

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.

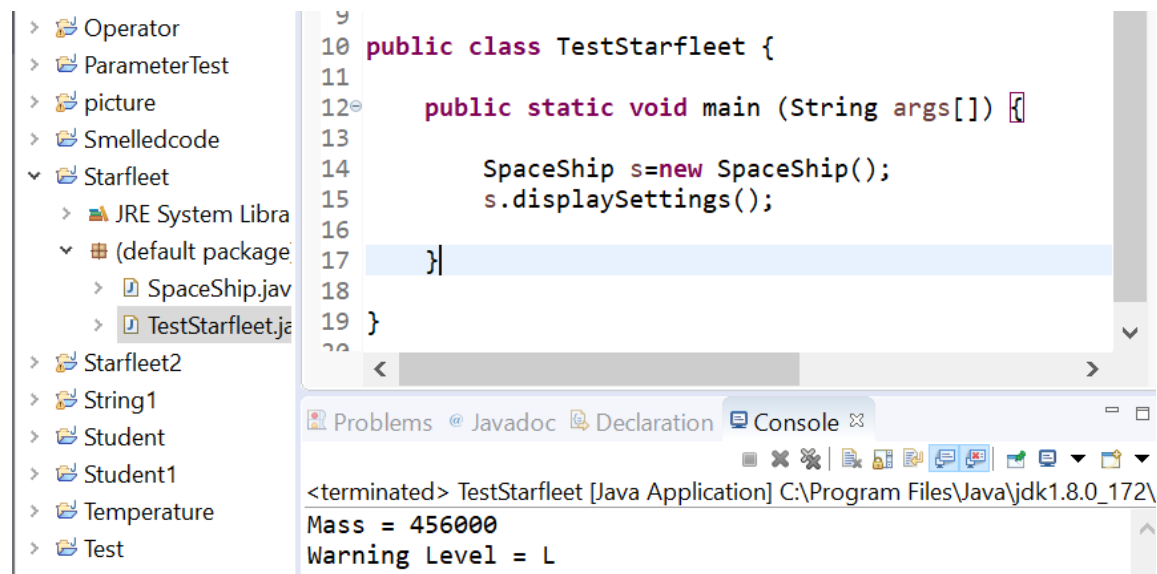
1.2.3: Assigning values to variables

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.

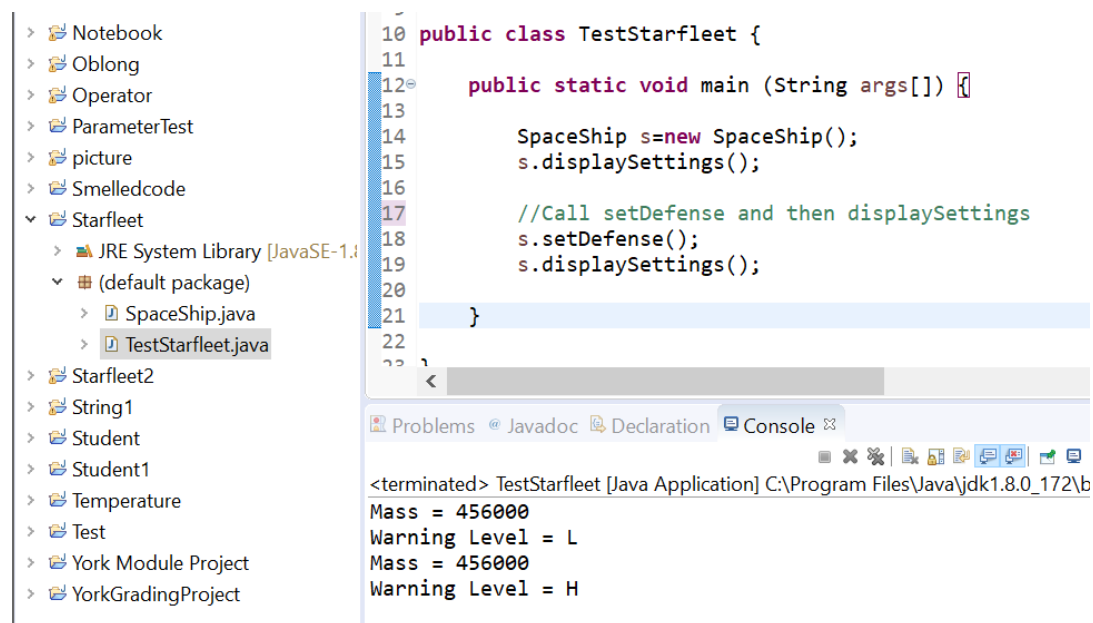
The snapshot below illustrates this.



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.



The screenshot shows the Eclipse IDE interface. On the left is the Project Explorer showing a project named 'Starfleet' with a package 'default package' containing 'SpaceShip.java' and 'TestStarfleet.java'. The 'TestStarfleet.java' file is selected and its code is displayed in the editor. The code is as follows:

```
10 public class TestStarfleet {
11
12     public static void main (String args[]) {
13
14         SpaceShip s=new SpaceShip();
15         s.displaySettings();
16
17         //Call setDefense and then displaySettings
18         s.setDefense();
19         s.displaySettings();
20
21     }
22 }
```

Below the editor is the Console window, which shows the output of the program:

```
<terminated> TestStarfleet [Java Application] C:\Program Files\Java\jdk1.8.0_172\b
Mass = 456000
Warning Level = L
Mass = 456000
Warning Level = H
```

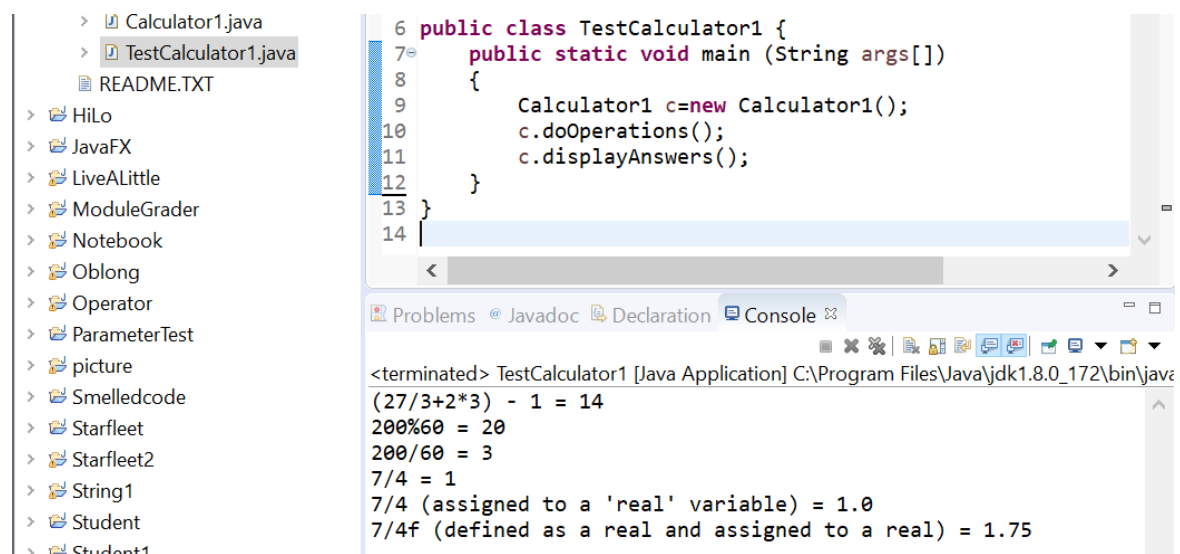
1.2.5: Arithmetic operators

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 TestCalculator1 class and check whether the outputs match your calculation.

Here it is the snapshot for running the program.



The screenshot shows the Eclipse IDE with the **TestCalculator1.java** file open. The code defines a **TestCalculator1** class with a **main** method that creates a **Calculator1** object and calls **doOperations()** and **displayAnswers()**.

```
6 public class TestCalculator1 {
7     public static void main (String args[])
8     {
9         Calculator1 c=new Calculator1();
10        c.doOperations();
11        c.displayAnswers();
12    }
13 }
14
```

The console output shows the results of the calculations:

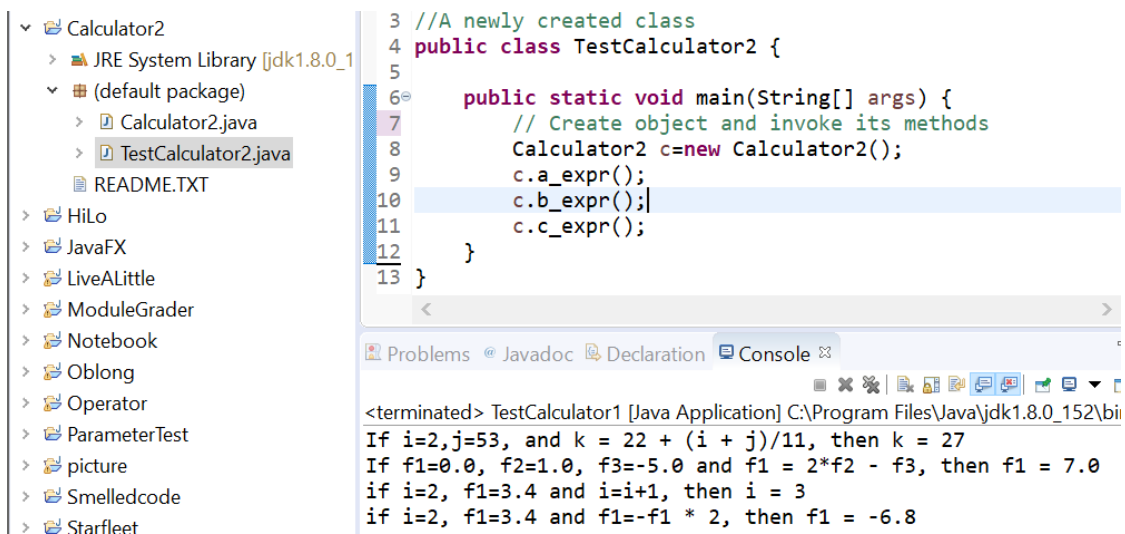
```
<terminated> TestCalculator1 [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\java
(27/3+2*3) - 1 = 14
200%60 = 20
200/60 = 3
7/4 = 1
7/4 (assigned to a 'real' variable) = 1.0
7/4f (defined as a real and assigned to a real) = 1.75
```

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.



The screenshot shows the Eclipse IDE with the 'Calculator2' project selected in the Package Explorer. The 'TestCalculator2.java' file is open in the Editor. The code defines a public class TestCalculator2 with a main method that creates a Calculator2 object and calls its a_expr(), b_expr(), and c_expr() methods. The Console window at the bottom displays the output of the program, showing the results of the calculations performed by the Calculator2 class.

```
3 //A newly created class
4 public class TestCalculator2 {
5
6     public static void main(String[] args) {
7         // Create object and invoke its methods
8         Calculator2 c=new Calculator2();
9         c.a_expr();
10        c.b_expr();
11        c.c_expr();
12    }
13 }
```

<terminated> TestCalculator1 [Java Application] C:\Program Files\Java\jdk1.8.0_152\bin
If i=2,j=53, and k = 22 + (i + j)/11, then k = 27
If f1=0.0, f2=1.0, f3=-5.0 and f1 = 2*f2 - f3, then f1 = 7.0
if i=2, f1=3.4 and i=i+1, then i = 3
if i=2, f1=3.4 and f1=-f1 * 2, then f1 = -6.8

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.

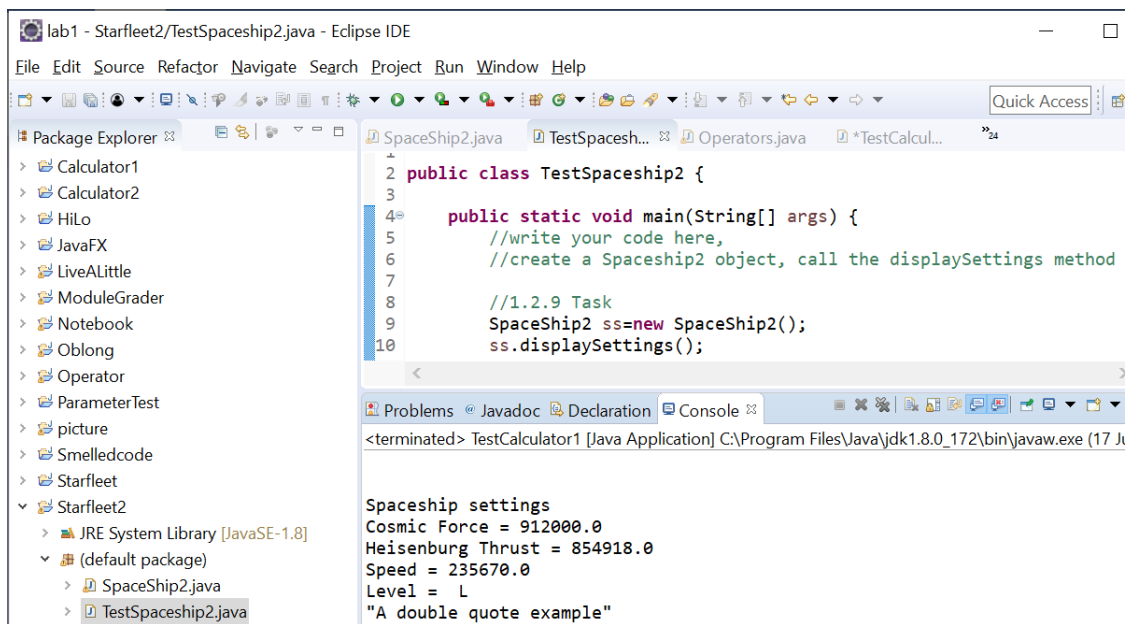
1. logicalAnswer = ! (false);
2. logicalAnswer = ! (a < b);

- | | |
|-------------------------------|-------|
| 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.

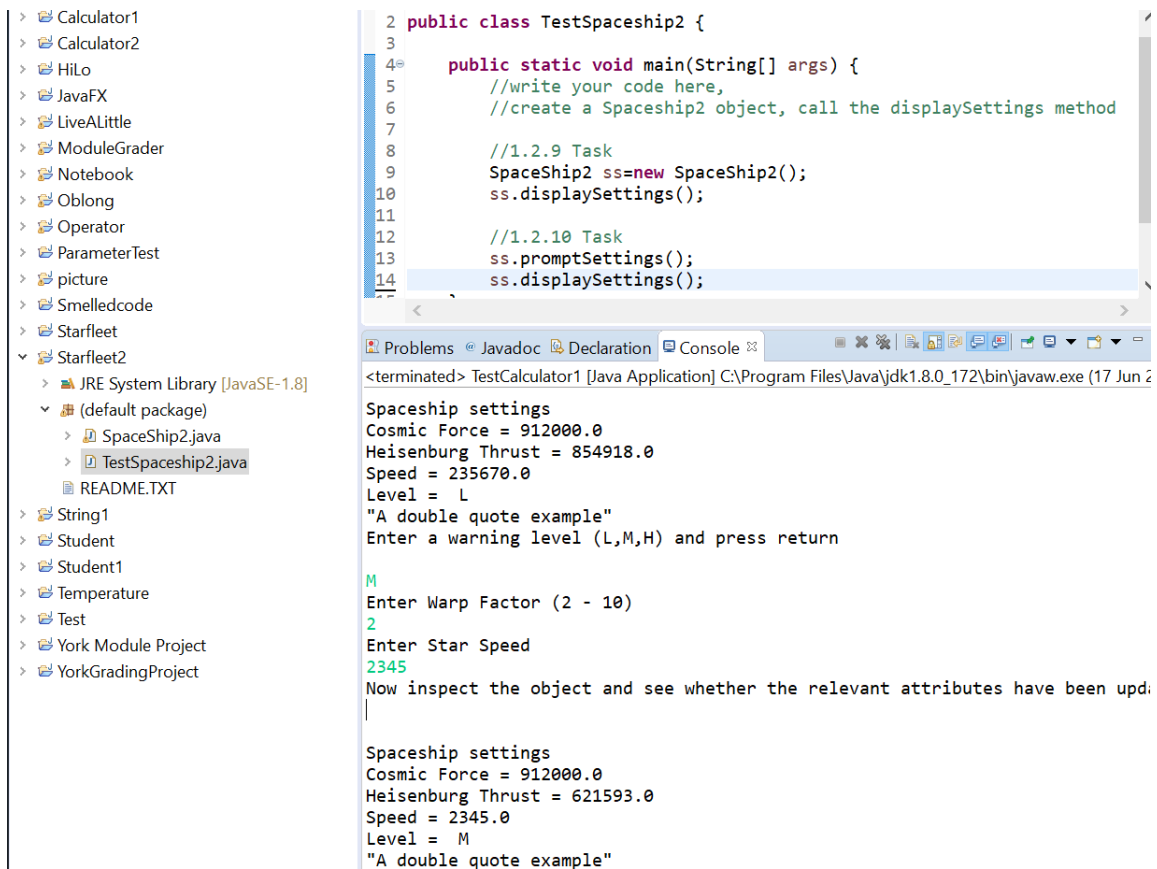
The snap shot below illustrates this.



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.



The screenshot shows an IDE with a project explorer on the left and a code editor on the right. The project explorer lists various files and folders, including 'Starfleet2' which contains 'SpaceShip2.java' and 'TestSpaceship2.java'. The code editor displays the `TestSpaceship2` class with a `main` method. The console output shows the results of running the program, including the initial settings and the updated values after user input.

```
2 public class TestSpaceship2 {
3
4     public static void main(String[] args) {
5         //write your code here,
6         //create a Spaceship2 object, call the displaySettings method
7
8         //1.2.9 Task
9         SpaceShip2 ss=new SpaceShip2();
10        ss.displaySettings();
11
12        //1.2.10 Task
13        ss.promptSettings();
14        ss.displaySettings();
15    }
16 }
```

Problems Javadoc Declaration Console

<terminated> TestCalculator1 [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (17 Jun 2023)

Spaceship settings
Cosmic Force = 912000.0
Heisenburg Thrust = 854918.0
Speed = 235670.0
Level = L
"A double quote example"
Enter a warning level (L,M,H) and press return

M
Enter Warp Factor (2 - 10)
2
Enter Star Speed
2345

Now inspect the object and see whether the relevant attributes have been updated

Spaceship settings
Cosmic Force = 912000.0
Heisenburg Thrust = 621593.0
Speed = 2345.0
Level = M
"A double quote example"

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.

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer displays the project structure: Calculator1, Calculator2, HiLo, JavaFX, LiveALittle, ModuleGrader, Notebook, Oblong, Operator, ParameterTest, picture, Savings (expanded), JRE System Library [jdk1.8.0_152], (default package), SavingsAccount.java, TestSavingsAccount.java, and README.TXT. The main editor shows the code for TestSavingsAccount.java:

```
1
2 public class TestSavingsAccount {
3
4     public static void main(String[] args) {
5         /* Create SavingsAccount object and
6          * call the methods
7          */
8         SavingsAccount s=new SavingsAccount();
9         s.initialiseAccountDetails();
10        s.deposit();
11        s.printStatement();
12        s.withdraw();
13        s.printStatement();
14    }
15 }
```

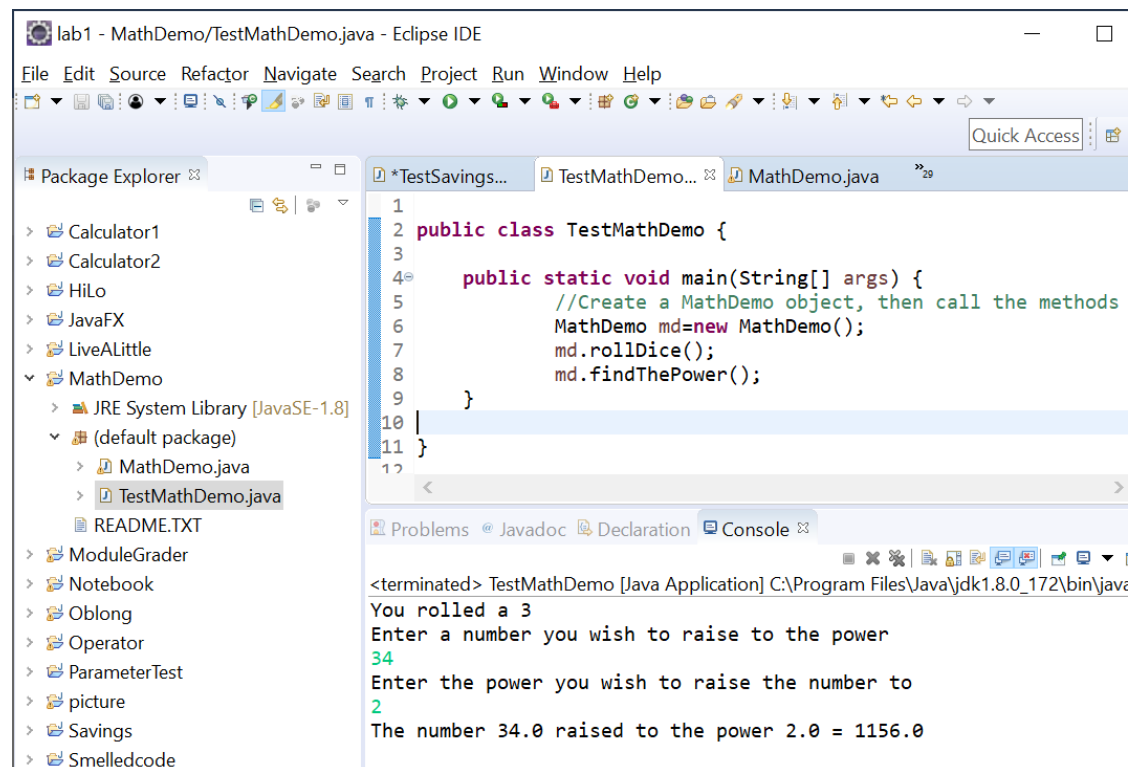
At the bottom, the Console tab shows the output of the program:

```
<terminated> TestCalculator1 [Java Application] C:\Program Files\Java\jdk1.
***** Creating A New Savings Account *****
Enter account ID (integer)
234
Please enter an initial balance
100
Account Created
ID = 234
Balance = 100.0
Please enter amount you wish to deposit
100
Transaction completed
Statement for account ID = 234
Current balance = 200.0
Please enter amount you wish to withdraw
300
Transaction completed
Statement for account ID = 234
Current balance = -100.0
```

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.



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure, with the MathDemo package expanded, showing MathDemo.java and TestMathDemo.java. The main editor window shows the TestMathDemo.java file with the following code:

```
1 public class TestMathDemo {
2
3
4     public static void main(String[] args) {
5         //Create a MathDemo object, then call the methods
6         MathDemo md=new MathDemo();
7         md.rollDice();
8         md.findThePower();
9     }
10
11 }
12
```

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

```
<terminated> TestMathDemo [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javac
You rolled a 3
Enter a number you wish to raise to the power
34
Enter the power you wish to raise the number to
2
The number 34.0 raised to the power 2.0 = 1156.0
```

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.

lab1 - CinemaTicket/TestCinemaTicket.java - Eclipse IDE

Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- Calculator1
- Calculator2
- CinemaTicket
 - JRE System Library [jdk1.8.0_...
 - (default package)
 - CinemaTicket.java
 - TestCinemaTicket.java**
 - README.TXT
- HiLo
- JavaFX
- LiveALittle
- MathDemo
- ModuleGrader
- Notebook
- Oblong
- Operator
- ParameterTest

```
1 public class TestCinemaTicket {
2
3
4     public static void main(String[] args) {
5         // create CinemaTicket object
6         CinemaTicket t=new CinemaTicket();
7         t.calculatePrice();
8
9     }
10 }
```

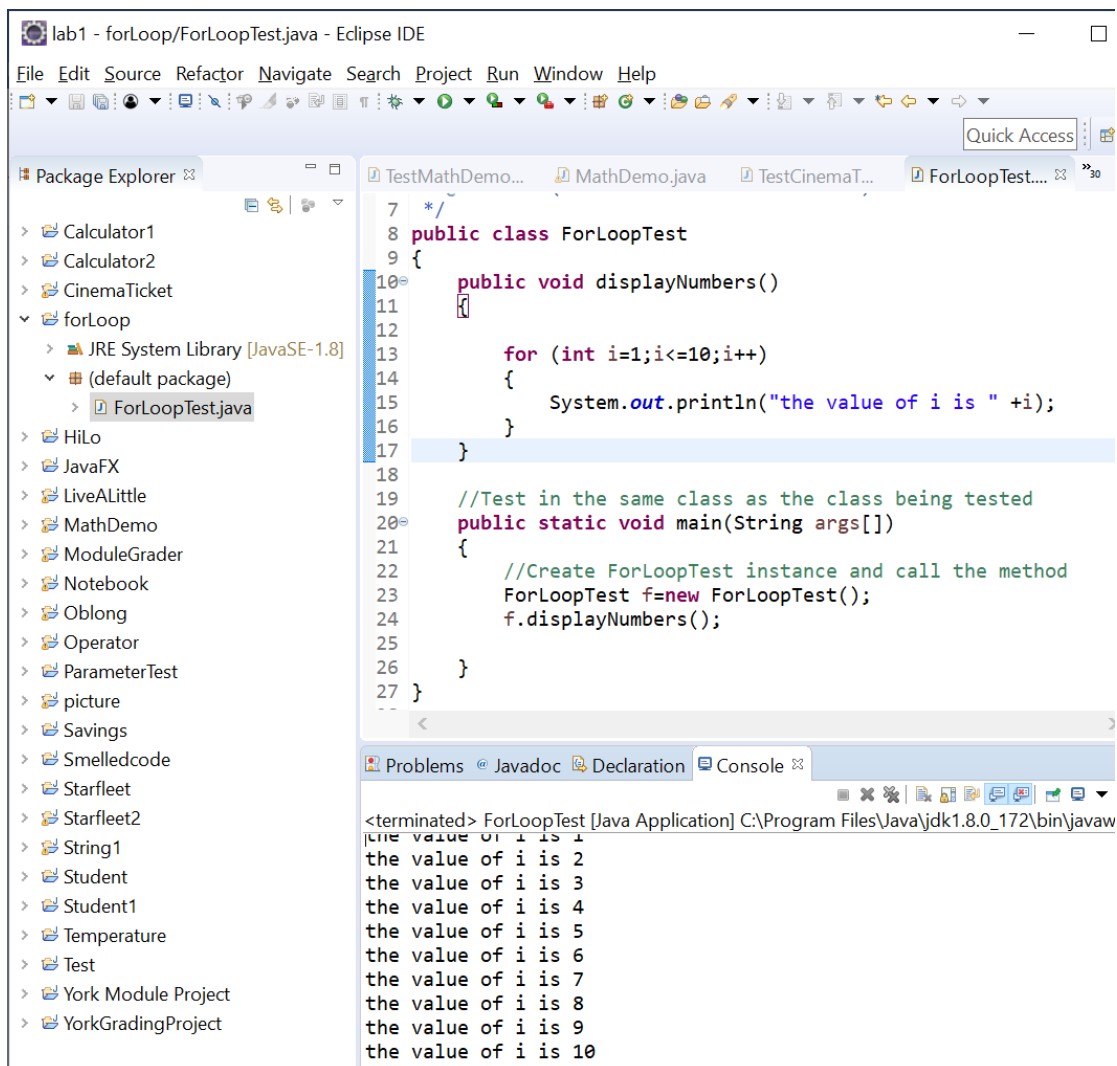
Problems @ Javadoc Declaration Console

<terminated> TestCinemaTicket [Java Application] C:\Program Files\Ja
What is the age of the customer ?
67
Is the customer a student (y/n)
n
This customer should be charged £4.0

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 screenshot shows the Eclipse IDE interface. The Package Explorer on the left lists a project named 'forLoop' containing a package 'default package' with a file 'ForLoopTest.java'. The main editor displays the source code of 'ForLoopTest.java'. The code defines a public class 'ForLoopTest' with a method 'displayNumbers()' that uses a for loop to print the values of 'i' from 1 to 10. A static 'main' method is also present, which creates an instance of 'ForLoopTest' and calls 'displayNumbers()'. The Console window at the bottom shows the output of the program, displaying the text 'the value of i is' followed by the numbers 1 through 10 on separate lines.

```
lab1 - forLoop/ForLoopTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access

Package Explorer
> Calculator1
> Calculator2
> CinemaTicket
> forLoop
  > JRE System Library [JavaSE-1.8]
  > (default package)
    > ForLoopTest.java
> HiLo
> JavaFX
> LiveALittle
> MathDemo
> ModuleGrader
> Notebook
> Oblong
> Operator
> ParameterTest
> picture
> Savings
> Smelledcode
> Starfleet
> Starfleet2
> String1
> Student
> Student1
> Temperature
> Test
> York Module Project
> YorkGradingProject

TestMathDemo... MathDemo.java TestCinemaT... ForLoopTest... 30

7  */
8  public class ForLoopTest
9  {
10     public void displayNumbers()
11     {
12
13         for (int i=1;i<=10;i++)
14         {
15             System.out.println("the value of i is " +i);
16         }
17     }
18
19     //Test in the same class as the class being tested
20     public static void main(String args[])
21     {
22         //Create ForLoopTest instance and call the method
23         ForLoopTest f=new ForLoopTest();
24         f.displayNumbers();
25     }
26 }
27

Problems Javadoc Declaration Console
<terminated> ForLoopTest [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw
the value of i is 1
the value of i is 2
the value of i is 3
the value of i is 4
the value of i is 5
the value of i is 6
the value of i is 7
the value of i is 8
the value of i is 9
the value of i is 10
```

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

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left lists several projects, with 'forLoop' expanded to show 'ForLoopTest.java'. The main editor displays the code for 'ForLoopTest.java'. The code is as follows:

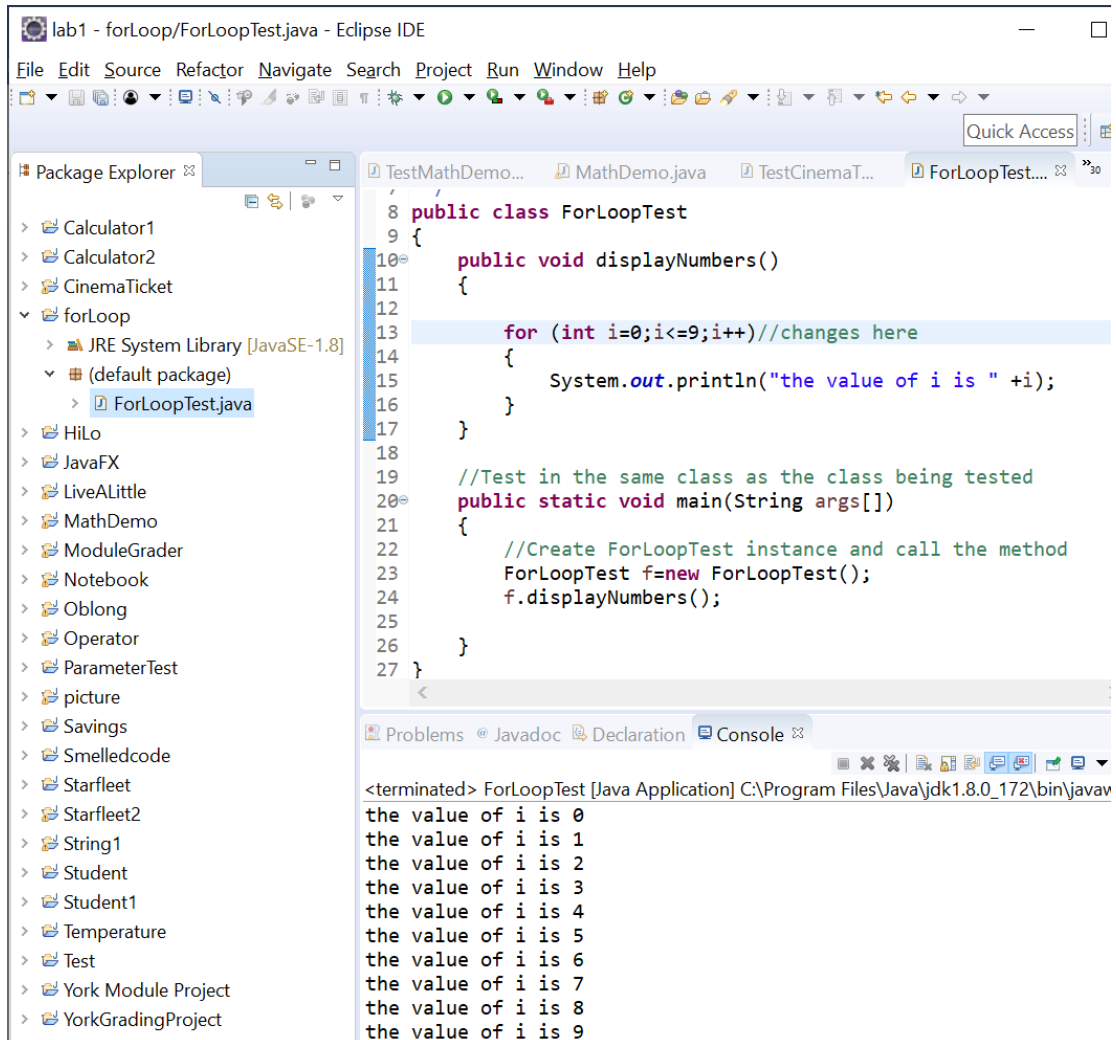
```
8 public class ForLoopTest
9 {
10     public void displayNumbers()
11     {
12
13         for (int i=10;i>=1;i--)//changes here
14         {
15             System.out.println("the value of i is " +i);
16         }
17     }
18
19     //Test in the same class as the class being tested
20     public static void main(String args[])
21     {
22         //Create ForLoopTest instance and call the method
23         ForLoopTest f=new ForLoopTest();
24         f.displayNumbers();
25     }
26 }
27 }
```

The console output at the bottom shows the program's execution, printing the values of 'i' from 10 down to 1 in reverse order:

```
<terminated> ForLoopTest [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe
the value of i is 10
the value of i is 9
the value of i is 8
the value of i is 7
the value of i is 6
the value of i is 5
the value of i is 4
the value of i is 3
the value of i is 2
the value of i is 1
```

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:



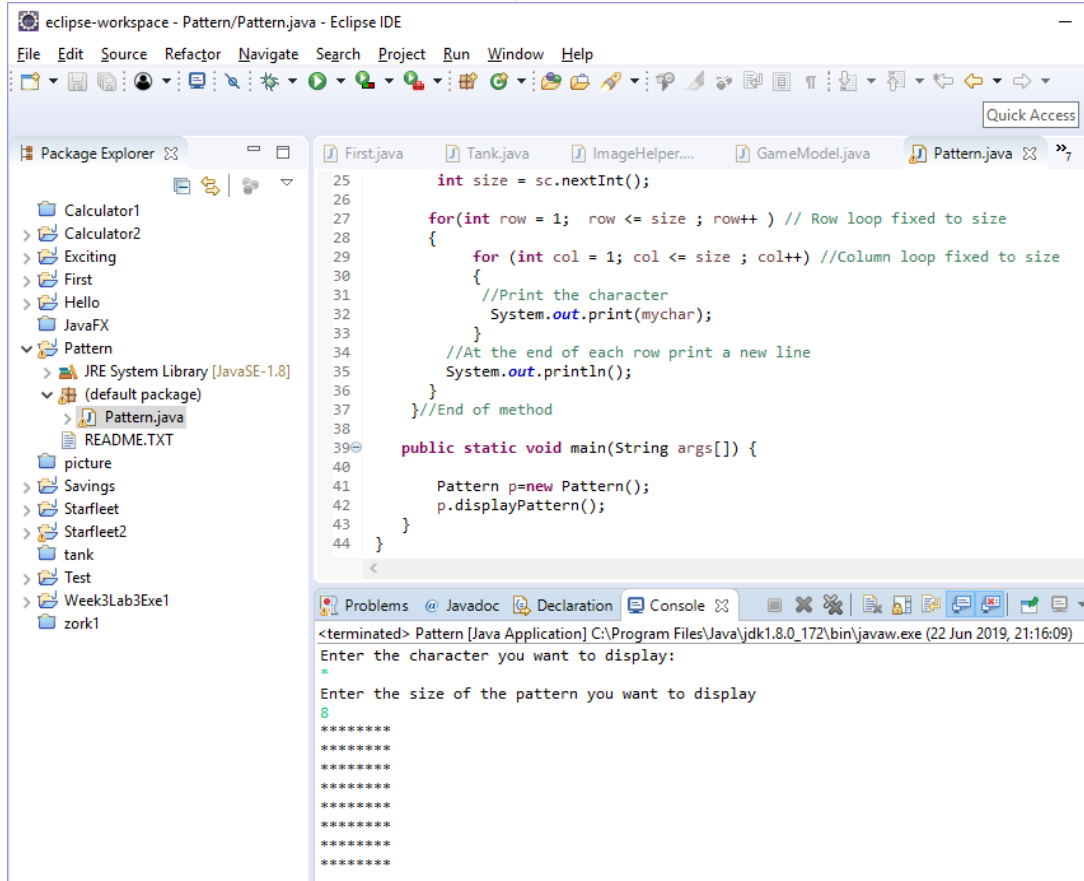
```
lab1 - forLoop/ForLoopTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
  Calculator1
  Calculator2
  CinemaTicket
  forLoop
    JRE System Library [JavaSE-1.8]
    (default package)
      ForLoopTest.java
  HiLo
  JavaFX
  LiveALittle
  MathDemo
  ModuleGrader
  Notebook
  Oblong
  Operator
  ParameterTest
  picture
  Savings
  Smelledcode
  Starfleet
  Starfleet2
  String1
  Student
  Student1
  Temperature
  Test
  York Module Project
  YorkGradingProject
  TestMathDemo...
  MathDemo.java
  TestCinemaT...
  ForLoopTest...
  Quick Access

8 public class ForLoopTest
9 {
10 public void displayNumbers()
11 {
12     for (int i=0;i<=9;i++)//changes here
13     {
14         System.out.println("the value of i is " +i);
15     }
16 }
17
18 //Test in the same class as the class being tested
19 public static void main(String args[])
20 {
21     //Create ForLoopTest instance and call the method
22     ForLoopTest f=new ForLoopTest();
23     f.displayNumbers();
24 }
25
26 }
27 }

Problems Javadoc Declaration Console
<terminated> ForLoopTest [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javav
the value of i is 0
the value of i is 1
the value of i is 2
the value of i is 3
the value of i is 4
the value of i is 5
the value of i is 6
the value of i is 7
the value of i is 8
the value of i is 9
```

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:



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left lists several projects, with 'Pattern' selected. The main editor displays the code for 'Pattern.java'. The code defines a 'Pattern' class with a 'displayPattern()' method that takes a character and a size as input. The 'main' method calls 'displayPattern()' with '*' and 8. The console at the bottom shows the program's execution, including prompts for character and size, and the resulting 8x8 asterisk pattern.

```
25     int size = sc.nextInt();
26
27     for(int row = 1; row <= size ; row++ ) // Row loop fixed to size
28     {
29         for (int col = 1; col <= size ; col++) //Column loop fixed to size
30         {
31             //Print the character
32             System.out.print(mychar);
33         }
34         //At the end of each row print a new line
35         System.out.println();
36     }
37 } //End of method
38
39 public static void main(String args[]) {
40
41     Pattern p=new Pattern();
42     p.displayPattern();
43 }
44 }
```

<terminated> Pattern [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (22 Jun 2019, 21:16:09)
Enter the character you want to display:
*
Enter the size of the pattern you want to display
8

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

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left lists several projects, with 'Pattern' selected. The main editor displays the code for 'Pattern.java'. The code defines a 'Pattern' class with a 'displayPattern()' method that takes a character and a size as input. The 'main' method calls 'displayPattern()' with 'A' and '5'. The Console at the bottom shows the execution output, including the prompts 'Enter the character you want to display:' and 'Enter the size of the pattern you want to display:', followed by the character 'A' and the size '5'. The final output is a 5x5 grid of 'A's.

```
25     int size = sc.nextInt();
26
27     for(int row = 1; row <= size ; row++ ) // Row loop fixed to size
28     {
29         for (int col = 1; col <= size ; col++) //Column loop fixed to size
30         {
31             //Print the character
32             System.out.print(mychar);
33         }
34         //At the end of each row print a new line
35         System.out.println();
36     }
37 }//End of method
38
39 public static void main(String args[]) {
40
41     Pattern p=new Pattern();
42     p.displayPattern();
43 }
44 }
```

<terminated> Pattern [Java Application] C:\Program Files\Java\jdk1.8.0_172\bin\javaw.exe (22 Jun 2019, 21:21:04)
Enter the character you want to display:
A
Enter the size of the pattern you want to display
5
AAAAA
AAAAA
AAAAA
AAAAA
AAAAA

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)