Computer Programing in Java

Lecture 1
Introduction, Syntax, Output, Comments

م.م حسين عجام

What is Java?

Java is a popular programming language, created in 1995.
It is owned by Oracle, and more than **3 billion** devices run Java.
It is used for:

- Mobile applications (specially Android apps)
- Desktop applications
- Web applications
- Web servers and application servers
- Games
- Database connection
- And more.

Why Use Java?

- Java works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.)
- It is one of the most popular programming language in the world
- It has a large demand in the current job market
- It is easy to learn and simple to use
- It is open-source and free
- It is secure, fast and powerful
- It has a huge community support (tens of millions of developers)
- Java is an object oriented language which gives a clear structure to programs and allows code to be reused, lowering development costs
- As Java is close to C++ and C#, it makes it easy for programmers to switch to Java or vice versa

Quick start

- In Java, every application begins with a **class name**, and that class must match the filename.
- Let's create our first Java file, called Main.java, which can be done in any text editor (like Eclips).

- The file should contain a "Hello World" message, which is written with the following code:

Quick start

```
public class Main {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

Java Syntax

Every line of code that runs in Java must be inside a class. In our example, we named the class Main. A class should always start with an uppercase first letter.

Note: Java is case-sensitive: "MyClass" and "myclass" has different meaning.

The main Method

- The main() method is required and you will see it in every Java program:

public static void main(String[] args)

- Any code inside the main() method will be executed. Don't worry about the keywords before and after main. You will get to know them bit by bit while reading this tutorial.

System.out.println()

- Inside the main() method, we can use the println() method to print a line of text to the screen:

```
public static void main(String[] args) {
        System.out.println("Hello World");
}
```

Java Output / Print

- Print Text
- You learned from the previous slides that you can use the println() method to output values or print text in Java:

```
System.out.println("Hello World!");
```

- You can add as many println() methods as you want. Note that it will add a new line for each method:

```
System.out.println("Hello World!");
System.out.println("I am learning Java.");
System.out.println("It is awesome!");
```

Double Quotes

- When you are working with text, it must be wrapped inside double quotations marks "".
- If you forget the double quotes, an error occurs:

```
System.out.println("This sentence will work!");
System.out.println(This sentence will produce an error);
```

The Print() Method

- There is also a print() method, which is similar to println().

The only difference is that it does not insert a new line at the end of the output:

```
System.out.print("Hello World! ");
System.out.print("I will print on the same line.");
```

Java Output Numbers

- Print Numbers

You can also use the println() method to print numbers.

However, unlike text, we don't put numbers inside double quotes:

```
System.out.println(3);
System.out.println(358);
System.out.println(50000);
```

- You can also perform mathematical calculations inside the println() method:

```
System.out.println(3 + 3);
System.out.println(2 * 5);
```

Java Comments

- Java Comments

Comments can be used to explain Java code, and to make it more readable. It can also be used to prevent execution when testing alternative code.

1- Single-line Comments

2- Java Multi-line Comments

Single-line Comments

- Single-line comments start with two forward slashes (//).

Any text between // and the end of the line is ignored by Java (will not be executed).

This example uses a single-line comment before a line of code:

```
// This is a comment
System.out.println("Hello World");
```

- This example uses a single-line comment at the end of a line of code:

```
System.out.println("Hello World"); // This is a comment
```

Java Multi-line Comments

- Multi-line comments start with /* and ends with */.

Any text between /* and */ will be ignored by Java.

This example uses a multi-line comment (a comment block) to explain the code:

```
/* The code below will print the words Hello World
to the screen, and it is amazing */
System.out.println("Hello World");
```

The End

- Thank you for listening