File I/O & Loops

COMP2026

PROBLEM SOLVING USING OBJECT ORIENTED PROGRAMMING

Overview

- File Input
- File Output
- File Append
- While loop
- Do… while loop
- For loop

File I/O

Reading Text File

- In Java, the most convenient way for reading text is to use Scanner class (java.util.Scanner)
- To read input from a file, the Scanner class relies on another class called File (java.io.File)
- The File class describes files and directories in disk

Reading Text File

- To create a File object with the filename input.txt:

```
File inputFile = new File("input.txt");
```

- Use the File object to create a Scanner object:

```
Scanner in = new Scanner(inputFile);
```

- This Scanner object reads text from the file input.txt instead of System.in

Reading Text File

- We can use the Scanner methods to read data from the input file
- nextInt(), nextDouble(), next(), nextLine(), etc.
- http://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html
- For example, you can use the following loop to process the input file line by line

```
while (in.hasNextLine())
{
  String line = in.nextLine();
  ...  //codes to process the line goes here
}
```

Method **hasNextLine()** checks if there another line in the file for reading

Reading Text File

- When you are done processing a file, be sure to close the Scanner

```
in.close();
```

FileInputExample

FileInputExample.java

```java
import java.util.Scanner;
import java.io.File;

public class FileInputExample {
    public static void main(String[] args) throws Exception {
        new FileInputExample().runApp();
    }


    void runApp() throws Exception {

        String filename = "input.txt";
        File inputFile = new File(filename);

        if (!inputFile.exists()) {
            System.out.println("The file " + filename + " is not found.");
            System.exit( status: 0);
        }

        Scanner in = new Scanner(inputFile);

        while (in.hasNextLine()) {
            String line = in.nextLine();
            System.out.println(line);
        }
        in.close();
    }
}
```

import the classes

Terminate the method if exception occurs
(If the input file for the Scanner doesn't exist, exception occurs when the Scanner object is creating.)

Create a File object

Check whether the file exists

Create a Scanner object

Read the file line by line and print the lines

Close the Scanner

Write Data to a File

- Create a **`PrintWriter`** object to open a file for writing

```
PrintWriter out = new PrintWriter("output.txt");
```

- Write data to the file by **`print`** and **`println`** method

```
out.print("Java ");
out.println("Programming");
```

- Use **`close`** method to close the file

```
out.close();
```

FileOutputExample

FileOutputExample.java

```java
import java.io.*;

public class FileOutputExample {
    public static void main(String[] args) throws Exception {
        new FileOutputExample().runApp();
    }


    void runApp() throws Exception {

        String filename = "output.txt";
        PrintWriter out = new PrintWriter(filename);

        out.print("This ");
        out.print("is ");
        out.println("line 1.");
        out.println("This is line 2.");

        out.close();
    }
}
```

import the classes

Terminate the method if exception occurs
(If the PrintWriter cannot open the file for writing, exception occurs when the PrintWriter object is creating.)

Create a PrintWriter object

Use **print()** to print sting
Use **println()** to print the string and moves the cursor to a newline

Close the PrintWriter

Appending Data to a File

- Create a **`FileWriter`** object with a filename and a true value for arguments
- Create a **`PrintWriter`** object so that you can use the **`print`** and **`println`** methods to write data to the file

```
FileWriter fwriter = new FileWriter("output.txt", true);
PrintWriter out = new PrintWriter(fwriter);
```

```
out.print("Java ");
out.println("Programming");
```

```
out.close();
```

- **Any data written to the file will be appended to the file's existing content**

FileAppendExample

FileAppendExample.java

import the classes

Terminate the method if exception occurs
(If the PrintWriter cannot open the file for writing, exception occurs when the PrintWriter object is creating.)

```java
import java.io.

public class FileAppendExample {
    public static void main(String[] args) throws Exception {
        new FileAppendExample().runApp();
    }

    void runApp() throws Exception {

        String filename = "appendOutput.txt";
        FileWriter fwriter = new FileWriter(filename, append: true);
        PrintWriter out = new PrintWriter(fwriter);

        out.println("This is a line.");

        out.close();
    }
}
```

Create a FileWriter object with true append value

Create a PrintWriter object

Use **println()** to print the string and moves the cursor to a newline

Close the PrintWriter

Loops

Consider a problem…

- Write a program to prompt user to enter an integer number in a range 1 to 100 (inclusive). If the input number is not in range, the program should allow the user to enter the number again **until a correct number is entered**.

You may think about…

```
…
//read input number
Scanner in = new Scanner(System.in);
System.out.print("Please input an integer number (1-100): ");
int num = in.nextInt();

if(num < 1 || num > 100 ){
  System.out.print("Please input an integer number (1-100): ");
  num = in.nextInt();
}

if(num < 1 || num > 100 ){
  System.out.print("Please input an integer number (1-100): ");
  num = in.nextInt();
}

...
```

If the input is not correct…

If the input is not correct again!!

Again and again…
We don't know how many times!!

How to solve it?

- Tell the computer to **repeat the steps by while loop**

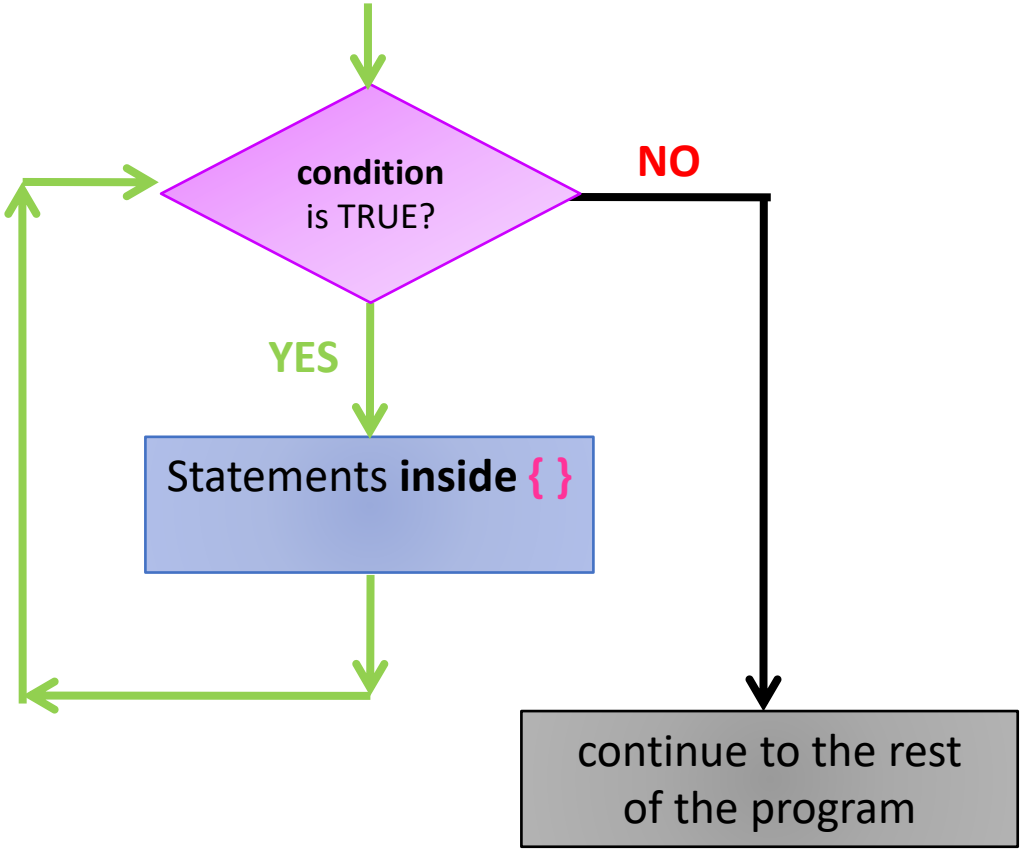
```
…
//read input number
Scanner in = new Scanner(System.in);
System.out.print("Please input an integer number (1-100): ");
int num = in.nextInt();

while(num < 1 || num > 100 ){
  System.out.print("Please input an integer number (1-100): ");
  num = in.nextInt();
}
```

Change `if` into `while`

while loop

- A repetition structure which repeats some actions **while** the condition remains **TRUE**

```
while (condition testing)
{
    // actions to repeat
    // while the condition is TRUE
    //
}
```



while loop

```
int i = 0;
while (i < 3) {
  System.out.println(i);
   i++;
}
```

Flow of Control

Let's trace an example:

```
int i = 0;
while (i < 3) {
  System.out.println(i);
   i++;
}
```

Flow of Control

```
int i = 0;
while (i < 3) {
  System.out.println(i);
   i++;
}
```

i = 0

Flow of Control

```
int i = 0;
while (i < 3) {
  System.out.println(i);
   i++;
}
```

i = 0

0 < 3 ➔ true

Flow of Control

```
int i = 0;
while (i < 3) {
  System.out.println(i);
  i++;
}
```

i = 0

0 < 3 ➔ true

print 0

Flow of Control

```
int i = 0;
while (i < 3) {
  System.out.println(i);
  i++;
}
```

i = 1

Flow of Control

```
int i = 0;
while (i < 3) {
  System.out.println(i);
   i++;
}
```

i = 1

1 < 3 ➔ true

Flow of Control

```
int i = 0;
while (i < 3) {
  System.out.println(i);
   i++;
}
```

i = 1

1 < 3 ➔ true

print 1

Flow of Control

```
int i = 0;
while (i < 3) {
  System.out.println(i);
   i++;
}
```

i = 2

Flow of Control

```
int i = 0;
while (i < 3) {
  System.out.println(i);
   i++;
}
```

i = 2

2 < 3 ➔ true

Flow of Control

```
int i = 0;
while (i < 3) {
  System.out.println(i);
  i++;
}
```

i = 2

2 < 3 ➔ true

print 2

Flow of Control

```
int i = 0;
while (i < 3) {
  System.out.println(i);
  i++;
}
```

i = 3

Flow of Control

```
int i = 0;
while (i < 3) {
  System.out.println(i);
   i++;
}
```

i = 3

3 < 3 ➔ false

Flow of Control

```
int i = 0;
while (i < 3) {
  System.out.println(i);
   i++;
}
```

i = 3

3 < 3 ➔ false

Stop and leave the loop

So, the while loop **repeats 3 times and prints:**

0
1
2

Problematic while loop

- What is the problem of the following while loop?

```
int i = 0;
while (i < 3) {
   System.out.println(i);
}
```

Let's trace it!

Problematic while loop

- What is the problem of the following while loop?

```
int i = 0;
while (i < 3) {
   System.out.println(i);
}
```

i = 0

Problematic while loop

- What is the problem of the following while loop?

```
int i = 0;
while (i < 3) {
   System.out.println(i);
}
```

i = 0

0 < 3 ➔ true

Problematic while loop

- What is the problem of the following while loop?

```
int i = 0;
while (i < 3) {
  System.out.println(i);
}
```

i = 0

0 < 3 ➔ true

print 0

Problematic while loop

- What is the problem of the following while loop?

```
int i = 0;
while (i < 3) {
   System.out.println(i);
}
```

i = 0

0 < 3 ➔ true

Problematic while loop

- What is the problem of the following while loop?

```
int i = 0;
while (i < 3) {
  System.out.println(i);
}
```

i = 0

0 < 3 ➔ true

print 0

Problematic while loop

- What is the problem of the following while loop?

```
int i = 0;
while (i < 3) {
   System.out.println(i);
}
```

i = 0

0 < 3 ➔ true

Problematic while loop

- What is the problem of the following while loop?

```
int i = 0;
while (i < 3) {
  System.out.println(i);
}
```

i = 0

0 < 3 ➔ true

print 0

`i` is not updated, the **condition is always true**.
The loop runs infinitely (never stop).
We called this infinite loop!

Think about…

```
…
//read input number
Scanner in = new Scanner(System.in);
System.out.print("Please input an integer number (1-100): ");
int num = in.nextInt();

while(num < 1 || num > 100 ){
  System.out.print("Please input an integer number (1-100): ");
  num = in.nextInt();
}
```

Repeating statements
An integer must be read before checking the range

do…while loop

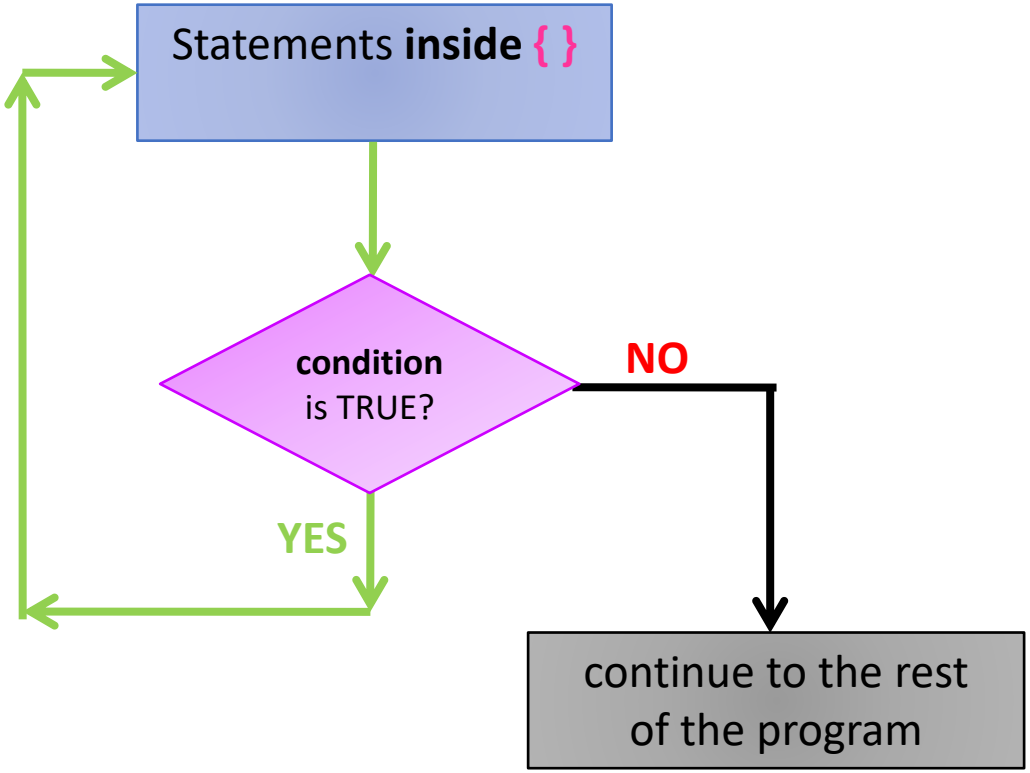
```
…
//read input number
Scanner in = new Scanner(System.in);

do{
  System.out.print("Please input an integer number (1-100): ");
  num = in.nextInt();
} while(num < 1 || num > 100 );
```

Do the loop body once before checking the condition

do…while loop

- A repetition structure which loop body must be executed **at least once**

```
do
{
    // actions to do
    //
} while  (condition testing) ;
   // repeat the actions while the
   // condition is true
```



do…while loop

```
int i = 0;
do {
  System.out.println(i);
   i++;
} while (i < 3);
```

Flow of Control

Let's trace an example:

```
int i = 0;
do {
  System.out.println(i);
   i++;
} while (i < 3);
```

Flow of Control

```
int i = 0;
do {
  System.out.println(i);
   i++;
} while (i < 3);
```

i = 0

Flow of Control

```
int i = 0;
do {
  System.out.println(i);
  i++;
} while (i < 3);
```

i = 0

print 0

Flow of Control

```
int i = 0;
do {
  System.out.println(i);
  i++;
} while (i < 3);
```

i = 1

Flow of Control

```
int i = 0;
do {
  System.out.println(i);
   i++;
} while (i < 3);
```

i = 1

1 < 3 ➔ true

Flow of Control

```
int i = 0;
do {
  System.out.println(i);
   i++;
} while (i < 3);
```

i = 1

print 1

Flow of Control

```
int i = 0;
do {
  System.out.println(i);
  i++;
} while (i < 3);
```

i = 2

Flow of Control

```
int i = 0;
do {
  System.out.println(i);
   i++;
} while (i < 3);
```

i = 2

2 < 3 ➔ true

Flow of Control

```
int i = 0;
do {
  System.out.println(i);
  i++;
} while (i < 3);
```

i = 2

print 2

Flow of Control

```
int i = 0;
do {
  System.out.println(i);
  i++;
} while (i < 3);
```

i = 3

Flow of Control

```
int i = 0;
do {
  System.out.println(i);
   i++;
} while (i < 3);
```

i = 3

3 < 3 ➔ false

Flow of Control

```
int i = 0;
do {
  System.out.println(i);
   i++;
} while (i < 3);
```

i = 3

Stop and
leave the loop

So, the do…while loop
repeats 3 times and prints:
0
1
2

do…while loop must execute **at least once**

```
int i = 10;
do {
  System.out.println(i);
   i++;
} while (i < 3);
```

do…while loop must execute **at least once**

```
int i = 10;
do {
  System.out.println(i);
   i++;
} while (i < 3);
```

i = 10

do…while loop must execute **at least once**

```
int i = 10;
do {
  System.out.println(i);
  i++;
} while (i < 3);
```

i = 10

print 10

do…while loop must execute **at least once**

```
int i = 10;
do {
  System.out.println(i);
  i++;
} while (i < 3);
```

i = 11

do…while loop must execute **at least once**

```
int i = 10;
do {
  System.out.println(i);
   i++;
} while (i < 3);
```

i = 11

11 < 3 ➔ false

do…while loop must execute **at least once**

```
int i = 10;
do {
  System.out.println(i);
   i++;
} while (i < 3);
```

i = 11

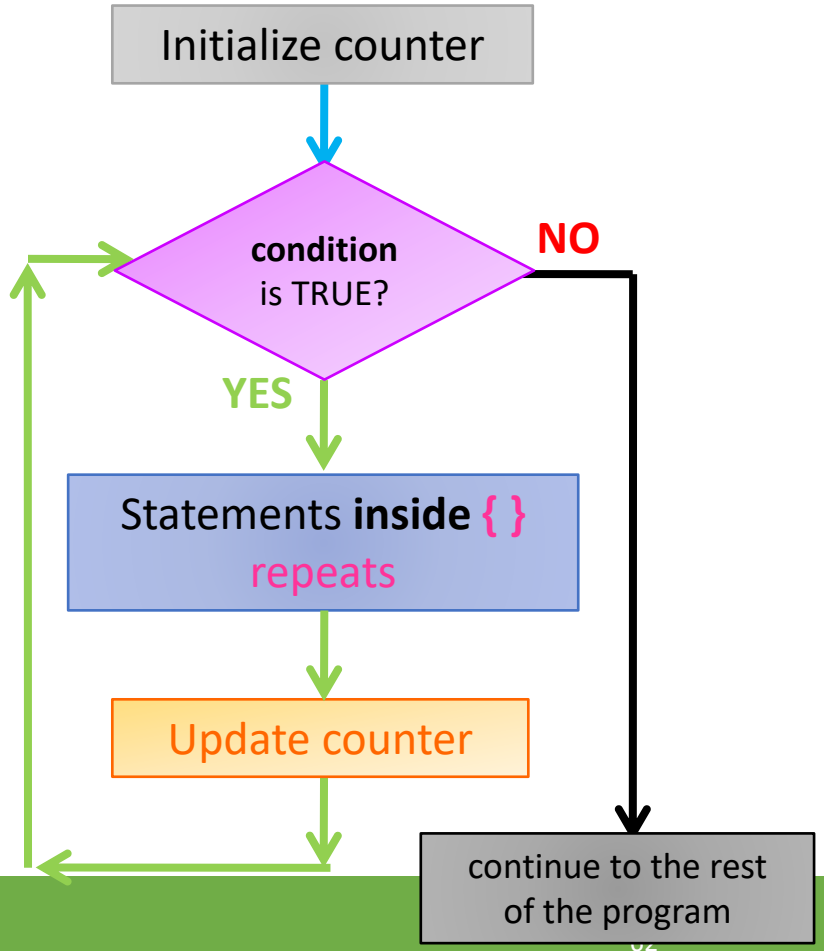
Stop and leave the loop

So, the do…while loop **executes once and prints: 10**

for loop

- Another repetition structure to repeat a block of code **for** a specified number of times

```
for(1. initialization; 2. condition; 4. update counter)
{
    // 3. actions to repeat
    // when the condition is TRUE
    //:
}
```



for loop

```
for (int i = 0; i < 3; i++)
{
  System.out.println(i);
}
```

Flow of Control

```
for (int i = 0; i < 3; i++)
{
   System.out.println(i);
}
```

i = 0

Flow of Control

```
for (int i = 0; i < 3; i++)
{
   System.out.println(i);
}
```

i = 0

0 < 3 ➔ true

Flow of Control

```
for (int i = 0; i < 3; i++)
{
    System.out.println(i);
}
```

i = 0

0 < 3 ➔ true

print 0

Flow of Control

```
for (int i = 0; i < 3; i++)
{
   System.out.println(i);
}
```

i = 1

Flow of Control

```
for (int i = 0; i < 3; i++)
{
   System.out.println(i);
}
```

i = 1

1 < 3 ➔ true

Flow of Control

```
for (int i = 0; i < 3; i++)
{
    System.out.println(i);
}
```

i = 1

1 < 3 ➔ true

print 1

Flow of Control

```
for (int i = 0; i < 3; i++)
{
   System.out.println(i);
}
```

i = 2

Flow of Control

```
for (int i = 0; i < 3; i++)
{
   System.out.println(i);
}
```

i = 2

2 < 3 ➔ true

Flow of Control

```
for (int i = 0; i < 3; i++)
{
    System.out.println(i);
}
```

i = 2

2 < 3 ➔ true

print 2

Flow of Control

```
for (int i = 0; i < 3; i++)
{
   System.out.println(i);
}
```

i = 3

Flow of Control

```
for (int i = 0; i < 3; i++)
{
   System.out.println(i);
}
```

i = 3

3 < 3 ➔ false

Flow of Control

```
for (int i = 0; i < 3; i++)
{
   System.out.println(i);
}
```

i = 3

3 < 3 ➔ false

Stop and leave the loop

So, the for loop **repeats 3 times and prints:**

0
1
2

Part A
Discovery Exercises

Type your answers in **XXXXXXXX_lab03.docx**

Part B
Programming Exercises

How to start?

- Suppose you are told to write a Java program called MyClass

MyClass.java

Class name

```
1   public class MyClass {
2       public static void main (String[] args) {
3           new MyClass().runApp();
4       }
5
6       void runApp(){
7
8
9       }
10  }
11
12
```

Write your code here!

Hints for Task 2

```
Enter the start day: 3
Enter the number of days: 30
Sun Mon Tue Wed Thu Fri Sat
--- --- --- --- --- --- ---
                1   2   3   4
  5   6   7   8   9  10  11
 12  13  14  15  16  17  18
 19  20  21  22  23  24  25
 26  27  28  29  30
--- --- --- --- --- --- ---
```

How many empty days?

When to move to next line?

When to stop?

Nested Loops

HINTS FOR TASK 5

Suppose…

- Suppose you are told to write a program that reads in an integer *n* and horizontally prints *n* asterisks (*)

```
n: 3
***
```

The solution is…

```java
System.out.print("n: ");
Scanner in = new Scanner(System.in);
int n = in.nextInt();

for (int i = 0; i < n; i++) {
    System.out.print("*");

}
```

Let's change the problem to…

- Write a program that reads in two integers *m* and *n*, and prints *m* rows of *n* asterisks (*)

```
Console   Prob
<terminated> Asterisk (1) [Ja
m: 5
n: 3
***
***
***
***
***
```

```
Console   Pro
<terminated> Asterisk (1) [J
m: 4
n: 10
**********
**********
**********
**********
```

How to solve it?

```
...

   for (int i = 0; i < n; i++) {
       System.out.print("*");
   }


...
```

Prints a line of *n* asterisks

Output:

```
**********
```

How to solve it?

```
...

    for (int i = 0; i < n; i++) {
        System.out.print("*");
    }
    System.out.println();

...
```

Prints a line of *n* asterisks

To **move to next line** after *n* asterisks is printed

Output:

```
**********
```

How to solve it?

```
...
for (int j = 0; j < m; j++) {
    for (int i = 0; i < n; i++) {
        System.out.print("*");
    }
    System.out.println();
}
...
```

Add an **outer loop** to repeat the line of *n* asterisks *m* times

Prints a line of *n* asterisks

To **move to next line** after *n* asterisks is printed

Output:

```
**********
```

```
**********
**********
**********
**********
```

How to solve it?

```
...
for (int j = 0; j < m; j++) {
    for (int i = 0; i < n; i++) {
        System.out.print("*");
    }
    System.out.println();
}
...
```

Add an **outer loop** to repeat the line of *n* asterisks *m* times

Prints a line of *n* asterisks

To **move to next line** after *n* asterisks is printed

A loop inside another loop statement. This is what we called nested loop.

Output:

```
**********
```

```
**********
**********
**********
**********
```

Nested Loop

```
...
for (int j = 0; j < m; j++) {
    for (int i = 0; i < n; i++) {
        System.out.print("*");
    }
    System.out.println();
}
...
```

outer loop iterates over rows

inner loop deals with the columns (asterisks) in the current row

Lab Exercise Submission

- Submit the following to Moodle
  - ***XXXXXXXX_ lab03.docx***
  - ***XXXXXXXX_lab03.zip***

*Replace "***XXXXXXXX***" with your **student ID**

Deadline: Next Monday 11:59 am

References

- Dean, J., & Dean, R. (2008). Introduction to programming with Java: A problem solving approach. Boston: McGraw-Hill.
- Forouzan, B. A., & Gilberg, R. F. (2007). Computer science: A structured programming approach using C (3rd ed.). Boston, MA: Thomson Course Technology.
- Gaddis, T. (2016). Starting out with Java (6th ed.). Pearson.
- Liang, Y. D. (2013). *Introduction to Java programming: Comprehensive version*. (8th ed.). Pearson.
- Schildt, H. (2006). *Java a beginner's guide*. New York: McGraw Hill.
- Wu, C. T. (2010). *An introduction to object-oriented programming with Java*. Boston: McGraw Hill Higher Education
- Xavier, C. (2011). Java programming: A practical approach. New Delhi: Tata McGraw Hill.
- Zakhour, S., Kannan, S., & Gallardo, R. (2013). The Java tutorial: A short course on the basics (5th ed.).
- yet another insignificant Programming Notes. (n.d.). Retrieved from https://www3.ntu.edu.sg/home/ehchua/programming