



## Computer Programming 2

### Module 1: Getting Started to Java Programming

Name (LN, FN, MN): \_\_\_\_\_ Program/Yr/Block: \_\_\_\_\_

#### I. Introduction

This module will give a review about the fundamental ideas of programming and an introduction about Java technology. This module also enables you to get started in Java programming by introducing the tools that you need to write your first Java program. After completing this module, you should be able to have the basic knowledge about object-oriented programming and how Java code is executed using the console or an online Java editor.

Please feel free to use other references or tutorials about Java so that we can have an interactive discussion during our synchronous classes.

Welcome to Java programming and we hope that you will enjoy coding.

#### II. Learning Objectives

After completing this module, you should be able to:

1. Install JDK and execute your first script using the console.
2. Describe the anatomy of a basic Java program and the process of developing a program in Java.
3. Describe the features of Java technology (OOP, Multi-platform, High-level PL)
4. Differentiate JDK, JRE and JVM from each other.
5. Explain the process of translating java source code into a byte code procedure.

#### Topics and Key Concepts

##### A. Why Programming?


"A human being should be able to change a diaper, plan an invasion, butcher a hog, conn a ship, design a building, write a sonnet, balance accounts, build a wall, set a bone, comfort the dying, take orders, give orders, cooperate, act alone, solve equations, analyze a new problem, pitch manure, **program a computer**, cook a tasty meal, fight efficiently, die gallantly. Specialization is for insects."

-Robert Heinlein, Time Enough for Love (1973)

You need to know how to program in order to be able to tell a computer what you want it to.

Programming is not just for experts. It is a natural, satisfying and creative experience that enables accomplishments not otherwise possible. Programming may also be considered a path to a new world of intellectual endeavor.

However, programming is a challenging undertaking because you need to learn what computers can do and you need to learn a programming language.

Machine language	Natural language	High-level language
<ul style="list-style-type: none"><li>• Easy for computer.</li><li>• Error-prone for human.</li></ul>	<ul style="list-style-type: none"><li>• Easy for human.</li><li>• Error-prone for computer.</li></ul>	<ul style="list-style-type: none"><li>• Some difficulty for both.</li><li>• An acceptable tradeoff.</li></ul>
<pre>10: 8A00  RA ← mem[00] 11: 8B01  RB ← mem[01] 12: 1CAB  RC ← RA + RB 13: 9C02  mem[02] ← RC 14: 0000  halt</pre>		<pre>for (int t = 0; t &lt; 2000; t++) {     a[0] = a[11] ^ a[9];     System.out.print(a[0]);     for (int i = 11; i &gt; 0; i--)         a[i] = a[i-1]; }</pre>

Machine language consists of binary or hexadecimal instructions which a computer can respond to directly. Natural language is the language that we humans use to communicate with one another. High-level language, on the other hand, are designed to enable programmers to develop programs that is easier to learn and implement than low-level/machine languages.

But which high-level language?



Of course, our choice is Java!

*“There are only two kinds of programming languages: those people always [gripe] about and those nobody uses.”*

– Bjarne Stroustrup



## B. Java Technology

Java technology is both a programming language and a platform.

### B.1 The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:



Simple  
Object oriented  
Distributed  
Multithreaded  
Dynamic

Architecture neutral  
Portable  
High performance  
Robust  
Secure

The Java Programming Language is a high-level language. Its syntax is similar to C and C++ but it removes many of the complex, confusing features of C and C++.

The Java Programming Language includes automatic storage (memory) management by using a garbage collector.

The Java Programming Language source code is compiled into the bytecode instruction set, which can be run inside the Java Virtual Machine (JVM) process.

#### Java features

- Widely used.
- Widely available.
- Continuously under development since early 1990s.
- Embraces full set of modern abstractions.
- Variety of automatic checks for mistakes in programs.



James Gosling

## B.2 The Java Platform

A platform is the hardware or software environment in which a program runs. Some of the most popular platforms like Microsoft Windows, Linux, Solaris OS, and Mac OS. Most platforms can be described as a combination of the operating system and underlying hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of another hardware-based platform.



The primary goals of the Java platform are:

- provide an object-oriented programming language
- provide interpreted and just-in-time runtime environment
- dynamic class loading
- multi-thread capability.

As a platform-independent environment, the Java platform can be a bit slower than native code. However, advances in compiler and virtual machine technologies are bringing performance close to that of native code without threatening portability.



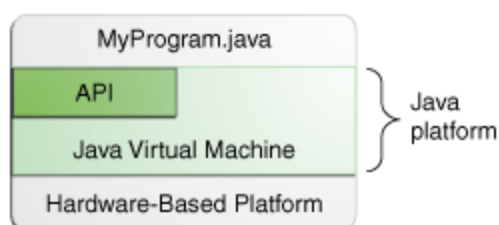
There are four platforms of the Java programming language:

- **Java Standard Edition (Java SE)**  
Java SE's API provides the core functionality of the Java programming language. It defines everything from the basic types and objects of the Java programming language to high-level classes that are used for networking, security, database access, graphical user interface (GUI) development, and XML parsing.
- **Java Enterprise Edition (Java EE)**  
The Java EE platform is built on top of the Java SE platform. The Java EE platform provides an API and runtime environment for developing and running large-scale, multi-tiered, scalable, reliable, and secure network applications.
- **Java Micro Edition (Java ME)**  
The Java ME platform provides an API and a small-footprint virtual machine for running Java programming language applications on small devices, like mobile phones. The API is a subset of the Java SE API, along with special class libraries useful for small device application development. Java ME applications are often clients of Java EE platform services.
- **JavaFX**  
JavaFX is a platform for creating rich internet applications using a lightweight user-interface API. JavaFX applications use hardware-



accelerated graphics and media engines to take advantage of higher-performance clients and a modern look-and-feel as well as high-level APIs for connecting to networked data sources. JavaFX applications may be clients of Java EE platform services.

All Java platforms consist of a Java Virtual Machine (VM) and an application programming interface (API). The **Java Virtual Machine** is a program, for a particular hardware and software platform, that runs Java technology applications. An **API** is a collection of software components that you can use to create other software components or applications. Each Java platform provides a virtual machine and an API, and this allows applications written for that platform to run on any compatible system with all the advantages of the Java programming language: platform-independence, power, stability, ease-of-development, and security.



The API and Java Virtual Machine insulate the program from the underlying hardware.

### C. What Can Java Technology Do?

The general-purpose, high-level Java programming language is a powerful software platform. Every full implementation of the Java platform gives you the following features:

**Development Tools:** The development tools provide everything you'll need for compiling, running, monitoring, debugging, and documenting your applications. As a new developer, the main tools you'll be using are the javac compiler, the java launcher, and the javadoc documentation tool.

**Application Programming Interface (API):** The API provides the core functionality of the Java programming language. It offers a wide array of useful classes ready for use in your own applications. It spans everything from basic objects, to networking and security, to XML generation and database access, and more.

**Deployment Technologies:** The JDK software provides standard mechanisms such as the Java Web Start software and Java Plug-In software for deploying your applications to end users.

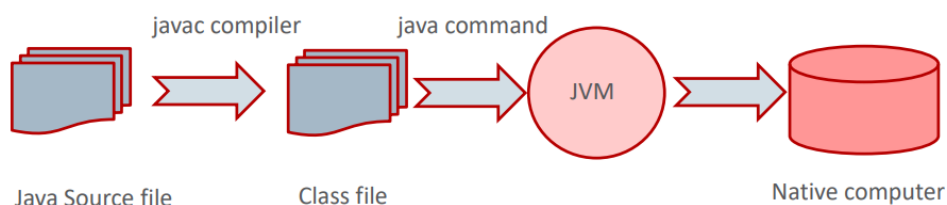


**User Interface Toolkits:** The JavaFX, Swing, and Java 2D toolkits make it possible to create sophisticated Graphical User Interfaces (GUIs).

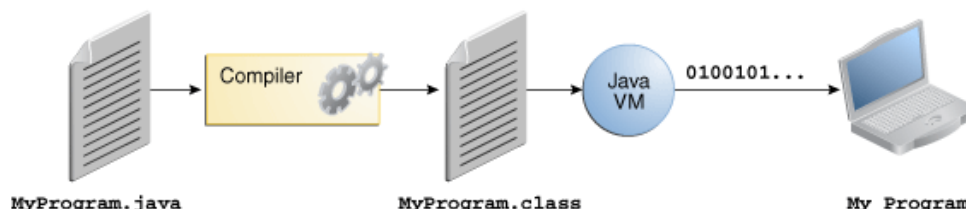
**Integration Libraries:** Integration libraries such as the Java IDL API, JDBC API, Java Naming and Directory Interface (JNDI) API, Java RMI, and Java Remote Method Invocation over Internet Inter-ORB Protocol Technology (Java RMI-IIOP Technology) enable database access and manipulation of remote objects.

#### D. Java Application

In the Java Language, all of the source code is written in plain text files with the .java extension name. The Java source code files are then compiled into .class extension files by the command javac. A .class file contains bytecode, which is a platform-independent instruction set. The java command then runs the application.

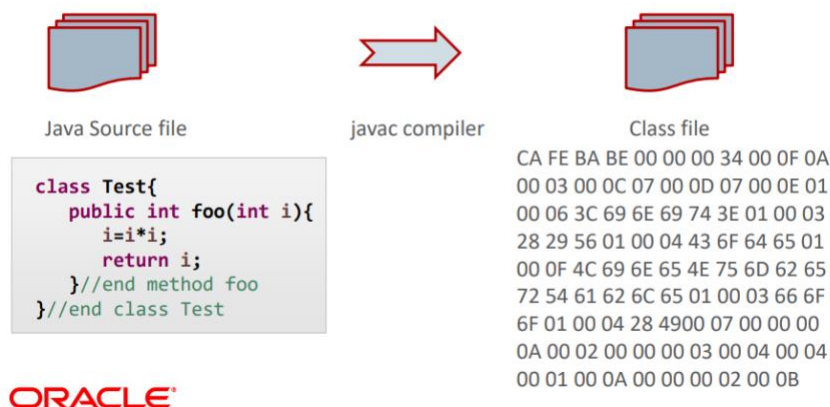


Translation from source code to byte code procedure



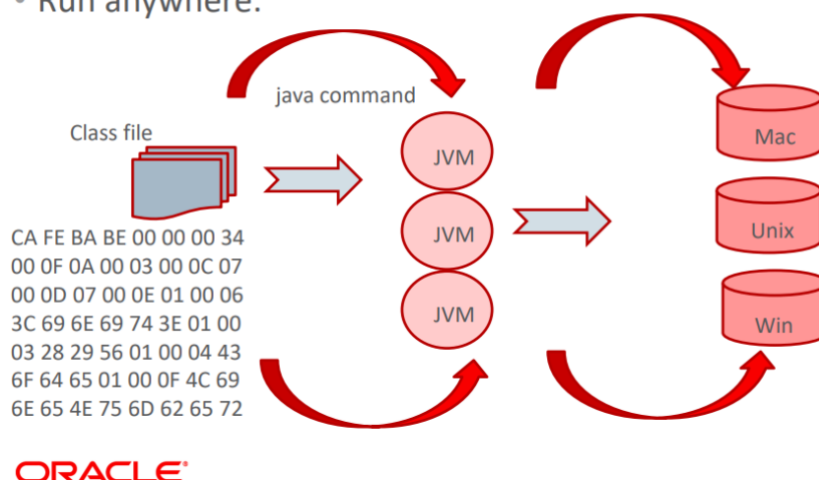
Because the Java VM is available on many different operating systems, the same .class files are capable of running on Microsoft Windows, the Solaris™ Operating System (Solaris OS), Linux, or Mac OS.

- Write once:



ORACLE®

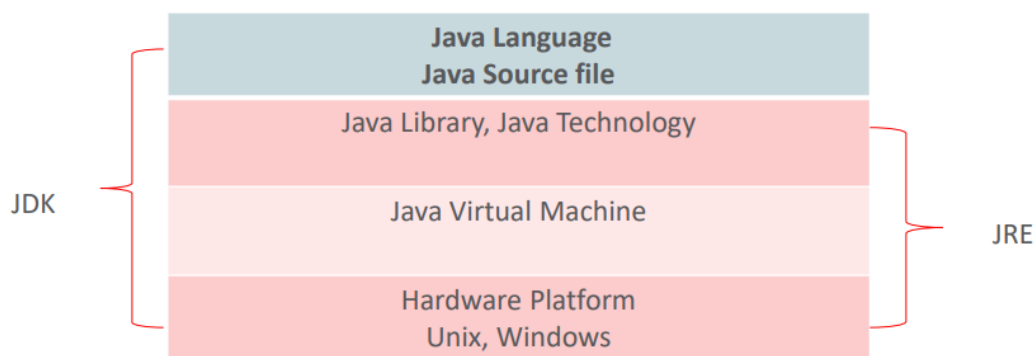
• Run anywhere:



Write once, run anywhere (WORA), or sometimes Write once, run everywhere (WORE), was a slogan to illustrate the cross-platform benefits of the Java language. Ideally, this meant that a Java program could be developed on any device, compiled into standard bytecode, and be expected to run on any device equipped with a Java virtual machine (JVM). The installation of a JVM or Java interpreter on chips, devices, or software packages became an industry standard practice.

### E. The Java SE Platform

Oracle has two products that implement the Java Platform Standard Edition, Java SE Development Kit (JDK) and Java SE Runtime Environment (JRE).



The Java SE Development Kit, or JDK, includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

The Java Runtime Environment, or JRE, is a software layer that runs on top of a computer's operating system software and provides the class libraries and other resources that a specific Java program needs to run.

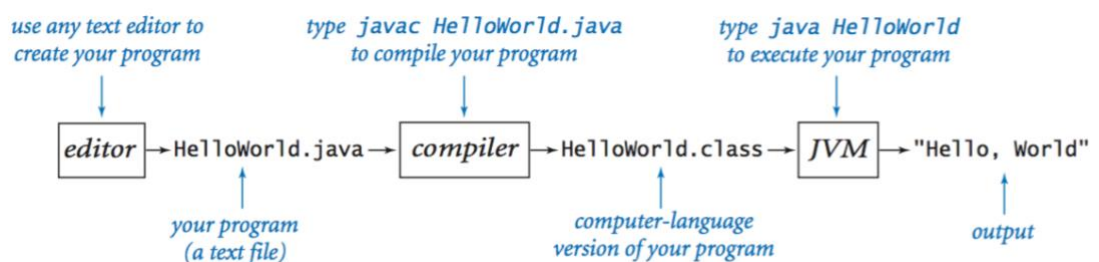
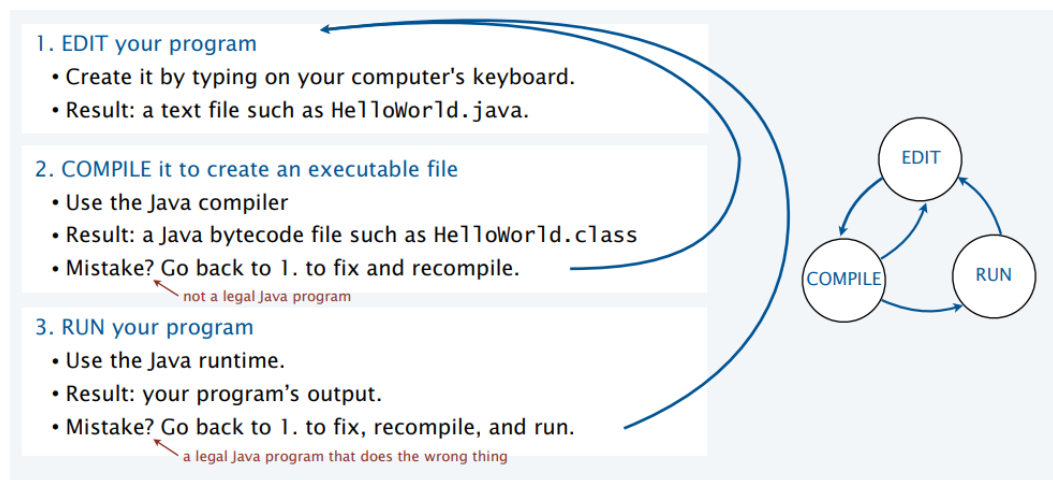


## F. Program Development in Java

We break the process of programming in Java into three steps:

1. Create the program by typing it into a text editor and saving it to a file named, say, MyProgram.java.
2. Compile it by typing "javac MyProgram.java" in the terminal window.
3. Execute (or run) it by typing "java MyProgram" in the terminal window.

The first step creates the program; the second translates it into a language more suitable for machine execution (and puts the result in a file named MyProgram.class); the third actually runs the program.



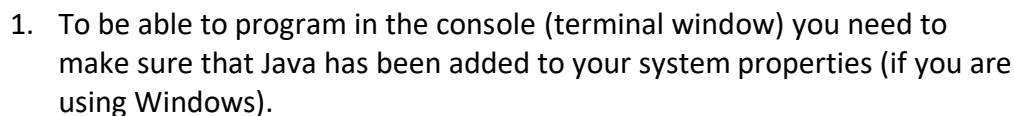
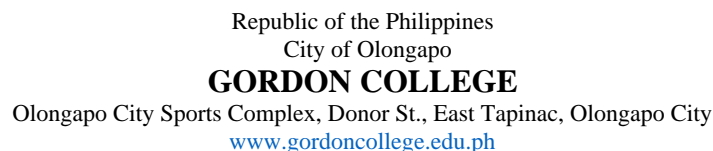
## G. Your First Java Program: The Hello World program

You are about to write your first Java program, but first you need to download and install the following software:

1. Java SE Development Kit (Java SE 15.0.2 is the latest release for the Java SE Platform). Download it [here](#).
2. Your preferred text editor (VS Code, Sublime, Atom, Notepad++, etc)

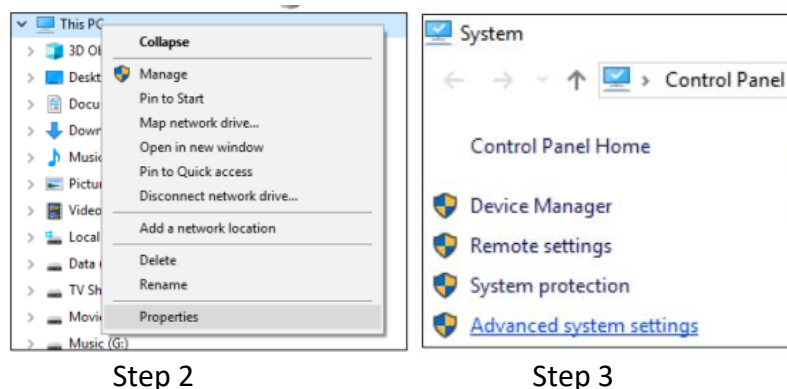
Follow these steps to check if you have Java installed in your system:





(Screenshot may differ to your device)

2. Right-click the PC icon and select properties to access the control panel home page on Windows.

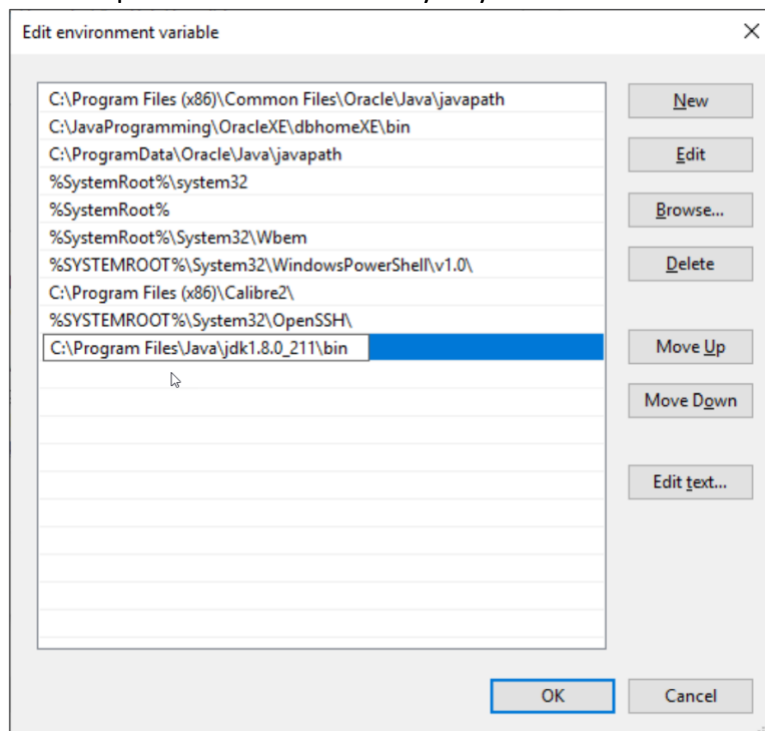


3. From the control panel select Advanced system settings from the menu.
4. Select the Environment Variables button from the System Properties dialog box.





5. Select the Path option within System Variables and then select Edit.
6. Click the New button.
7. Add the path to the bin directory of your JDK installation in the New field.



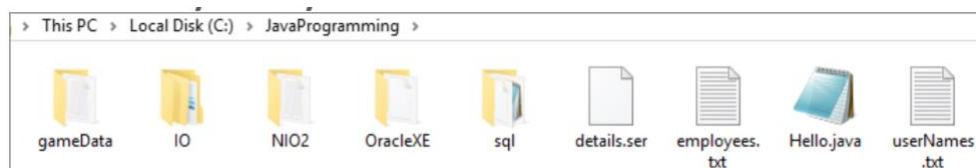
8. Click OK to add the path to your system.
9. Close all open system windows.

To write your first program, follow these steps:

1. Open a text editor and enter the following Java code:

```
public class Hello{  
    public static void main(String[] args){  
        System.out.println("Hello welcome to console programming!");  
    } //end method main  
} //end class Hello
```

2. Save the file as Hello.java in the JavaProgramming directory on your C Drive.



3. Open a command prompt and enter the following command to move to the JavaProgramming directory.



```
Command Prompt

C:\>cd C:\JavaProgramming

C:\JavaProgramming>_
```

4. Type the following command to check the version of Java that is running on your system.

```
C:\JavaProgramming>java -version
java version "1.8.0_211"
Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.211-b12, mixed mode)
```

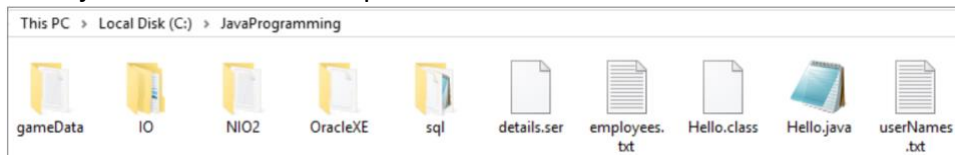
5. Compile the source code to generate the Class file.

```
C:\JavaProgramming>javac Hello.java

C:\JavaProgramming>
```

There is no successful compilation message although you will get an error if it does not compile!

6. A Java class file (Hello.class) will now have been created beside the Hello.java file that was compiled.



7. Run the Hello application by using the java command followed by the name of the class file (you do not need to add the file extension).

```
Command Prompt

C:\JavaProgramming>javac Hello.java

C:\JavaProgramming>java Hello
Hello welcome to console programming!

C:\JavaProgramming>_
```

This is the same process that happens when you click the run button in your IDE (Netbeans/Eclipse)!



## Teaching and Learning Materials Resources

- PC Computer | Laptop | Android Phone
- Gordon College LAMP
- Google Meet
- Facebook Messenger

## Learning Tasks

### A. Explore (50 points)

## LABORATORY ACTIVITY

1. Based on the process of developing a program in Java, please follow the steps below:
  - a. **Creating a Java program.** A program is nothing more than a sequence of characters, like a sentence, a paragraph, or a poem. To create one, we need only define that sequence characters using a text editor in the same way as we do for email. Type these characters into your text editor and save it into a file named HelloWorld.java in your JavaProgramming directory.

```
public class HelloWorld {  
    public static void main(String[] args) {  
        // Prints "Hello, World" in the terminal window.  
        System.out.println("Hello, World");  
    }  
}
```

Your code here: (Provide screenshot)

- b. **Compiling a Java program.** A compiler is an application that translates programs from the Java language to a language more suitable for executing on the computer. It takes a text file with the .java extension as input (your program) and produces a file with a .class extension (the computer-language version). To compile HelloWorld.java, use the javac command in the terminal window.



If you typed in the program correctly, you should see no error messages. Otherwise, go back and make sure you typed in the program exactly as it appears above.

Your terminal window (console) here: (Provide screenshot)

- c. **Executing (or running) a Java program.** Once you compile your program, you can execute it. This is the exciting part, where the computer follows your instructions. To run the HelloWorld program, use the java command in the terminal window.

If all goes well, you should see the following response:

```
Hello, World
```

Your terminal window (console) here: (Provide screenshot)

2. Typically, we want to provide **input** to our programs: data that they can process to produce a result. The simplest way to provide input data is illustrated in given source code below (UseArgument.java). Whenever this program is executed, it reads the command-line argument that you type after





the program name and prints it back out to the terminal as part of the message.

```
/******  
 * Compilation:  javac UseArgument.java  
 * Execution:    java UseArgument yourname  
 *  
 * Prints "Hi, Bob. How are you?" where "Bob" is replaced by the  
 * command-line argument.  
 *  
 * % java UseArgument Bob  
 * Hi, Bob. How are you?  
 *  
 * % java UseArgument Alice  
 * Hi, Alice. How are you?  
 *  
 *****/  
  
public class UseArgument {  
  
    public static void main(String[] args) {  
        System.out.print("Hi, ");  
        System.out.print(args[0]);  
        System.out.println(". How are you?");  
    }  
  
}
```

Sample Run:

```
% javac UseArgument.java  
% java UseArgument Alice  
Hi, Alice. How are you?  
% java UseArgument Bob  
Hi, Bob. How are you?
```

(We use the % symbol to denote the command prompt, but it may appear different depending on your system.)

Your code here: (Provide screenshot)



Your terminal window (output) here: (Provide screenshot)

B. Explain (40 points) **LECTURE ACTIVITY**

1. Describe what happens, in HelloWorld.java, if you omit:

a. main

b. String

c. HelloWorld

d. System.out

e. println



2. Describe what happens if, in HelloWorld.java, you omit:

a. the ;

b. the first "

c. the second "

d. the first {

e. the second {

f. the first }

g. the second }



3. Describe what happens if, in HelloWorld.java, you misspell (by, say, omitting the second letter)

a. main

b. String

c. HelloWorld

d. System.out

e. println

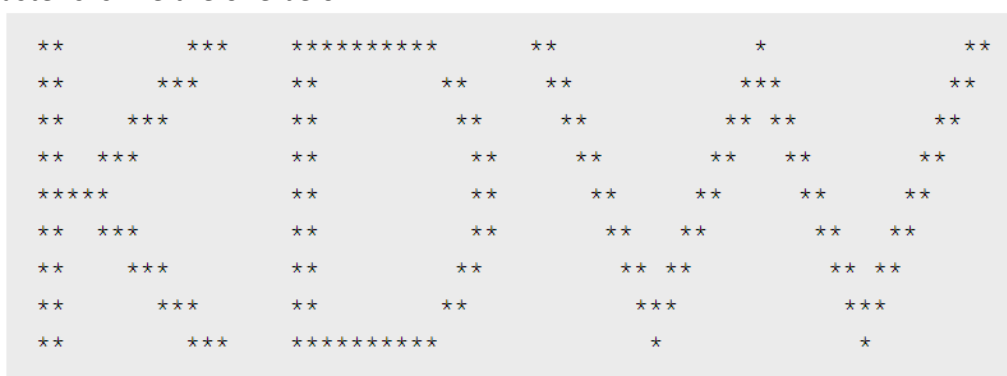
4. I typed in the following program. It compiles fine, but when I execute it, I get the error java.lang.NoSuchMethodError: main. What am I doing wrong?

```
public class Hello {  
    public static void main() {  
        System.out.println("Doesn't execute");  
    }  
}
```



C. Engage (60 points) **LABORATORY ACTIVITY**

1. Write a program Initials.java that prints your initials using nine rows of asterisks like the one below.



Your code here: (Provide screenshot)

Your terminal window (output) here: (Provide screenshot)





2. Write a program `TenHelloWorlds.java` that prints "Hello, World" ten times.

Your code here: (Provide screenshot)

Your terminal window (output) here: (Provide screenshot)

3. Modify `UseArgument.java` to make a program `UseThree.java` that takes three names and prints out a proper sentence with the names in the reverse of the order given, so that for example, "java UseThree Alice Bob Carol" gives "Hi Carol, Bob, and Alice."

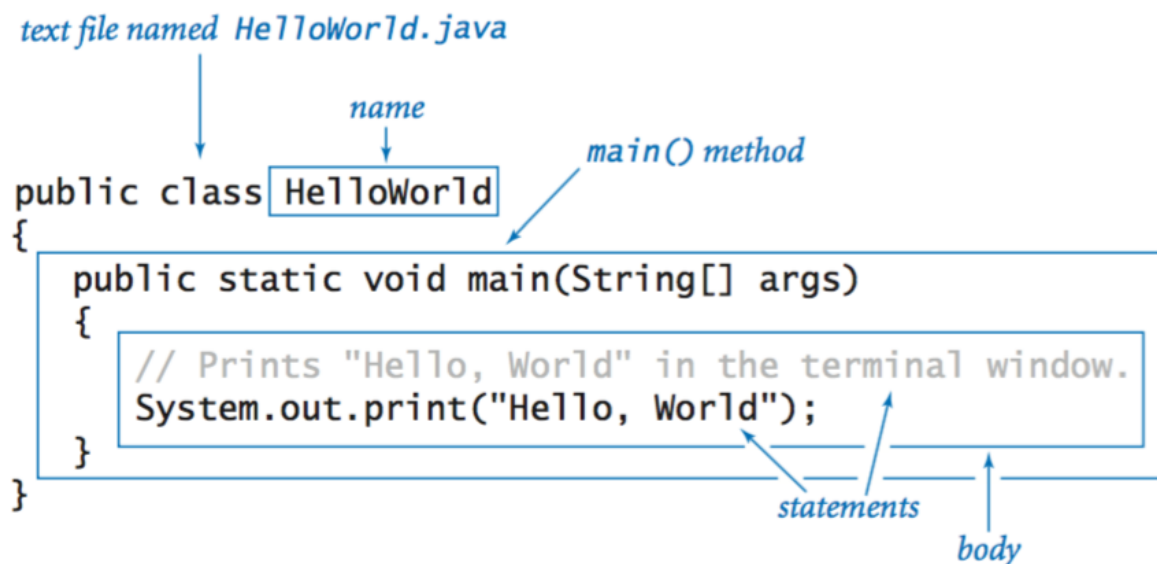
Your code here: (Provide screenshot)



Your terminal window (output) here: (Provide screenshot)

### Additional Information:

The Anatomy of the HelloWorld.java program:



### References

- Oracle. nd. "Oracle Java Documentation". <https://docs.oracle.com/javase/tutorial/>
- Oracle Academy, "Java Programming Instructor Resources". <https://academy.oracle.com>