Lab 01
Introduction to Java programming

Objectives:

1. What is JAVA?
2. Features of Java.
3. JAVA Basics.
4. Installing JDK and setting path.
5. Writing HelloWorld.java in Text Editor
6. JAVA variables & data types.
7. Output
8. Java Variable Type Conversion & Type Casting
9. Lab tasks
10. Post Lab Questions

1: **What is JAVA?**

Java was developed by Sun Microsystems in 1995. It is a **programming language** as well as **platform.** Java is among the most popular programming languages out there, mainly because of how versatile and compatible it is. Java is general purpose programming language as it is used for software development, mobile applications, web servers, and client-side web applications. It is the native language of the Android operating system, which operates on Android phones and tablets.

Versions of Java

There are many java versions that has been released.

JDK Alpha and Beta (1995)

JAVA 1.0 (23rd Jan, 1996)

JAVA 1.1 (19th Feb, 1997)

JAVA 1.2 (8th Dec, 1998)

JAVA 1.3 (8th May, 2000)

JAVA 1.4 (6th Feb, 2002)
JAVA 5.0 (30th Sep, 2004)
JAVA 6 (11th Dec, 2006)
JAVA 7 (28th July, 2011)
JAVA 8 (18th March, 2014)
JAVA 9 (21th Sep, 2017)
JAVA 10 (20th March, 2018)
JAVA 11 (25th Sep 2018)
JAVA 12 (19th March 2019)
JAVA 13 (17th Sep 2019)
JAVA 14 (17th March 2020)
JAVA 15 (15th Sep 2020)
JAVA 16 (16th March 2021)

Platform: Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

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/OpenJFX

2: **Features of JAVA**

The following are Java's main features:

- **Object Oriented** − In Java, everything is an object, creating objects that contain data and methods.
- **Architecture-neutral** − 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 compiler generates an architecture-neutral object file format called as bytecode, which makes the compiled code executable on many machines but with the presence of Java runtime system.

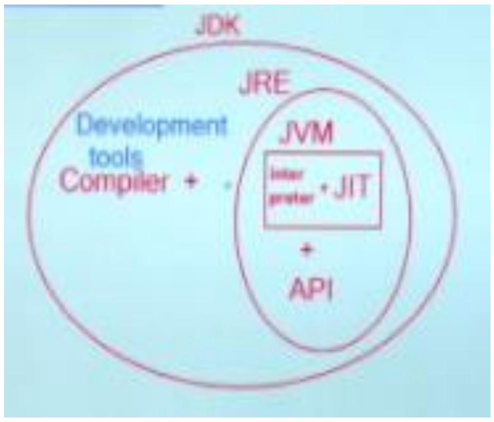
- **Platform Independent** − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

- **Portable** − Bytecode can be run by any system in which Java is installed. This is because when java is installed, Java virtual machine is also installed that is specific to that system. It is this machine's responsibility to convert the bytecode into the final instructions of that particular machine.
  By making it the system's responsibility to do this final conversion, Java has created a write once, run anywhere language where anyone can hand you a Java program and you can run it on your machine
  "Write once, run everywhere"

3: **JAVA Basics**

JDK

- It stands for Java Development Kit, is a software development environment used for developing Java applications and applets.
- It compiles and executes new and already built applications.
- It is a collection of development tools as well as 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

- It 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), interpreter, JIT, core classes, and supporting files.

JVM

- It stands for Virtual Machine (JVM)
- It is responsible for executing bytecode where interpreter provides machine code for the current machine and has JIT as well.

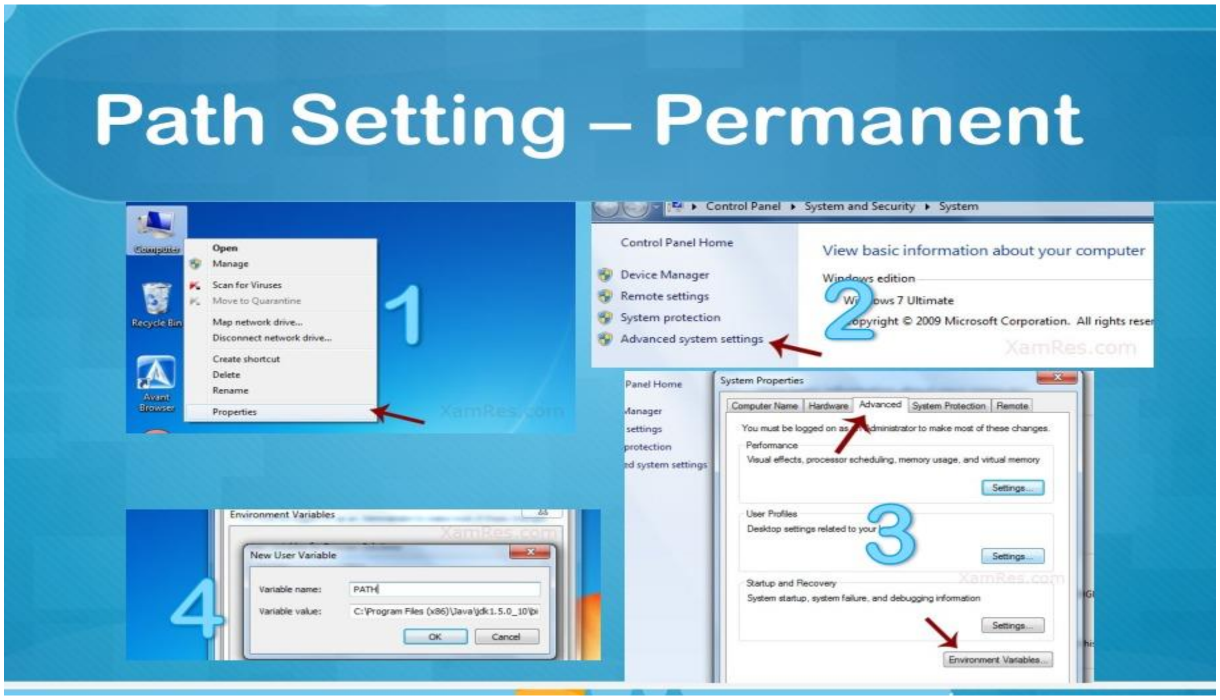
JIT

- It stands for Just-in-time Compiler, is the part of the Java Virtual Machine (JVM) that is used to speed up the execution time.
- JIT interpret 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 and setting path**

To develop Java applications on our computers, we require a JDK. Visit the link below to download the JDK setup.
https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe

After successful installation, path is to be set which can be temporary through command line or the permanent path which can be set using properties of 'this PC', go to advanced system setting and then Environment variable.



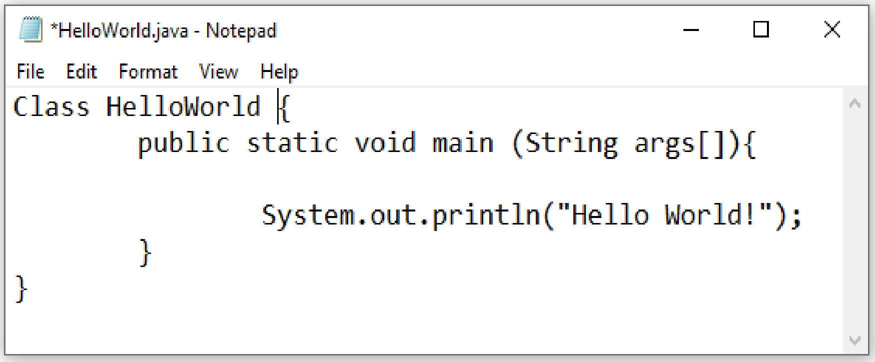
Path Setting – Permanent

Variable name: PATH
Variable value: C:\Program Files (x86)\Java\jdk1.5.0_10\bi

5: **Writing HelloWorld.java in Text Editor**

Follow the below given steps:

i. Run notepad and enter below given code. Save this file with Class name and end it with ".java" extension. Save it on desktop for now!



```
Class HelloWorld {
        public static void main (String args[]){

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

ii. Go to search bar in taskbar, write cmd and Open Command Prompt then write the following commands;
   a. cd desktop //for going to desktop
   b. javac HelloWorld.java
   c. java HelloWorld

The Java programming language compiler (javac) takes your source file and translates the code into instructions known as bytecodes. "Java ClassName" will enable Java virtual machine to run your application/code.

6: **Variables and data types in JAVA**

In Java, there are three types of variables:

- Local Variables
- Instance Variables
- Static Variables

Local Variables

Local Variables are declared inside the body of a method.

Scope: Variables declared inside a method have method level scope and cannot be accessed outside the method

Instance Variables

Instance variables are defined without the ‘static’ keyword . They are defined outside a method within a class. Access modifiers can be given for instance variables. They are Object specific.

Scope: Dependent on the access modifier

Class/Static Variables

It is declared with the keyword ‘static’, outside the method within a class. Static variables are created when the program starts and destroyed when the program stops. There would only be one copy of each static variable per class, regardless of how many objects are created.

Scope: Visibility is like instance variables. However, most static variables are declared public since they must be available for users of the class

<u>Example: Types of Variables in Java</u>

```
class Variab {
   int InsVarExam = 29; //instance variable
   static int IsStatVar = 15; //static variable
   void method() {
      int IsLocalVar = 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:

- Primitive Data Types
- Non-primitive Data Types

Non primitive as arrays, strings

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)

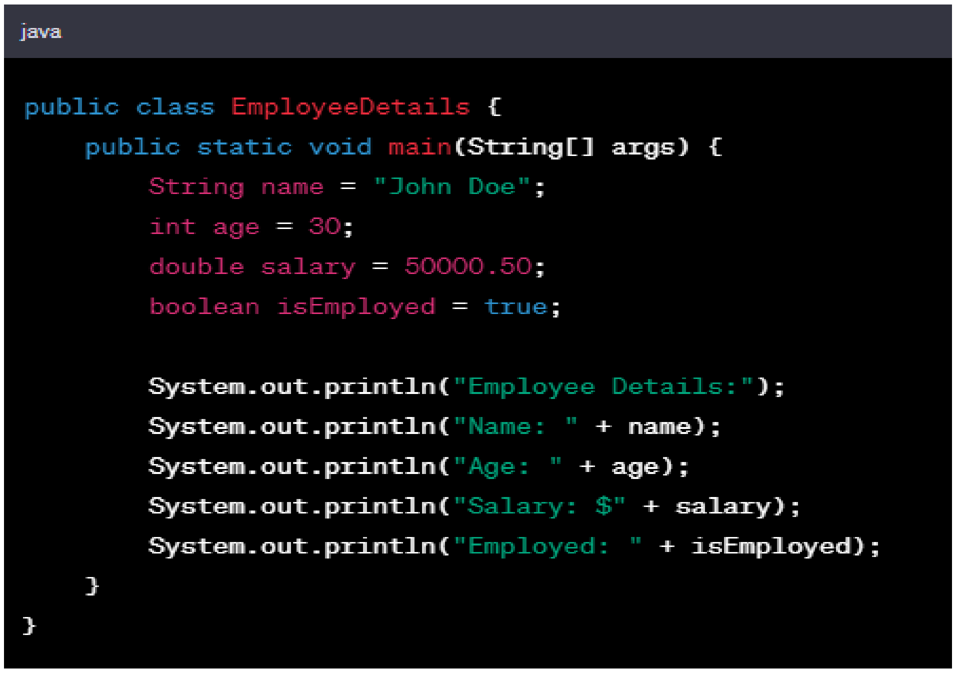
double (8 bytes)

Textual Data Type

char (2 bytes)

Logical

boolean (1 bit) (true/false)



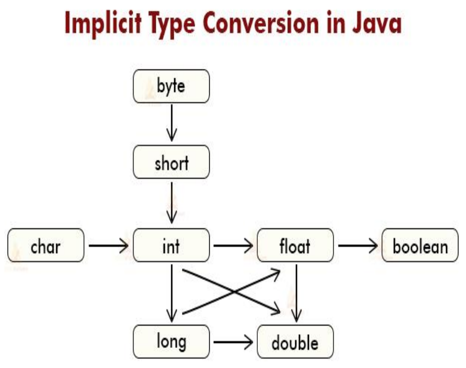
```java
public class EmployeeDetails {
    public static void main(String[] args) {
        String name = "John Doe";
        int age = 30;
        double salary = 50000.50;
        boolean isEmployed = true;

        System.out.println("Employee Details:");
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Salary: $" + salary);
        System.out.println("Employed: " + isEmployed);
    }
}
```

7: Java Variable Type Conversion & Type Casting

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

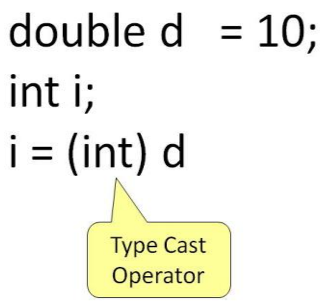
Case 1:Variable of smaller capacity is be assigned to another variable of bigger capacity.

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



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

Example1: 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
double converted to byte
b and x 128.128 -128
```

Example2: To Understand Type Casting

```
public class AverageCalculator {
    public static void main(String[] args) {
        // Declare subject scores
        int subject1 = 85;
        int subject2 = 90;
        int subject3 = 78;

        // Calculate the average
        double average = (subject1 + subject2 + subject3) / 3.0;  // Explicit type casting to ensure
floating-point division

        // Display the result
        System.out.println("Average (rounded down): " + (int)average);
        System.out.println("Average (floating-point): " + average);
    }
}
```

Lab Tasks

Exercise 1 (JAVA Environment Installation & Error Messages)

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.printn("Hello World")
```

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

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

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

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 to identify if the given input by the user is even or odd.
- Write a Java program to print the results of the following operations.
  Test Data:
  a. -5 + 8 * 6
  b. (55+9) % 9
  c. 20 + -3*5 / 8
  d. 5 + 15 / 3 * 2 - 8 % 3
- Write Java code for following mathematical expressions to compute the result.

a) f = 5 * 7 / 3

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

c) int x = 5 + 'a'

d) int x = 2147483647 + 1

Exercise 3 (Type casting)

- Perform division using two double variables and store the result in int variable and print the results
- Calculate the area of a circle and print the result as an integer value.
- Declare a variable of char data type and assign 'B' to it. Then typecast this variable to integer and display the original and converted value.

Post lab questions to ponder

1. Can you cast string into int? If so, How?
2. Why JAVA when there are other OOP languages?