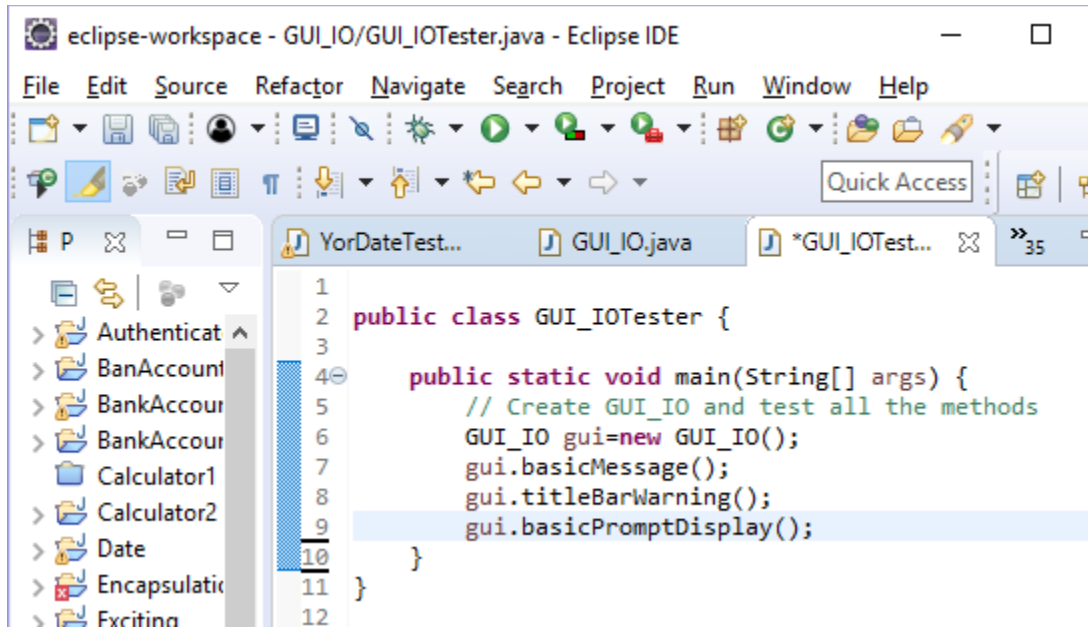
Task: Open the Date project and investigate the isValid static method in the YorDate class. Within the YorDateTester class, create and run a few more tests for the isValid method.

Here it is the snapshot below:



Task: Open the GUI_IO project, within the GUI_IOTester class and create a GUI_IO object. Test all the methods.

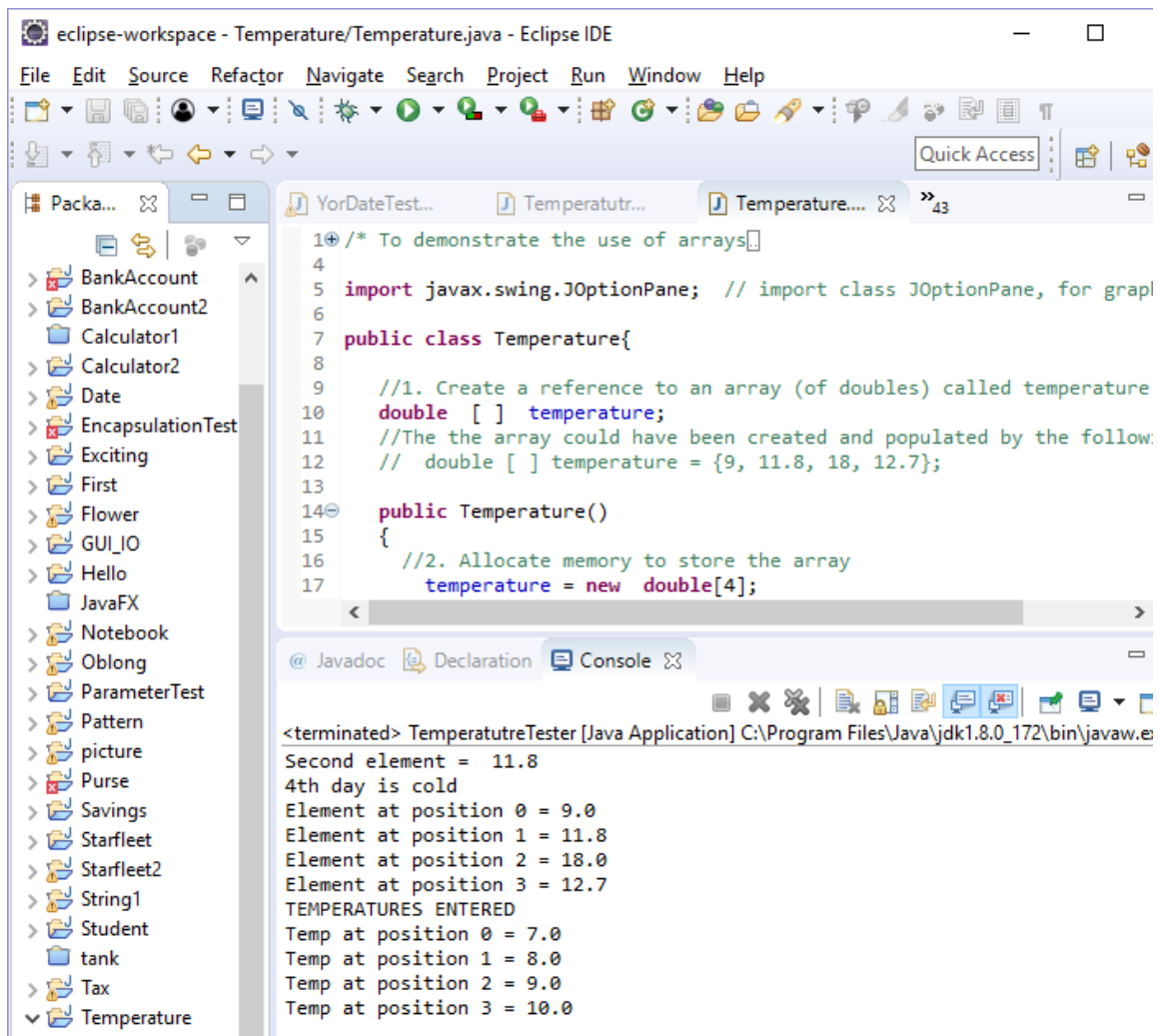
Here it is in the snapshot below:



2.4.1 - Fixed-Size Collections - Arrays

Task: Investigate the **Temperature** project, notice the various uses of arrays. Run and interact with the program, match the output with the program code.

After entering four temperatures 7, 8, 9 and 10 via JOptionPane, I get this:



The screenshot shows the Eclipse IDE with the `Temperature.java` file open. The code defines a `Temperature` class with a `double` array and a constructor that initializes it. The console output shows the results of running the program, including the temperatures entered and the corresponding array elements.

```
1 /* To demonstrate the use of arrays */
2
3
4
5 import javax.swing.JOptionPane; // import class JOptionPane, for graphi
6
7 public class Temperature{
8
9     //1. Create a reference to an array (of doubles) called temperature
10    double [ ] temperature;
11    //The the array could have been created and populated by the follow:
12    // double [ ] temperature = {9, 11.8, 18, 12.7};
13
14    public Temperature()
15    {
16        //2. Allocate memory to store the array
17        temperature = new double[4];
18    }
19 }
```

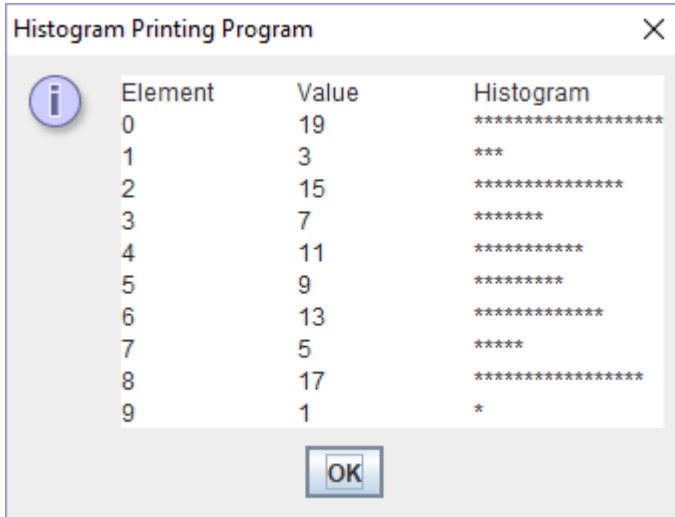
Console Output:

```
<terminated> TemperatutreTester [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.e
Second element = 11.8
4th day is cold
Element at position 0 = 9.0
Element at position 1 = 11.8
Element at position 2 = 18.0
Element at position 3 = 12.7
TEMPERATURES ENTERED
Temp at position 0 = 7.0
Temp at position 1 = 8.0
Temp at position 2 = 9.0
Temp at position 3 = 10.0
```

Task: Investigate the **Histogram** project.

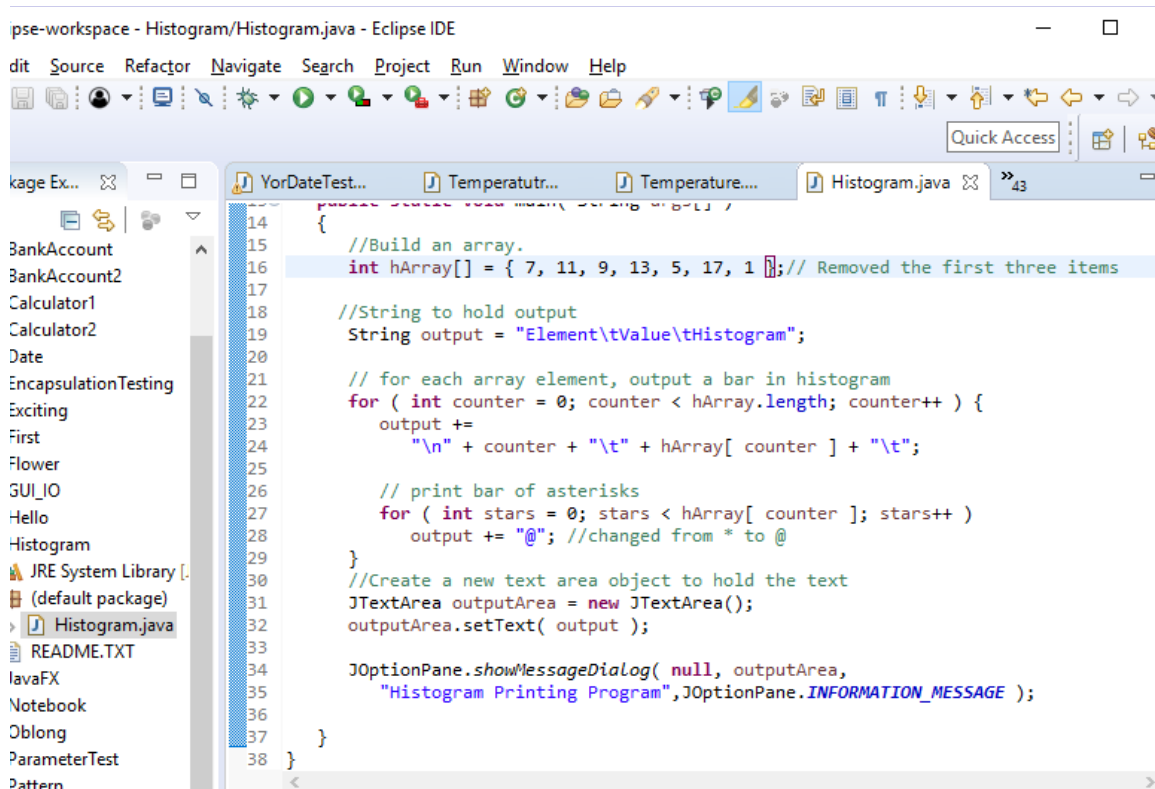
Try changing, adding or removing values in the array; for example changing the symbol from a '*' to another character.

I get this before changes:

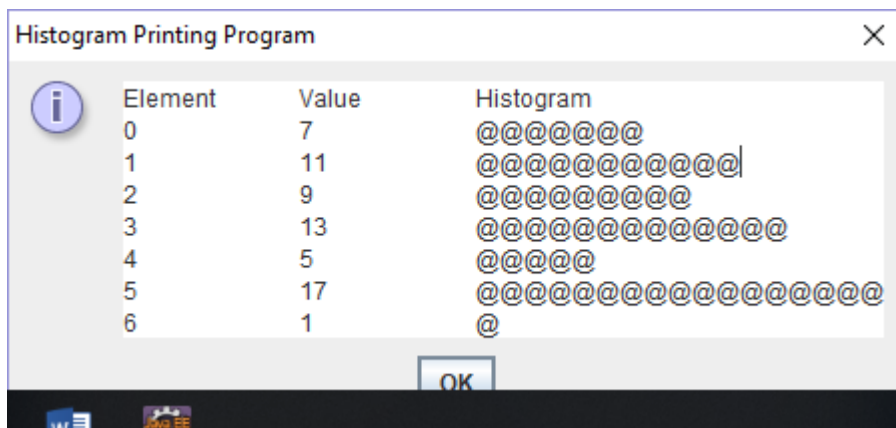


Element	Value	Histogram
0	19	*****
1	3	***
2	15	*****
3	7	*****
4	11	*****
5	9	*****
6	13	*****
7	5	*****
8	17	*****
9	1	*

I get the following snapshot after the changes (removed the first three element and use the character @ instead) made:



```
14 {
15     //Build an array.
16     int hArray[] = { 7, 11, 9, 13, 5, 17, 1 }; // Removed the first three items
17
18     //String to hold output
19     String output = "Element\tValue\tHistogram";
20
21     // for each array element, output a bar in histogram
22     for ( int counter = 0; counter < hArray.length; counter++ ) {
23         output +=
24             "\n" + counter + "\t" + hArray[ counter ] + "\t";
25
26         // print bar of asterisks
27         for ( int stars = 0; stars < hArray[ counter ]; stars++ )
28             output += "@"; //changed from * to @
29     }
30     //Create a new text area object to hold the text
31     JTextArea outputArea = new JTextArea();
32     outputArea.setText( output );
33
34     JOptionPane.showMessageDialog( null, outputArea,
35         "Histogram Printing Program",JOptionPane.INFORMATION_MESSAGE );
36
37 }
38 }
```



2.4.3 - Object Structures with Collections

Task: Investigate the Notebook project, store and display a few more notes.

Here it is the snapshot. Please note Wednesday is displayed first as a result of line 13.

