



week 0_2:

getting started
with Java

rajin teaches programming

INTRODUCTION TO JAVA:

- WHAT IS JAVA?

Java is a high-level, object-oriented programming language that follows the "write once, run anywhere" philosophy. Java code, once compiled, can run on any platform that has a Java Virtual Machine (JVM), making it highly portable.

- KEY CHARACTERISTICS:

Platform Independence: Java programs are compiled into bytecode, which can run on any machine that has a JVM installed.

Object-Oriented: Java follows the object-oriented programming paradigm, emphasizing reusability, modularity, and maintainability.

Automatic Memory Management: Java has a built-in garbage collector that automatically manages memory, reducing memory leaks.

Rich Standard Library: Java provides an extensive library of pre-built methods and classes, making it easier to perform various tasks like file handling, networking, and data structures.

- KEYWORDS:

JVM (Java Virtual Machine): The virtual machine that runs Java bytecode on any platform.

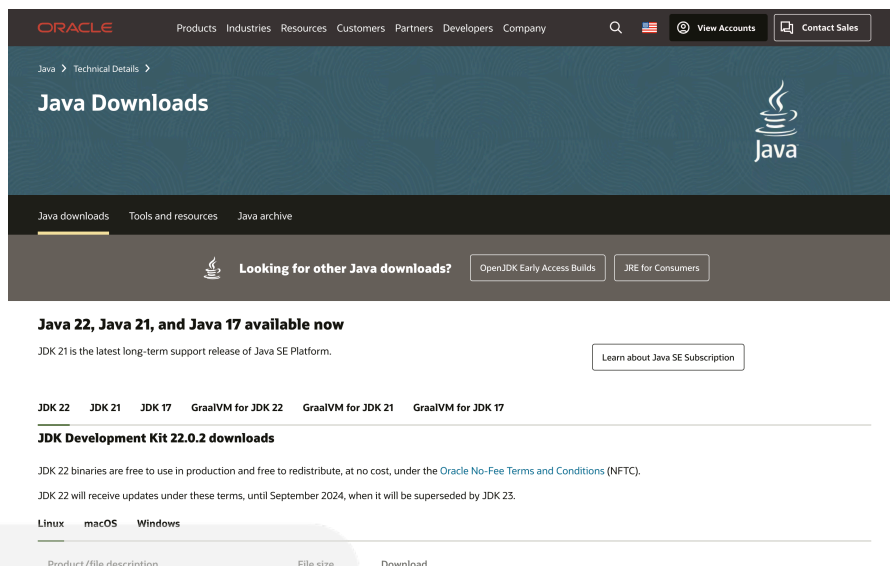
JRE (Java Runtime Environment): Provides the environment for executing Java programs.

JDK (Java Development Kit): A development kit that includes the JRE and development tools like the Java compiler.

GETTING STARTED WITH JAVA ON WINDOWS:

– SETTING UP THE ENVIRONMENT:

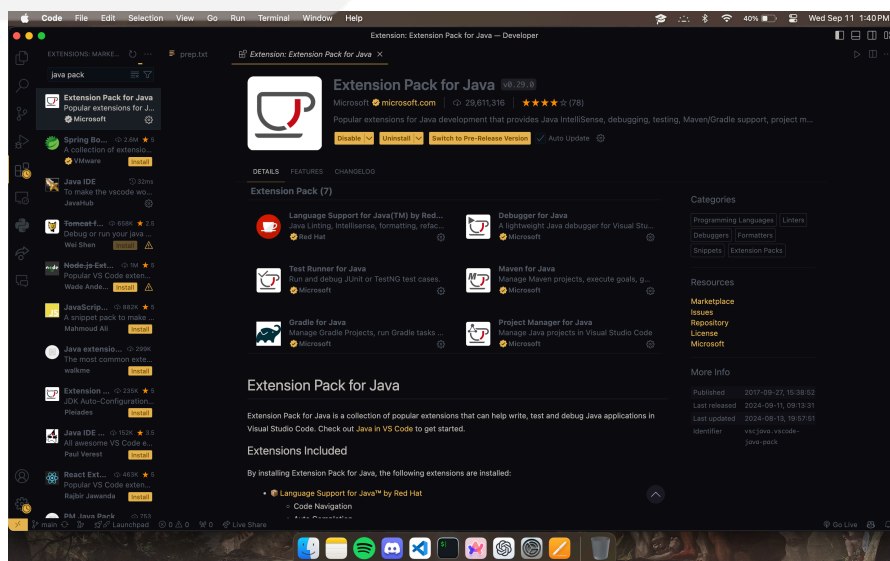
Download and install the latest **JDK** from the official [Oracle website](https://www.oracle.com/in/java/technologies/javase-downloads.html).



Then, set up the **JAVA_HOME** environment variable after installation (Will be shown in class).

– SETTING UP VSCODE:

If you haven't already, download and install [VSCode](https://code.visualstudio.com/).

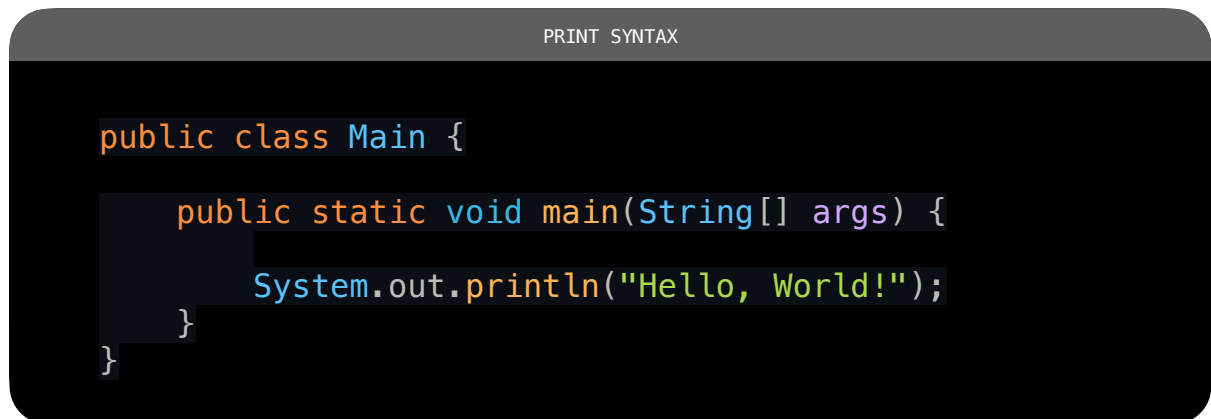


Go to the Extensions tab (left sidebar), search for "**Java Extension Pack**" and install it. This will add support for writing and running Java in VSCode.

WRITING YOUR FIRST PROGRAM:

Create a new project folder, and name it **Main.java** (It must start with a capital letter). The file name must also match the class name, as this is a **strict** rule in Java for public classes.

- GETTING OUTPUT:



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

- WHAT IT MEANS:

- **public class Main**: The Main class is a blueprint for your program. Every Java program starts by executing the main method in one of its classes.
- **File name must match the class name**: In Java, if your class is public (which is typical for the entry class of a program), the file name must exactly match the class name. So, if the class is named Main, the file must be Main.java (even the case must be matched).
- **public static void main(String[] args)**: This is the starting point of any Java application. When you run a program, the JVM looks for the main method to begin execution.
- **public**: The method can be called from anywhere.
- **static**: It belongs to the class itself, not an instance of the class.
- **void**: The method does not return any value.
- **main(String[] args)**: This is the method signature, where args is an array of strings passed to the program as arguments.

System.out.println("Hello, World!"): This line prints the text "Hello, World!" to the console. System.out is a built-in Java object that handles output, and println is a method that prints the specified message followed by a new line.

USING THE **System.out.println()** METHOD:

The System.out.println() method is a versatile function in Java used to print messages, variables, and even formatted strings to the console. Let's learn how to use this method effectively.

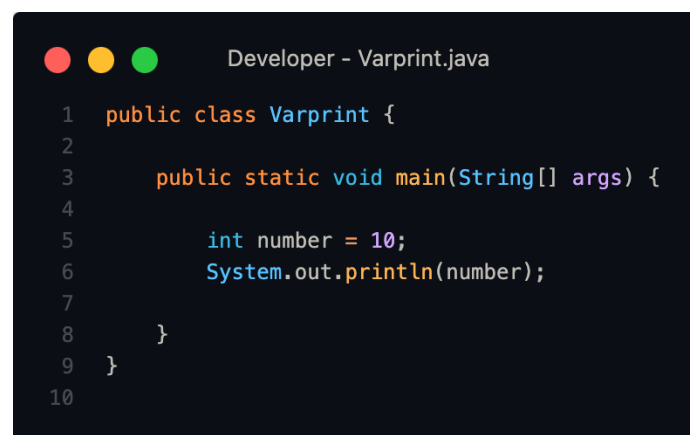
– BASIC USAGE:

A screenshot of a code editor window titled "Developer - basicprint.java". The code is as follows:

```
1 public class Basicprint {  
2  
3     public static void main(String[] args) {  
4  
5         System.out.println("This is a basic print statement");  
6     }  
7  
8 }
```

*code showing a basic print statement in Java.
Keep in mind that a new line is always added to
the end when you use the println() method.*

– PRINTING VARIABLES:

A screenshot of a code editor window titled "Developer - Varprint.java". The code is as follows:

```
1 public class Varprint {  
2  
3     public static void main(String[] args) {  
4  
5         int number = 10;  
6         System.out.println(number);  
7     }  
8  
9 }  
10
```

*this method can also be used to print variables
of different data types like integers, floats,
or strings.*

```
Developer - Combineprint.java

1 public class Combineprint {
2
3     public static void main(String[] args) {
4
5         String name = "Alice";
6         System.out.println("Hello, " + name + "!"); // Output: Hello, Alice!
7
8     }
9 }
```

you can also combine text with variables by concatenating them with the '+' operator, making it much easier to print things than in C (which used printf).

```
Developer - Multicombineprint.java

1 public class Multicombineprint {
2
3     public static void main(String[] args) {
4
5         int age = 25;
6         String name = "Bob";
7         System.out.println("Name: " + name + ", Age: " + age); // Output: Name: Bob, Age: 25
8
9     }
10 }
```

in the same manner, you can print multiple variables by using concatenation.

- print() VS println():

```
Developer - Main.java

1 public class Main {
2
3     public static void main(String[] args) {
4
5         System.out.print("Hello");
6         System.out.print("World");
7     }
8 }
```

```
cd "/Users/rajin/Developer/" && javac Main.java && java Main
~/Developer
cd "/Users/rajin/Developer/" && javac Main.java && java Main
HelloWorld
```

from the output on the right, you can see that System.out.print() prints the text but does not move the cursor to the next line after printing.

```
Developer - Main.java
1 public class Main {
2
3     public static void main(String[] args) {
4
5         System.out.println("Hello");
6         System.out.println("World");
7     }
8 }
```

```
~/Developer
cd "/Users/rajin/Developer/" && javac Main.java && java Main
Hello
World
```

from the output on the right, you can see that `System.out.println()` prints the text and moves the cursor to the next line after printing.

So, in theory, you could add multiple “lines” after what you print simply by typing `System.out.printlnlnlnln();` (and so on and so forth).

SUMMARY:

Class and File Name Matching: In Java, a file containing a public class must have the same name as the class (e.g., `Main.java` for the `Main` class).

`public static void main(String[] args):` This is the entry point of every Java program. It is the method that the JVM calls to start the execution.

`System.out.println():` This method is used to print messages, variables, or results to the console. The `println` function moves the cursor to the next line after printing, while `print()` does not.

Formatted Output: The `printf()` method allows you to format output in a structured manner.

FURTHER READING:

- BOOKS:

- "Head First Java" by Kathy Sierra and Bert Bates
- "Java: The Complete Reference" by Herbert Schildt

- ONLINE:

[Oracle Java Documentation](#)

[Java GeeksForGeeks Online Tutorial](#)



next class 1_0:
programming basics

rajin teaches programming

reach out for classes at: rajin.khan2001@gmail.com