

Lab 1: Introduction to JAVA.

Objective(s):

1. What is Java?
2. Features and applications of Java.
3. JAVA Basics.
4. Installing JDK.
5. Writing HelloWorld.java in Text Editor
6. JAVA variables & data types.
7. Input & Output
8. Java Variable Type Conversion & Type Casting

1: What is JAVA?

Java was developed by Sun Microsystems in 1995, but it has stood the test of time and remains highly relevant and widely used to this day. Java is a high-level, general-purpose object-oriented programming language.

[History of JAVA](#)

Java Versions

There are many java versions that has been released. Current stable release of Java is Java SE 8.

1. JDK Alpha and Beta (1995)
2. JAVA 1.0 (23rd Jan, 1996)
3. JAVA 1.1 (19th Feb, 1997)
4. JAVA 1.2 (8th Dec, 1998)
5. JAVA 1.3 (8th May, 2000)
6. JAVA 1.4 (6th Feb, 2002)
7. JAVA 5.0 (30th Sep, 2004)
8. JAVA 6 (11th Dec, 2006)
9. JAVA 7 (28th July, 2011)
10. JAVA 8 (18th March, 2014)
11. JAVA 9 (21th Sep, 2017)

12. JAVA 10 (20th March, 2018)
13. JAVA 11 (25th Sep 2018)
14. JAVA 12 (19th March 2019)
15. JAVA 13 (17th Sep 2019)

Java Platforms

According to Oracle, there are four platforms of the Java programming language

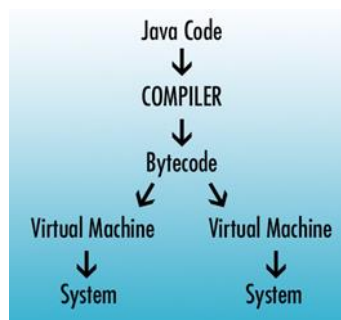
- Java Platform, Standard Edition (Java SE)
- Java Platform, Enterprise Edition (Java EE)
- Java Platform, Micro Edition (Java ME)
- JavaFX

2: Features and applications of JAVA

Features of Java

The following are Java's main features:

- **High level and general purpose:** Rather than being created to accomplish one very specific task, Java allows us to write computer-readable instructions in an open-ended environment. Because it's not really feasible, or even desirable, for every computer system to have its own specialized programming language, the vast majority of the code is written in high-level, general-purpose languages such as Java.
- **Object-oriented:** Java is also what we call an object-oriented language. While we won't get into the specifics of objects and classes until a bit later in this book, know for now that objects allow us to define modular entities within our program that make them much more human-readable and much more manageable to create large-scale software projects. A firm grasp of object-oriented concepts is absolutely essential for any modern software developer.
- **Platform-independent:** Lastly, Java was designed with the intention that it be a write once, run anywhere language. This means if you and I both have systems with Java installed and even if our systems are not normally identical--for example, I'm on a Windows machine and you're on a Mac--a Java program on my machine that I give to you will still run essentially the same on your machine without the need for it to be recompiled.



Note: Compiling a programming language such as JAVA is the act of taking the human-readable code that we've written and converting it into an interpreted machine-friendly code. Unfortunately, it's usually not very friendly for humans to read or write. To do this, we use a program called the compiler that takes in our code as text and converts it into machine code.

Traditionally, we would have to recompile a program for every system that it was going to run on because all systems have a different idea of what their machine code should look like. Java circumvents this issue by compiling all Java programs to the same type of interpreted code called bytecode.

A compiled Java program in bytecode can be run by any system in which Java is installed. This is because when we install Java on your system, we also install a Java virtual machine with it that's specific to that system. It is this machine's responsibility to convert the bytecode into the final instructions that head to the processor in that system.

By making it the system's responsibility to do this final conversion, Java has created a write once, run anywhere language where I can hand you a Java program and you can run it on your machine while being fairly certain that it's going to run in the same manner that it did on mine. This impressive level of cross-platform support on a language as powerful as Java has made it one of the software developing world's go-to tools for quite some time.

“Write once, run everywhere”

JAVA Applications:

In today's modern times, Java is used to develop desktop applications, web servers, and client-side web applications. It's the native language of the Android operating system, which operates on Android phones and tablets.

Java has been used to write video games and is sometimes even ported to smaller devices without a traditional operating system. It remains a huge player in today's technical world.

3: JAVA Basics

JDK stand for **Java Development Kit**, is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in Java development.

JRE stands for **Java Runtime Environment**. The Java Runtime Environment provides the minimum requirements for executing a Java application; it consists of the *Java Virtual Machine (JVM)*, *core classes*, and *supporting files*.

JVM: A **Java Virtual Machine (JVM)**, an implementation of the Java Virtual Machine Specification, interprets compiled Java binary code (called bytecode) for a computer's processor (or "hardware platform") so that it can perform a Java

program's instructions. Java was designed to allow application programs to be built that could be run on any platform without having to be rewritten or recompiled by the programmer for each separate platform.

JIT Just-in-time Compiler is the part of the Java Virtual Machine (JVM) that is used to speed up the execution time. JIT interprets parts of the bytecode that have similar functionality at the same time, and hence reduces the amount of time needed for full interpretation.

4: Installing JDK

Java Development Kit

To develop Java applications on our computers, we require a JDK. Visit the link below to download the JDK setup.

[Download JDK](#)

Java SE Development Kit 13.0.1

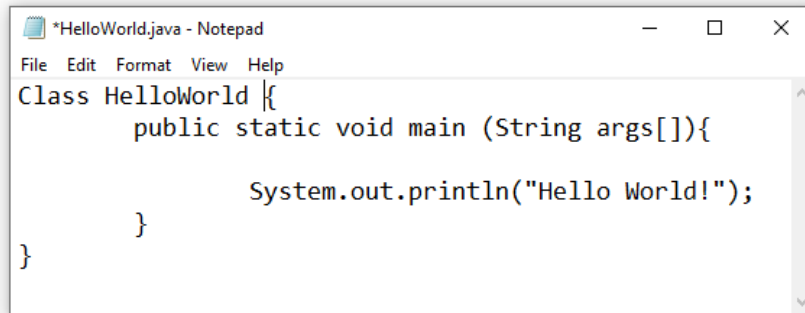
You must accept the [Oracle Technology Network License Agreement for Oracle Java SE](#) to download this software.

☐ Accept License Agreement ☒ Decline License Agreement

Product / File Description	File Size	Download
Linux	155.88 MB	jdk-13.0.1_linux-x64_bin.deb
Linux	163.17 MB	jdk-13.0.1_linux-x64_bin.rpm
Linux	180 MB	jdk-13.0.1_linux-x64_bin.tar.gz
macOS	172.78 MB	jdk-13.0.1_osx-x64_bin.dmg
macOS	173.11 MB	jdk-13.0.1_osx-x64_bin.tar.gz
Windows	159.84 MB	jdk-13.0.1_windows-x64_bin.exe
Windows	178.99 MB	jdk-13.0.1_windows-x64_bin.zip

5: Writing HelloWorld.java in Text Editor

1. **Create a source file** - A source file contains code, written in the Java programming language, that you and other programmers can understand. You can use any text editor to create and edit source files. Run notepad and enter your code. Save this file with the .java extension.

A screenshot of a Notepad window titled '*HelloWorld.java - Notepad'. The window contains the following Java code:

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

2. **Compile the source file into a .class file** - Open Command Prompt and enter **javac filename.java**. The Java programming language *compiler* (javac) takes your source file and translates its text into instructions known as *bytecodes*. Next, enter **java ClassName**. The Java application *launcher tool* (java) uses the Java virtual machine to run your application.

6: JAVA Variables & Data Types

Types of variables

In Java, there are three types of variables:

1. Local Variables
2. Instance Variables
3. Static Variables

1. Local Variables

Local Variables are a variable that are declared inside the body of a method.

2. Instance Variables

Instance variables are defined without the STATIC keyword. They are defined outside a method declaration. They are Object specific and are known as instance variables.

3. Static Variables

Static variables are initialized only once, at the start of the program execution. These variables should be initialized first, before the initialization of any instance variables.

Example: Types of Variables in Java

```
class Variables {  
    int data = 99; //instance variable
```

```
static int a = 1; //static variable
void method() {
    int b = 90; //local variable
}
```

Data Types in Java

Data types classify the different values to be stored in the variable. In java, there are two types of data types:

1. Primitive Data Types
2. Non-primitive Data Types

Primitive Data Types

Primitive Data Types are predefined and available within the Java language. Primitive values do not share state with other primitive values.

There are 8 primitive types: byte, short, int, long, char, float, double, and boolean

Integer data types

```
byte (1 byte)
short (2 bytes)
int (4 bytes)
long (8 bytes)
```

Floating Data Type

```
float (4 bytes)
double (8 bytes)
```

Textual Data Type

```
char (2 bytes)
```

Logical

```
boolean (1 bit) (true/false)
```

Lab 7: Input & Output

Output in Java Syntax:

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

Input in Java Syntax:

```
// import library
import java.util.Scanner;

// Creating scanner object
Scanner obj = new Scanner(System.in);

Taking input from user
int num = obj.nextInt(); // Integer Input
double num = obj.nextDouble(); // Double Input
```

Lab 8: Java Variable Type Conversion & Type Casting

A variable of one type can receive the value of another type. Here there are 2 cases.

Case 1) Variable of smaller capacity is be assigned to another variable of bigger capacity.

```
double d ;
int i = 10;
d = i;
```

This process is Automatic, and non-explicit is known as *Conversion*

Case 2) Variable of larger capacity is be assigned to another variable of smaller capacity.

```
double d = 10;
int i;
i = (int) d
```

Type Cast
Operator

In such cases, you have to explicitly specify the **type cast operator**. This process is known as *Type Casting*.

In case, you do not specify a type cast operator; the compiler gives an error. Since this rule is enforced by the compiler, it makes the programmer aware that the conversion he is about to do may cause some loss in data and prevents **accidental losses**.

Example: To Understand Type Casting

```
class Demo {
    public static void main(String args[]) {
        byte x;
        int a = 270;
        double b = 128.128;
        System.out.println("int converted to byte");
        x = (byte) a;
        System.out.println("a and x " + a + " " + x);
        System.out.println("double converted to int");
        a = (int) b;
        System.out.println("b and a " + b + " " + a);
        System.out.println("\ndouble converted to byte");
        x = (byte)b;
        System.out.println("b and x " + b + " " + x);
    }
}
```

Output:

```
int converted to byte
a and x 270 14
double converted to int
b and a 128.128 128

double converted to byte
b and x 128.128 -128
```

Lab 9: if/else, loops

If/Else in Java Syntax

if(condition1)


```

{
    //code to be executed if condition1 is true
}
else if(condition2)
{
    //code to be executed if condition2 is true
}
else if(condition3)
{
    //code to be executed if condition3 is true
}
...
else
{
    //code to be executed if all conditions are false
}

```

For Loop Syntax in Java

```

for(initialization;condition;incr/decr){
    //statement or code to be executed
}

```

Lab Tasks:

Exercise 1 (JAVA Environment Installation & Error Messages)

1. Set up a Java development environment. In the *main()* method of your program try to compile the following invalid Java code snippets. Record the error messages you receive. What do you think each error message indicates?

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

```
System.out.println(Hello World)
```

```
System.out.println"Hello World";
```

```
println("Hello World");
```

2. To generate one final error message, remove one of the brackets from the end of your program. Now what message do you receive?

Exercise 2 (Mathematical Expressions)

Write Java code for following mathematical expressions to compute the result.

```
f = 5 * 7 / 3
```

```
float f = (5 * 7) / 3.0f
```

```
int x = 5 + 'a'
```

```
int x = 2147483647 + 1
```

Exercise 3 (Java Variables)

Write a Java code in which create all of the primitives (except long and double) with different values. Concatenate them into a string and print it to the screen so it will print:

H3110 w0rld 2.0 true

Exercise 4 (Operators)

1. Write Java program to allow the user to input two integer values and then the program prints the results of adding, subtracting, multiplying, and dividing among the two values. See the example below:

```
Enter value a:30
```

```
Enter value b:10
```

```
The result of adding is 40.
```

```
The result of subtracting is 20;
```

```
The result of multiplying is 300.
```

```
The result of dividing is 3.
```

2. Change the following program to use compound assignments:

```

class ArithmeticDemo {
    public static void main (String[] args){
        int result = 1 + 2; // result is now 3
        System.out.println(result);

        result = result - 1; // result is now 2
        System.out.println(result);

        result = result * 2; // result is now 4
        System.out.println(result);

        result = result / 2; // result is now 2
        System.out.println(result);

        result = result + 8; // result is now 10
        result = result % 7; // result is now 3
        System.out.println(result);
    }
}

```

Exercise 5 (if/else & loops)

- Given Boolean variables A, B, and C, write if statements to check for the following conditions:
 - A and B, but not C
 - Either A, B, or C
 - All or none of A, B, and C
 - A and B
- Using a while loop, write Java code which will multiply all the integers from 1 to a given value. Now, utilize a for loop to generate the same functionality.