Week 1 In-Lesson Tasks Example Answers

1.2 Lesson 2: Java building blocks

What objects might exist in a Library System?

Book, Journal

Think of an object from your Library System – what information would it need to store and what functions should it carry out?

Book title, ISBN

What different classifications can you think of for a cow or a watch? What are some of the common attributes and functions for these classifications. How and where do we define what information our objects can hold and what they are capable of doing?

They can be classified as Cow or Watch. A cow has a species attribute and they are able to moo. A watch has a brand attribute and they are able to tick.

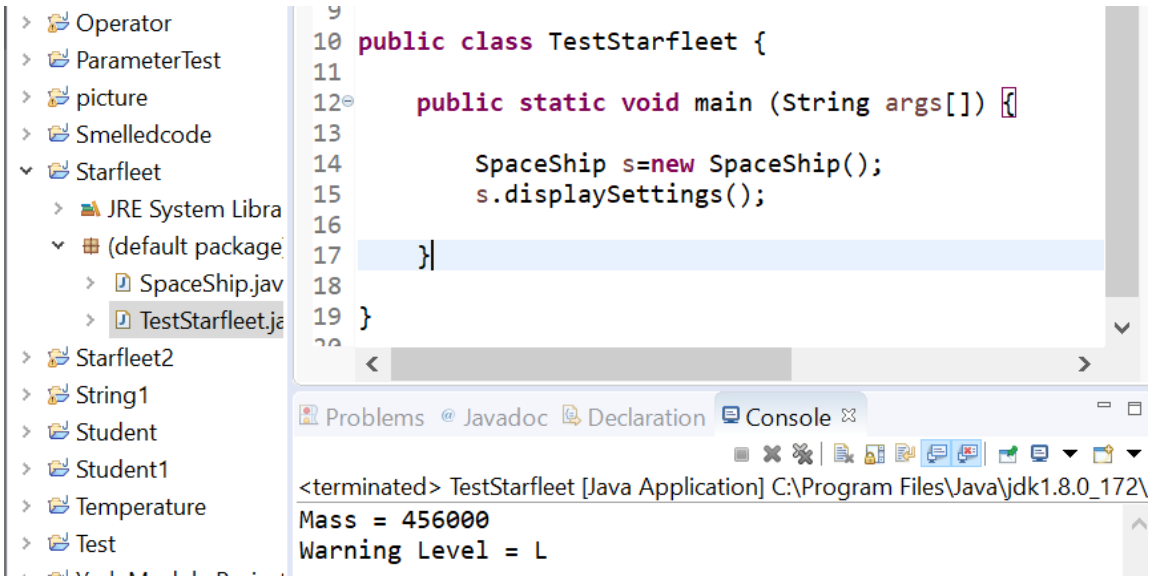
1.2.3: Assigning values to variables

Task: Have a go at declaring and initialising a variable for a high school Grade.

char grade= ‘A’;

Task: Open the Starfleet project in Eclipse and run the TestSpaceship class and ‘inspect’ the values of the displayed variables. In the test class creates a Starfleet object and invokes the displaySettings () method. **Tip:** Omitting the semicolon at the end of a statement is a syntax error. A syntax error is when the compiler cannot recognise the statement.

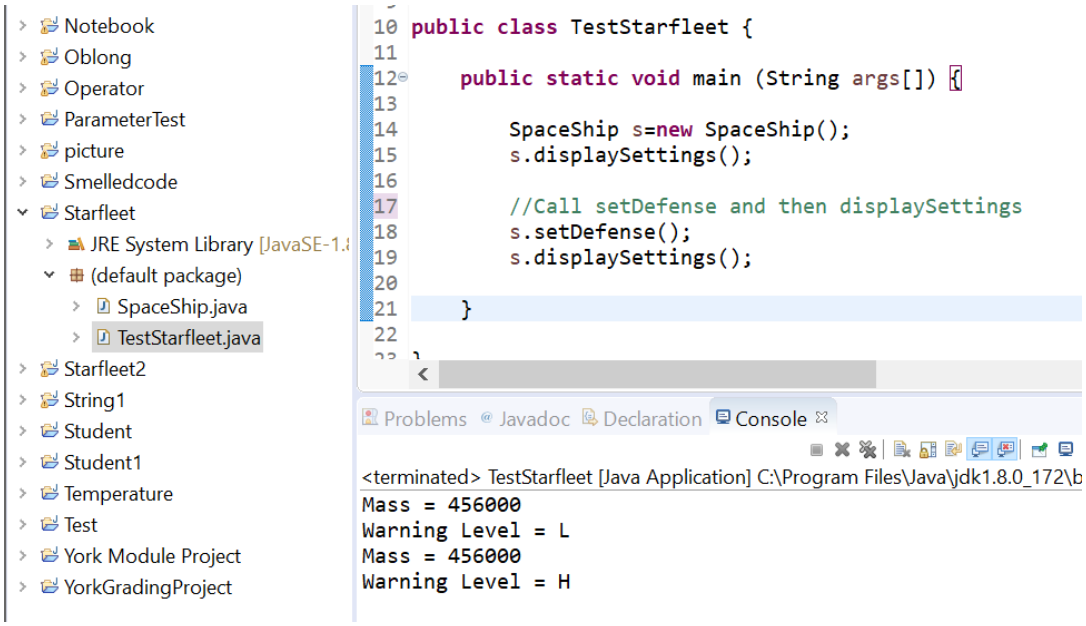
The snapshot below illustrates this.



1.2.4: Writing programming instructions

Task: Using Eclipse, within the TestStarfleet class in the Starfleet project, call the 'setDefense()' method and then invoke the displaySettings () method to inspect the values again. **Tip:** Whenever you type a left brace { in your program, immediately type the closing right brace }. This helps prevent errors due to missing braces.

Here it is in the snapshot below. Please note that the value for Warning Level is changed due to the call to the setDefense method.



1.2.5: Arithmetic operators

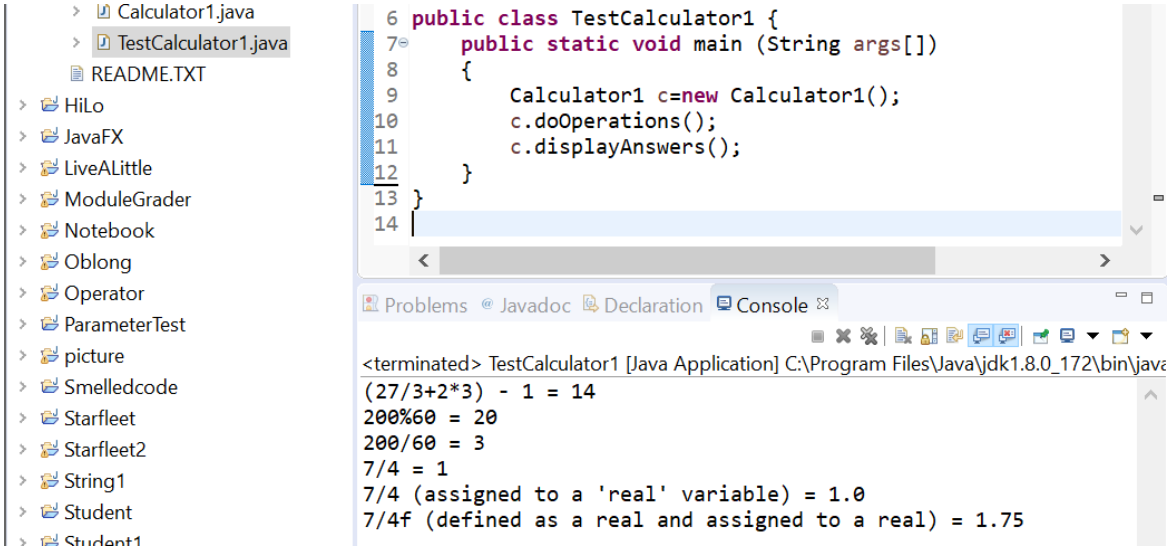
What are the answers to the following?

- 27/3+2*3
- 27/(3+2*3)
- (27/3) + (2*3)

- 27/3+2*3=15
- 27/(3+2*3)=3
- (27/3) + (2*3)=15

Task – Open the **Calculator1** project in Eclipse. Try and calculate the answers manually before performing 'doOperations()'. Run the TestCalulator1 class and check whether the outputs match your calculation.

Here it is the snapshot for running the program.

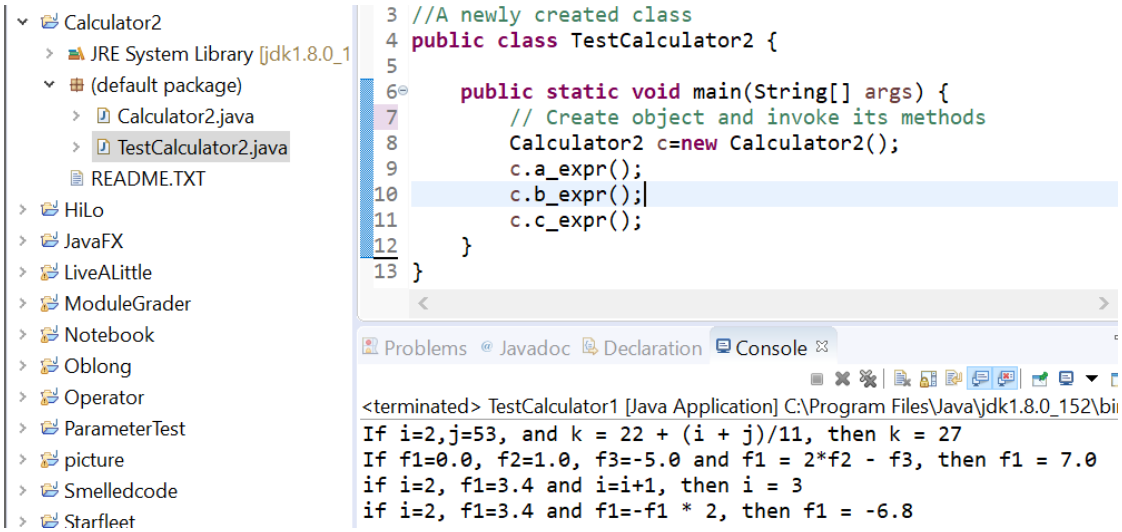


1.2.6: Expressions

Task: Open Calculator2 project in Eclipse and carry out the following:

- first calculate the results you would expect
- next create TestCalculator2 class (similar to TestCalculator1 in Calculator1 project)
- then create a 'Calculator2' object and check your answers by invoking the method a_expr(), followed by b_expr() and c_expr().

The snapshot below illustrates this.



1.2.7: Boolean operators and expressions

```
boolean logicalAnswer;
int a=5;
int b=7;
boolean solvent;
double income=10000;
double outgoings=13000;
```

Task: Using the above values, what is the value of **logicalAnswer** after each of the following statements?

```
logicalAnswer=(a<b);
logicalAnswer=(b>=7);
logicalAnswer=(a==b);
solvent=(outgoings<income);
```

logicalAnswer=(a<b);	TRUE
logicalAnswer=(b>=7);	TRUE
logicalAnswer=(a==b);	FALSE
solvent=(outgoings<income);	FALSE

Task: Using the above values find logicalAnswer:

1. logicalAnswer = ((a < b) || (b < 5))
2. logicalAnswer = ((b < 6) || true)
3. logicalAnswer = ((true) || (a > b) || (b == a))

1. logicalAnswer = ((a < b) || (b < 5))	TRUE
2. logicalAnswer = ((b < 6) || true)	TRUE
3. logicalAnswer = ((true) || (a > b) || (b == a))	TRUE

Task: Using the above initial values again (a = 5, b = 7), calculate the contents of logicalAnswer after the following expressions.

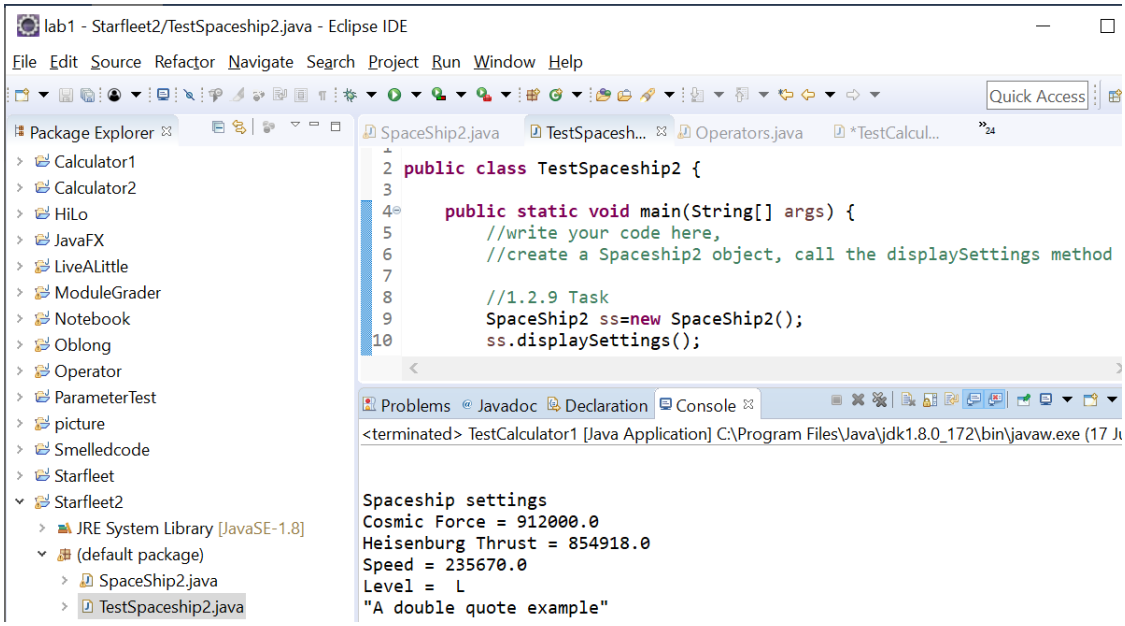
1. logicalAnswer = ! (false);
2. logicalAnswer = ! (a < b);

1. logicalAnswer = ! (false);	TRUE
2. logicalAnswer = ! (a < b);	FALSE

1.2.9: Output in Java

Task: Open the Starfleet2 project to Eclipse. Inspect the ‘displaySettings()’ method in the Spaceship2 class. Within TestSpaceship2 class, create a Spaceship2 object, then invoke the displaySettings() method.

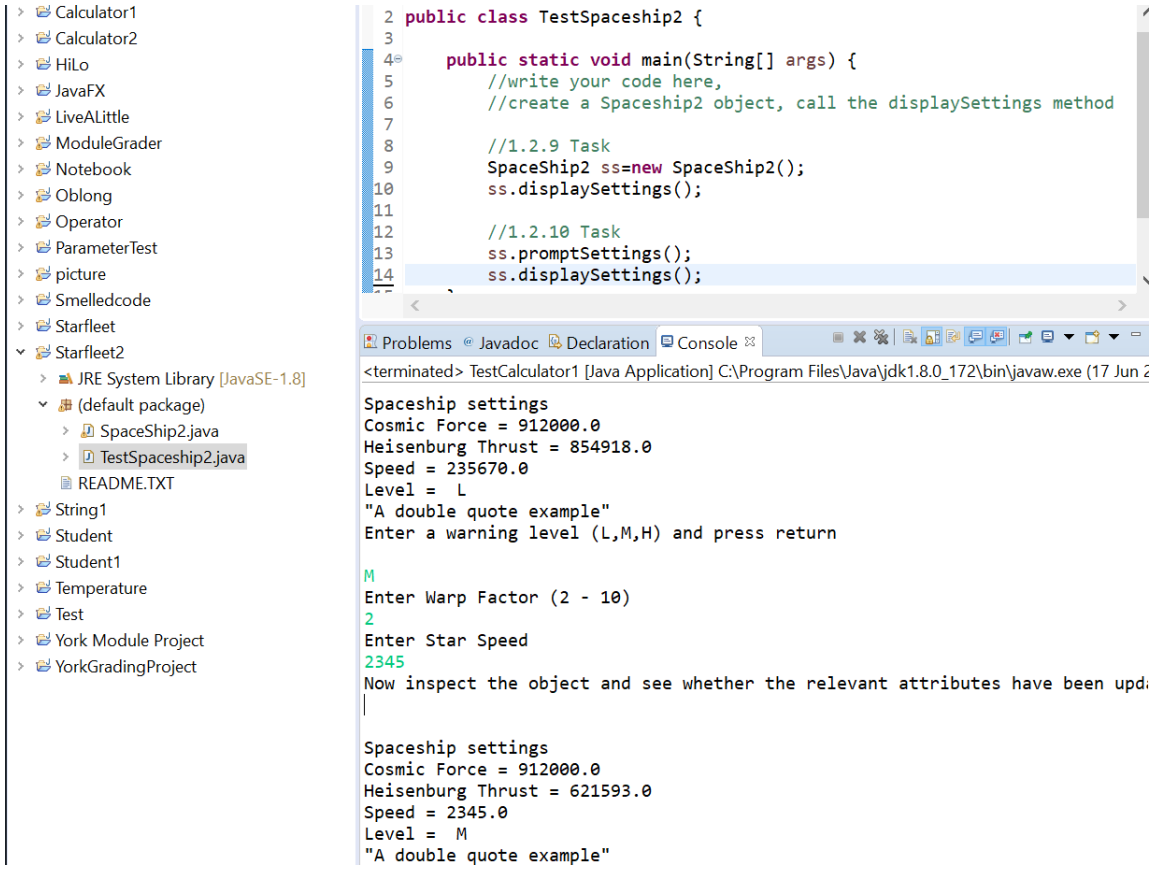
The snap shot below illustrates this.



1.2.10: Input in Java

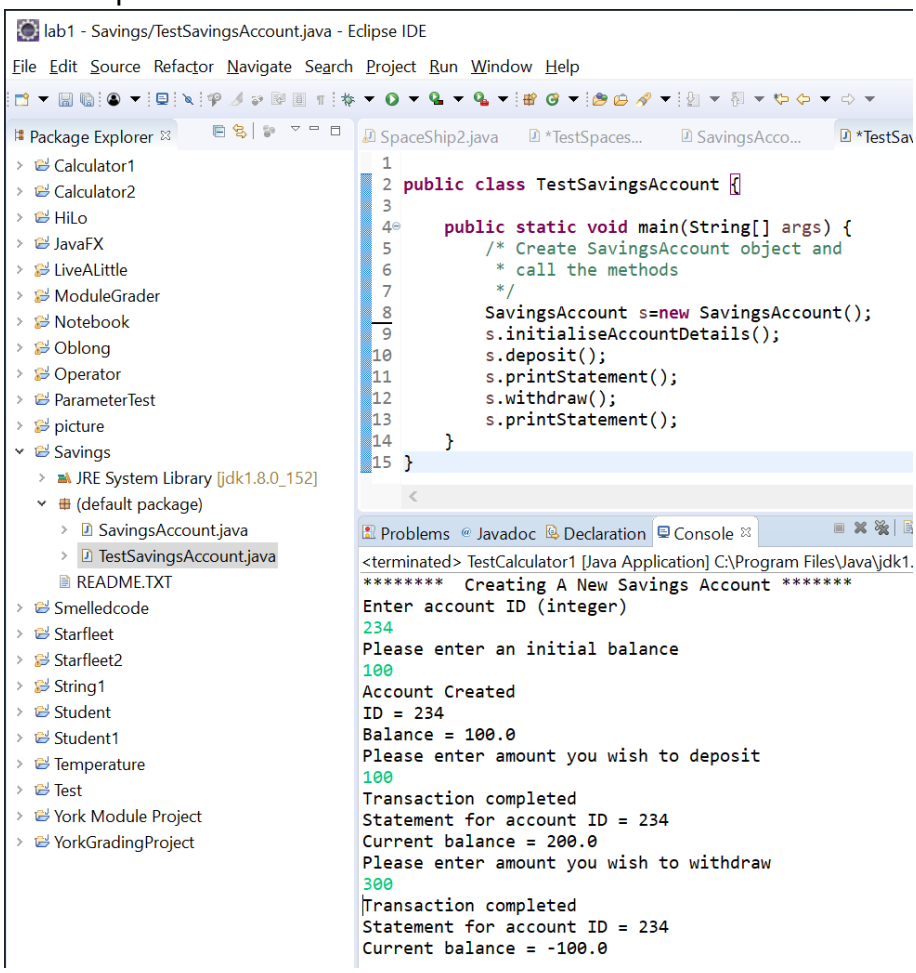
Task: Continue with the previous task, and invoke the above method. Now inspect the object by invoking the displaySettings() method again, and see whether the relevant attributes have been updated.

The snapshot below illustrates this. Please note the values for Warning Level and Speed have been updated.



Task: Open the Savings project in Eclipse. Within the SavingsAccount class, inspect the code in the methods with a particular attention on the use of scanner input. Within the TestSavingsAccount class, create a SavingsAccount object, then call some of the methods e.g. initialise account, deposit, print statement, withdraw and then print statement again.

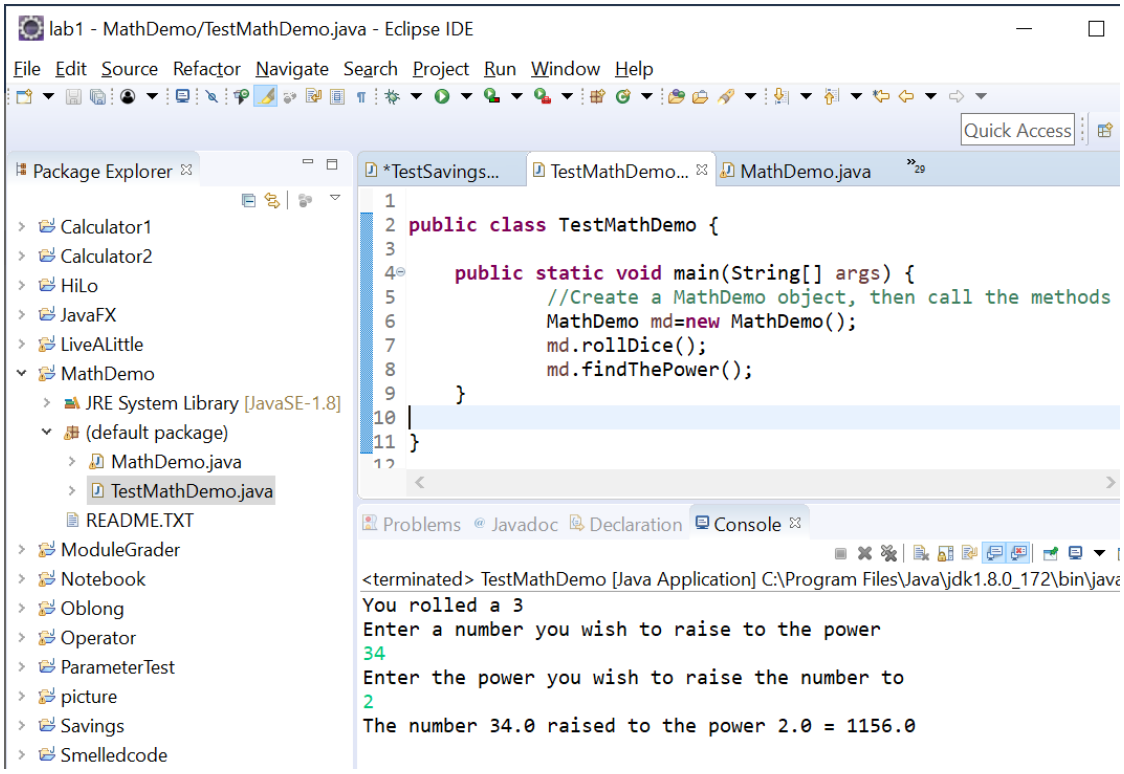
The snapshot below illustrate this.



1.2.11: Using prepackaged methods

Task: Open the MathDemo project in Eclipse, within the TestMathsDemo class, create a MathsDemo object then invoke the the rollDice and findThePower method. Run and interact with the program.

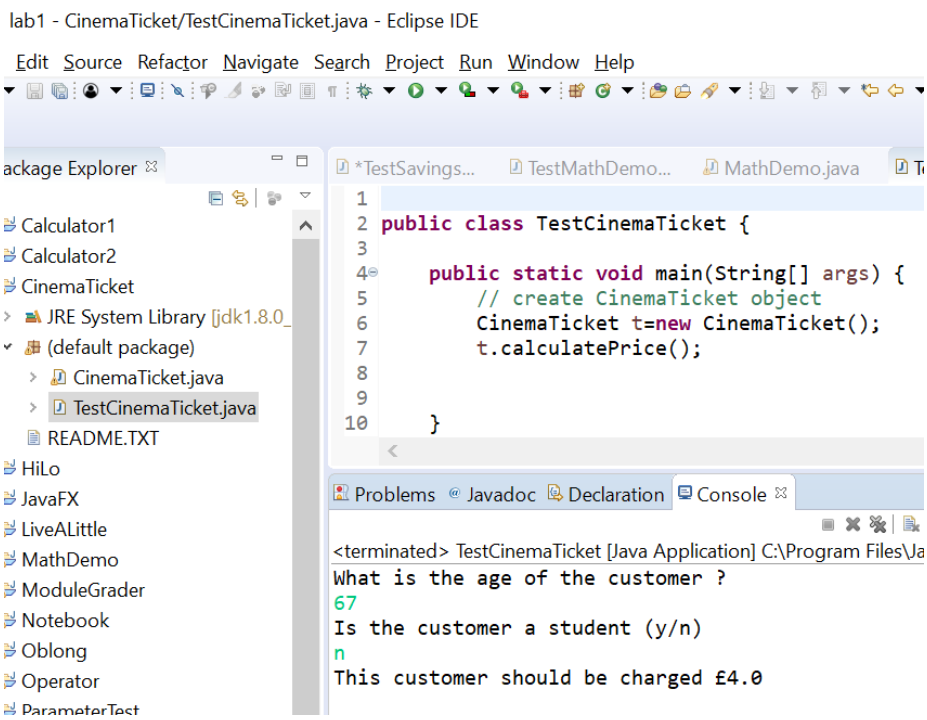
The snapshot below illustrates this.



1.3 Lesson 3: Selection

Task: Open the CinemaTicket Project in Eclipse. Inspect the if-else statements in the CinemaTicket class. Create a TestCinemaTicket class with a main method (as the TestHilo class in the Hilo project). Within the main method, create a CinemaTicket object then invoke the calculatePrice method. Run your TestCinemaTicket class then interact with the program by entering your age and whether you are a student as prompted.

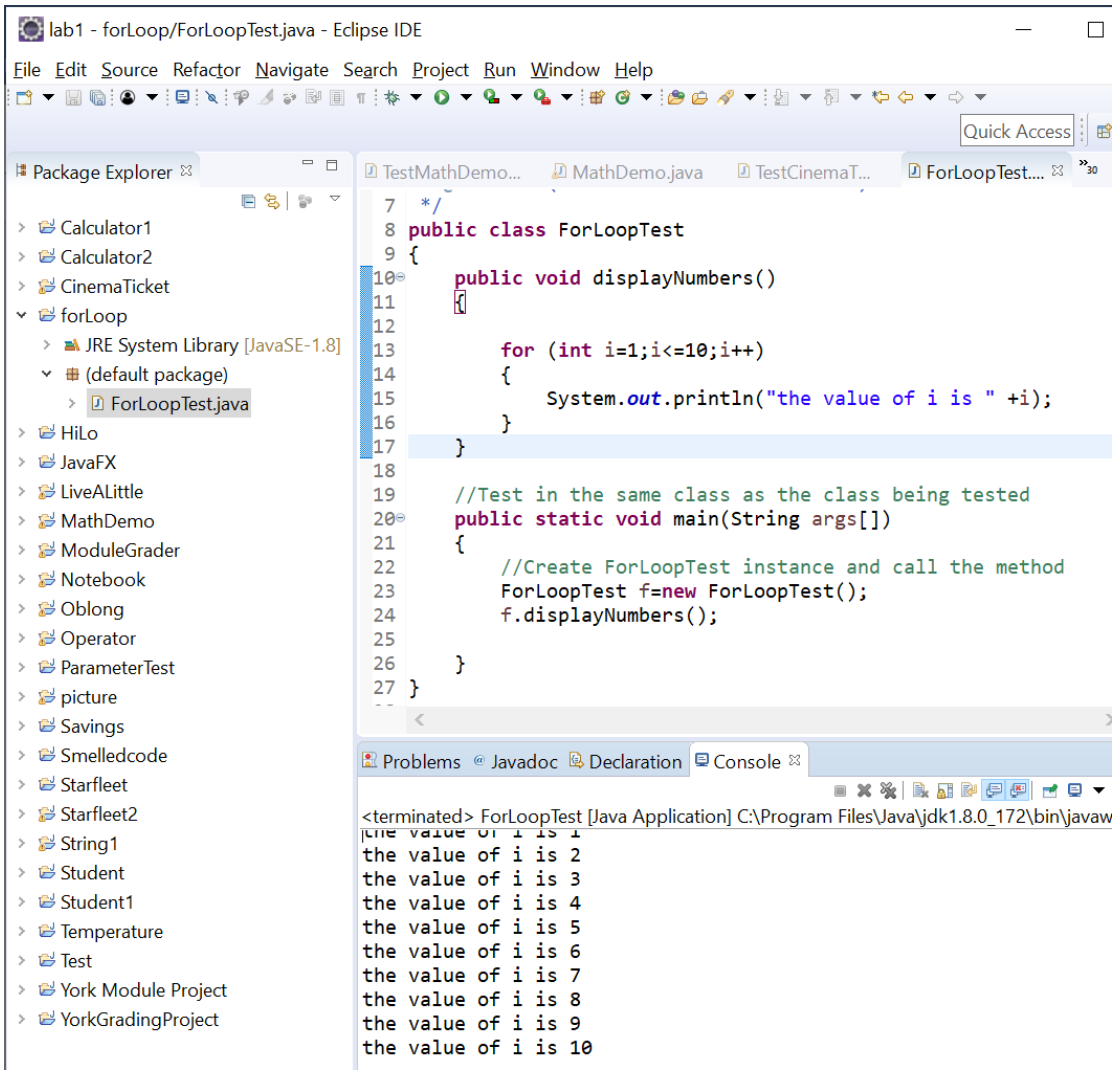
The snapshot below illustrate this.



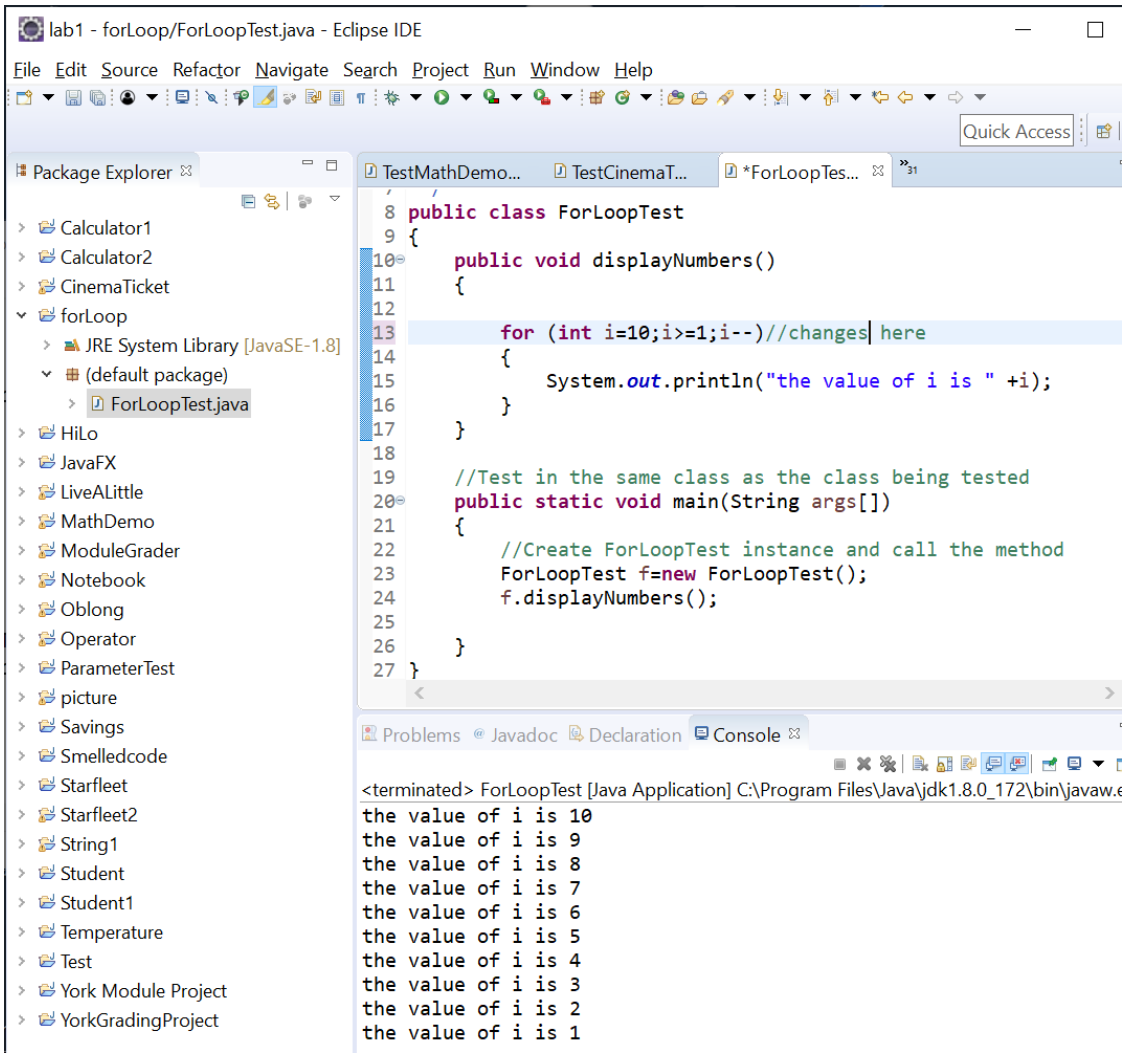
1.4 Lesson 4: Iteration

Task: The code can be found in the folder forLoop and is called ForLoopTest. Open up the project, you can see that an instance of the ForLoopTest class has been created and the method displayNumbers() called. Run the program and record the output. Modify the loop above so it prints out the statements in reverse order. For example, the value is 9, the value is 8, the value is 7, etc.

The snapshot below shows the execution of the original program.

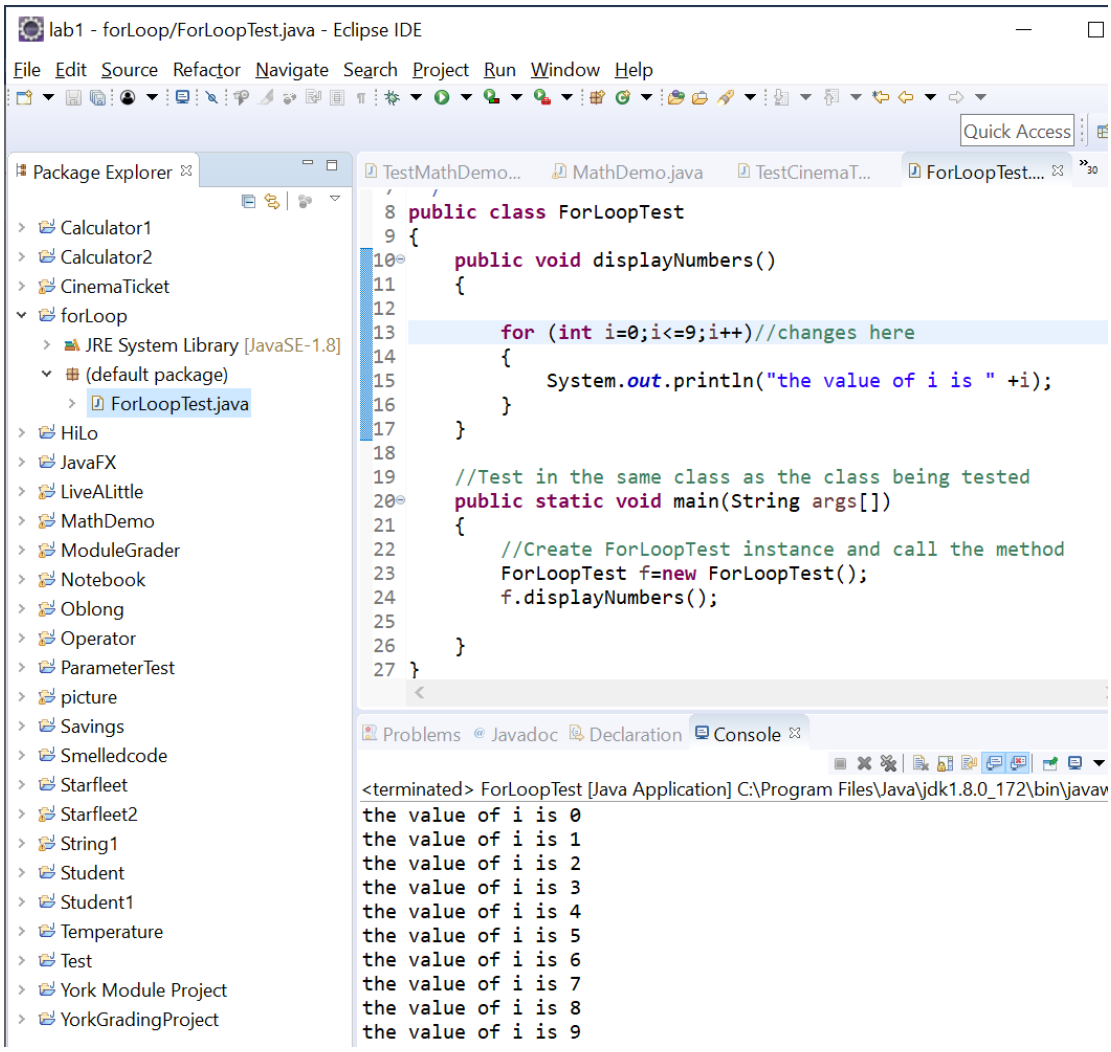


The snapshot below illustrate the modification of the program to print out numbers in reverse order.



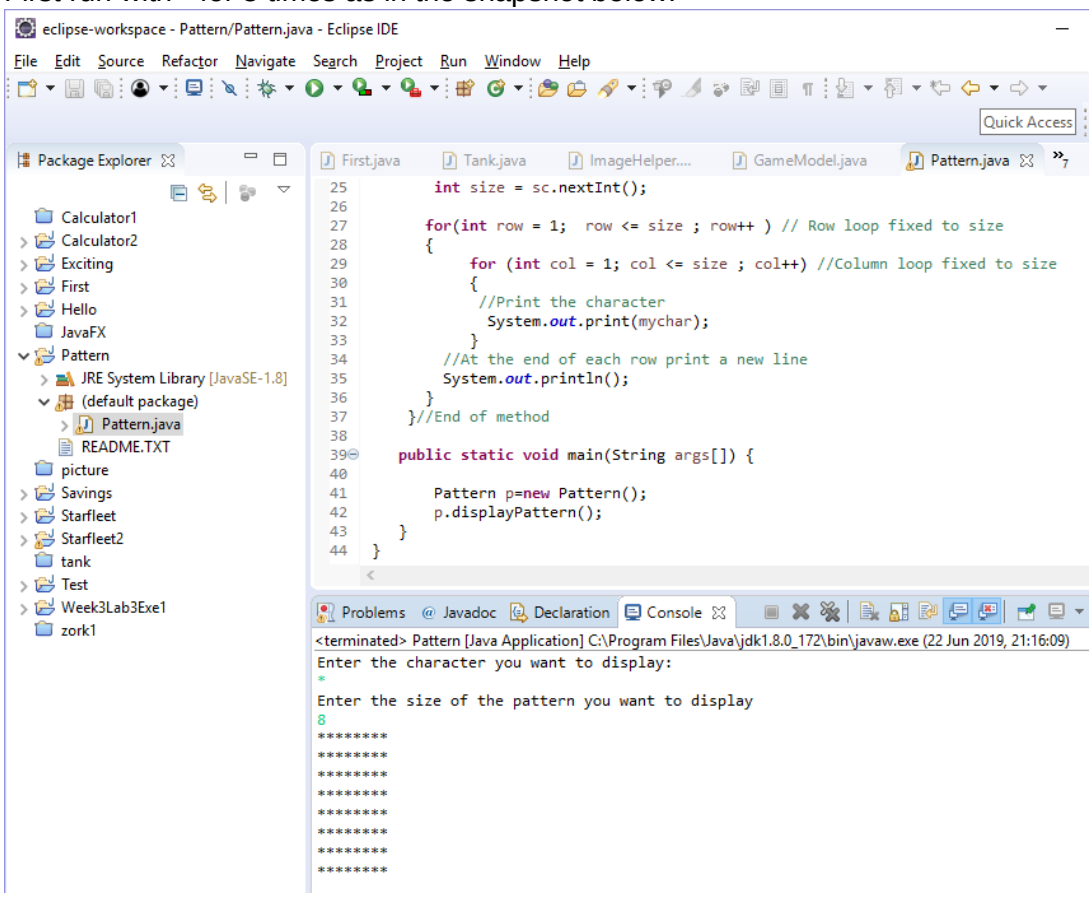
How could you modify the code above to start at 0 and stop at 9, rather than starting at 1 and stopping at 10?

Here is the snapshot that illustrate this:

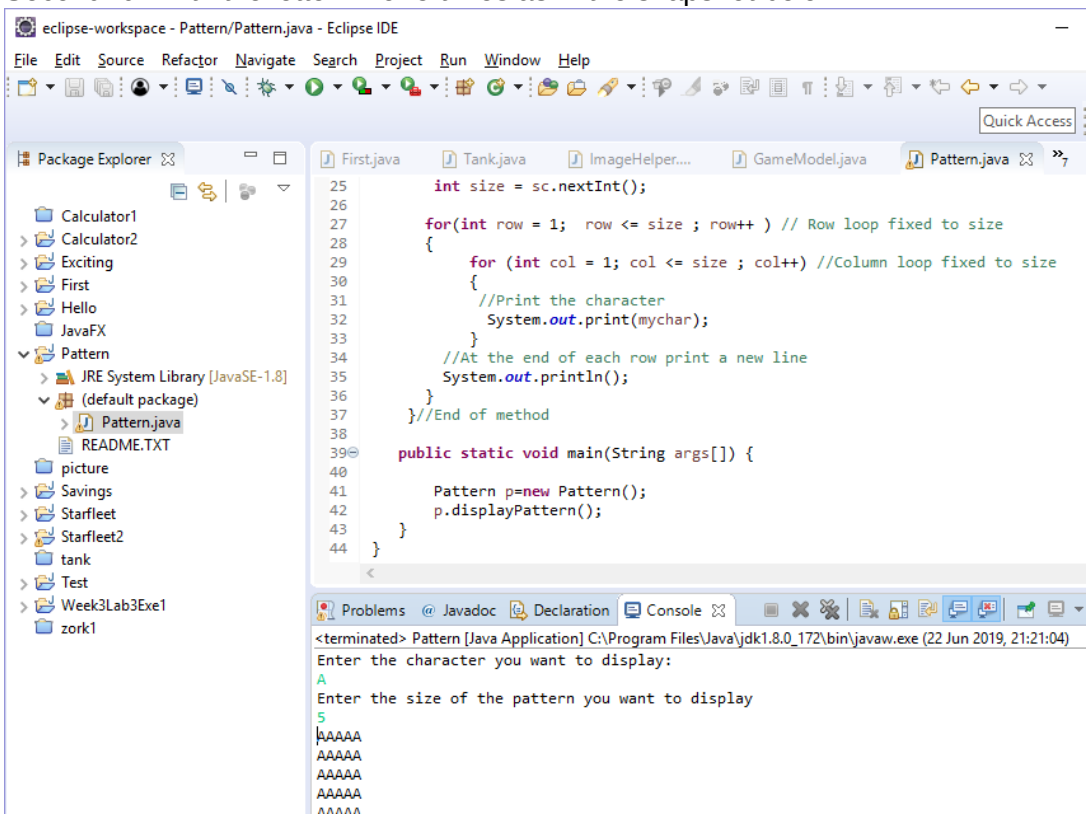


Task: Open Pattern project in Eclipse. Inspect the code within Pattern class. Run the Pattern program and interact with it a few times by entering the character and size of the pattern you want to display.

First run with * for 8 times as in the snapshot below:



Second run with the letter A for 5 times as in the snapshot below:



1.4.1: Other types of loop

```
mark=sc.nextInt();
while (mark < 0 || mark > 100) {
...
```

Task: See how the condition can be re-written using logical && and the ! (not) operators.

Here it is: while (!(mark >= 0 && mark <= 100))

1.4.3: Repetition - graphics example

Task: If size was set to 4, write a table of values for vert and horz when the program execution is at (or just below) the statement: ***//Code goes here***

Use the value pair (vert, horz) to represent the table as follows:

(0, 0) (0, 1) (0, 2) (0, 3)

(1, 0) (1, 1) (1, 2) (1, 3)

(2, 0) (2, 1) (2, 2) (2, 3)

(3, 0) (3, 1) (3, 2) (3, 3)