**Experiment No 1**

**Aim: - Implement Basic Constructs/Notions like Constants, variables and data types, Operators and Expressions, Branching and looping in Java**

**Theory: -**

**1. What is Java?**

Java is a popular programming language created in 1995. Java is used to develop mobile apps, web apps, desktop apps, games, Database connection **and much more.**

**Features of Java Programming Language**

1. **Object Oriented**
   In Java, everything is an Object.
2. **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 platform-independent 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.
3. **Simple**
   Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
4. **Secure**
   With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
5. **Architecture-neutral**
   Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
6. **Portable**
   Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. The compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
7. **Robust**
   Java makes an effort to eliminate error-prone situations by emphasizing mainly on compile time error checking and runtime checking.
8. **Multithreaded**
   With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
9. **Interpreted**
   Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
10. **High Performance**
    With the use of Just-In-Time compilers, Java enables high performance.
11. **Distributed**
    Java is designed for the distributed environment of the internet.

## 2. Compilation and Execution of a Java Program

Java, being a platform-independent programming language, doesn't work on the one-step compilation. Instead, it involves a two-step execution, first through an OS-independent compiler; and second, in a virtual machine (JVM) which is custom-built for every operating system.

**1. Compilation: -**
First, the source '.java' file is passed through the compiler, which then encodes the source code into a machine-independent encoding, known as Bytecode. The content of each class contained in the source file is stored in a separate '.class' file.

**2. Execution**

The class files generated by the compiler are independent of the machine or the OS, which allows them to be run on any system. To run, the main class file (the class that contains the method main) is passed to the JVM and then goes through three main stages before the final machine code is executed. These stages are:
These states do include:

A. Class Loader
B. Bytecode Verifier
C. Just-In-Time Compiler

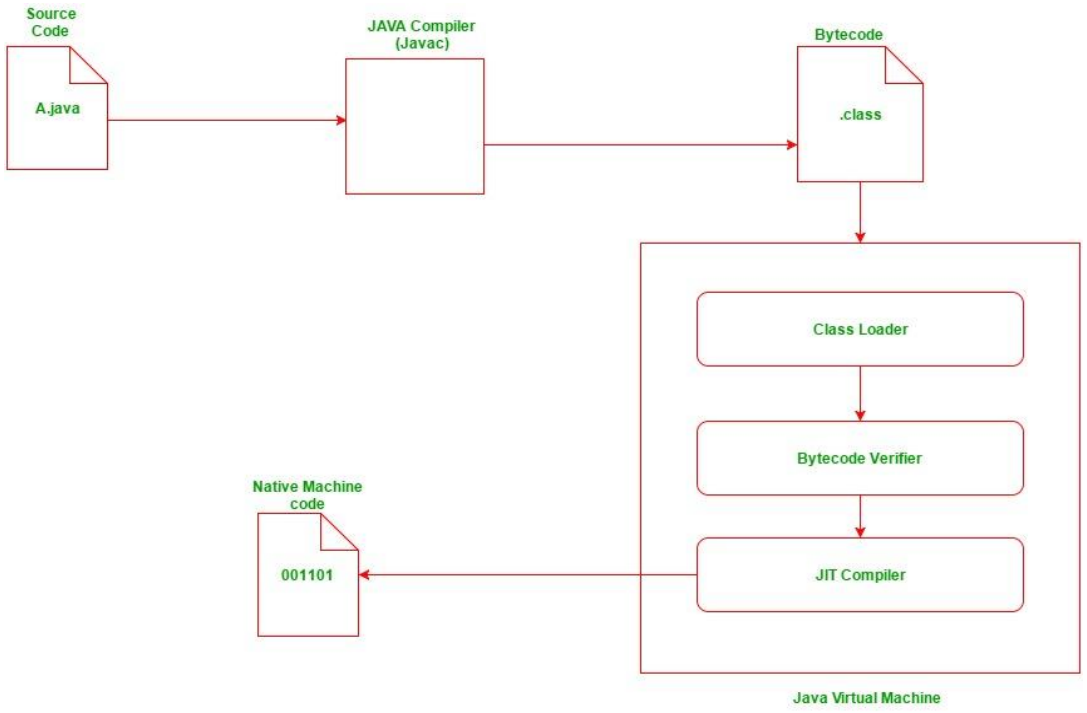
**A. Class Loader**
The main class is loaded into the memory bypassing its '.class' file to the JVM, through invoking the latter. All the other classes referenced in the program are loaded through the class loader. A class loader, itself an object, creates a flat namespace of class bodies that are referenced by a string name.

**B. Bytecode Verifier**
After the bytecode of a class is loaded by the class loader, it has to be inspected by the bytecode verifier, whose job is to check that the instructions don't perform damaging actions. The following are some of the checks carried out:

Variables are initialized before they are used.
Method calls match the types of object references.
Rules for accessing private data and methods are not violated.
Local variable accesses fall within the runtime stack.
The run-time stack does not overflow.
If any of the above checks fail, the verifier doesn't allow the class to be loaded.

**C. Just-In-Time Compiler**

This is the final stage encountered by the java program, and its job is to convert the loaded bytecode into machine code. When using a JIT compiler, the hardware can execute the native code, as opposed to having the JVM interpret the same sequence of bytecode repeatedly and incurring the penalty of a relatively lengthy translation process. This can lead to performance gains in the execution speed unless methods are executed less frequently.




### 3. Steps to install Java

To check if you have Java installed on a Windows PC, search in the start bar for Java or type the following in Command Prompt (cmd.exe):

C:\Users\Your Name>java –version

If Java is installed, you will see something like this (depending on version):

```
java version "11.0.1" 2018-10-16 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.1+13-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.1+13-LTS, mixed mode)
```

If you do not have Java installed on your computer, you can download it for free at oracle.com.

**Setup for Windows**

To install Java on Windows:

1. Go to "System Properties" (Can be found on Control Panel > System and Security > System > Advanced System Settings)
2. Click on the "Environment variables" button under the "Advanced" tab
3. Then, select the "Path" variable in System variables and click on the "Edit" button
4. Click on the "New" button and add the path where Java is installed, followed by **\bin**. By default, Java is installed in C:\Program Files\Java\jdk-11.0.1 (If nothing else was specified when you installed it). In that case, You will have to add a new path with: **C:\Program Files\Java\jdk-11.0.1\bin**
   Then, click "OK", and save the settings
5. At last, open Command Prompt (cmd.exe) and type **java -version** to see if Java is running on your machine

## 4. Java Syntax

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

**Explanation of above example**

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. Java is case-sensitive: "MyClass" and "myclass" has different meaning. The name of the java file must match the class name. When saving the file, save it using the class name and add ".java" to the end of the filename.

**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.

**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");
}
```

The curly braces {} marks the beginning and the end of a block of code. System is a built-in Java class that contains useful members, such as out, which is short for "output". The println () method, short for "print line", is used to print a value to the screen (or a file). Just know that you need them together to print stuff to the screen. Each code statement must end with a semicolon (;).

## 5. Java Variables

Variables are containers for storing data values. In Java, there are different types of variables, for example:

String - stores text, such as "Hello". String values are surrounded by double quotes
int - stores integers (whole numbers), without decimals, such as 123 or -123
float - stores floating point numbers, with decimals, such as 19.99 or -19.99
char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
Boolean - stores values with two states: true or false

**Declaring (Creating) Variables**

To create a variable, you must specify the type and assign it a value:

**Syntax**

type variableName = value;

Where type is one of Java's types (such as int or String), and variableName is the name of the variable (such as x or name). The equal sign is used to assign values to the variable.

To create a variable that should store text, look at the following example:

**Example**
Create a variable called name of type String and assign it the value "John":
String name = "John";
System.out.println(name);

**Final Variables**
If you don't want others (or yourself) to overwrite existing values, use the final keyword (this will declare the variable as "final" or "constant", which means unchangeable and read-only):

**Example**
final int myNum = 15;
myNum = 20; // will generate an error: cannot assign a value to a final variable

**Data types are divided into two groups**:

Primitive data types - includes byte, short, int, long, float, double, boolean and char
Non-primitive data types - such as String, Arrays and Classes

**Java Type Casting**
Type casting is when you assign a value of one primitive data type to another type.

**In Java, there are two types of casting:**

**Widening Casting (automatically)** - converting a smaller type to a larger type size
byte -> short -> char -> int -> long -> float ->

**double int myInt = 9;**
**double myDouble = myInt; // Automatic casting: int to double**

**Narrowing Casting (manually)** - converting a larger type to a smaller size type
double -> float -> long -> int -> char -> short -> byte

**double myDouble = 9.78d;**
**int myInt = (int) myDouble; // Manual casting: double to int**

**6. Java Operators**

Operators are used to perform operations on variables and values.

**Java divides the operators into the following groups:**

- **Arithmetic operators (+, -, *, /, %, ++, --)**
- **Assignment operators (=, +=, -=, *=, /=, %=)**
- **Comparison operators (==, !=, <=, >=, <, >)**
- **Logical operators (&&, ||,!)**
- **Bitwise operators**

**7. Java conditional statements:**

Use **if** to specify a block of code to be executed, **if a specified condition is true**
Use **else** to specify a block of code to be executed, **if the same condition is false**
Use **else if** to specify a new condition to test, **if the first condition is false**
Use **switch** to specify **many alternative** blocks of code to be executed

**The if Statement**
Use the if statement to specify a block of Java code to be executed if a condition is true.
**Syntax**

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

**The else Statement**
Use the else statement to specify a block of code to be executed if the condition is false.
**Syntax**

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

**The else if Statement**
Use the else if statement to specify a new condition if the first condition is false.

**Syntax**

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```

**Java Switch Statements**
Use the switch statement to select one of many code blocks to be executed.

**Syntax**

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

This is how it works:
The switch expression is evaluated once.
The value of the expression is compared with the values of each case.
If there is a match, the associated block of code is executed.
The break and default keywords are optional

**The break Keyword**

When Java reaches a break keyword, it breaks out of the switch block.
This will stop the execution of more code and case testing inside the block.
When a match is found, and the job is done, it's time for a break. There is no need for more testing.

**8. Java Loops**

Loops can execute a block of code as long as a specified condition is reached.
Loops are handy because they save time, reduce errors, and they make code more readable.
**while loop** loops through a block of code as long as a specified condition is true:

**Syntax**

```
while (condition) {
  // code block to be executed
}
```

**Do/While Loop**

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

**Syntax**

```
do {
  // code block to be executed
}
while (condition);
```

**For Loop**

When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:

**Syntax**

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

Statement 1 is executed (one time) before the execution of the code block.
Statement 2 defines the condition for executing the code block.
Statement 3 is executed (every time) after the code block has been executed.

**9. Implementation of First Java Program to print "Hello World"**

```
// Your First Program

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

**10. Lab Assignments to complete in this session**

1. Implement a java program to calculate gross salary and net salary taking the following data.
   Input: empno, empname, basic
   Process
   DA=70% of basic
   HRA=30% of basic
   CCA= Rs. 240/-
   PF=10% of basic
   PT=Rs.100/-

2. Write menu driven java program which will read a number and should implement following methods
   i. Factorial ()
   ii. testArmstrong ()
   iii. testPalindrome ()

3. Write a Java Program to take an integer N and print its first 10 multiples. Each multiple N * i (where 1<=i<=10) should be printed on a new line in the form: N x i = result.

4. Take input of age of three people by user and determine oldest and youngest among them.

5. If
   x = 2
   y = 5

z = 0
Then find values of the following expressions:
a. x == 2
b. x != 5
c. x != 5 && y >= 5
d. z != 0 || x == 2
e. !(y < 10)

6. A shop will give discount of 10% if the cost of purchased quantity is more than 1000.
Ask user for quantity
Suppose, one unit will cost 100.
Judge and print total cost for u