Welcome

Core Java

ph no: 9513216462 |email : info@emexotechnologies.com |website: https://www.emexotechnologies.com

What's Java Technology ?

- **Java technology is both a programming language and a platform**

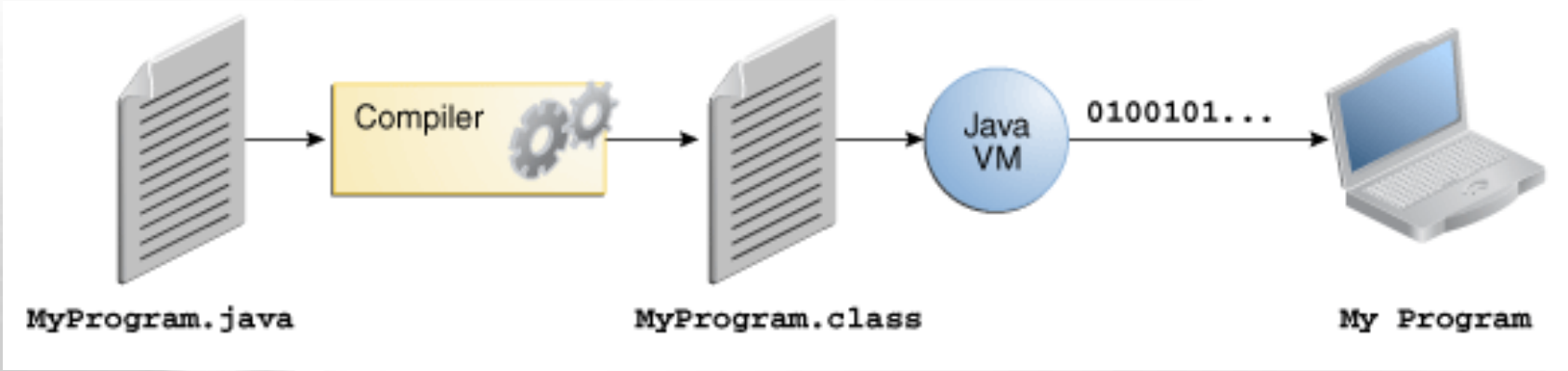
Java Programming Language

- Java is one of the most popular Programming Language
- Java is a High Level Programming Language
- Java is a platform-independent language
- Java can do everything you want to develop

What makes Java so popular

- ***Platform independent*:** A Java program can run on any platform including Windows, Linux, Mac, Solaris
- ***Powerful language*:** The Java programming language is strong typed, has clear syntax and addresses
- ***Secure*:** Java has a built-in strong security manager from ground-up that prevents malicious code and viruses.
- ***High performance*:** with the JVM has been improved over the years, Java applications are fast.
- ***Networking*:** with built-in support for networking, developers can write internet-based and networked applications easily.

Java Execution Flow



Java Execution Flow (Contd..)

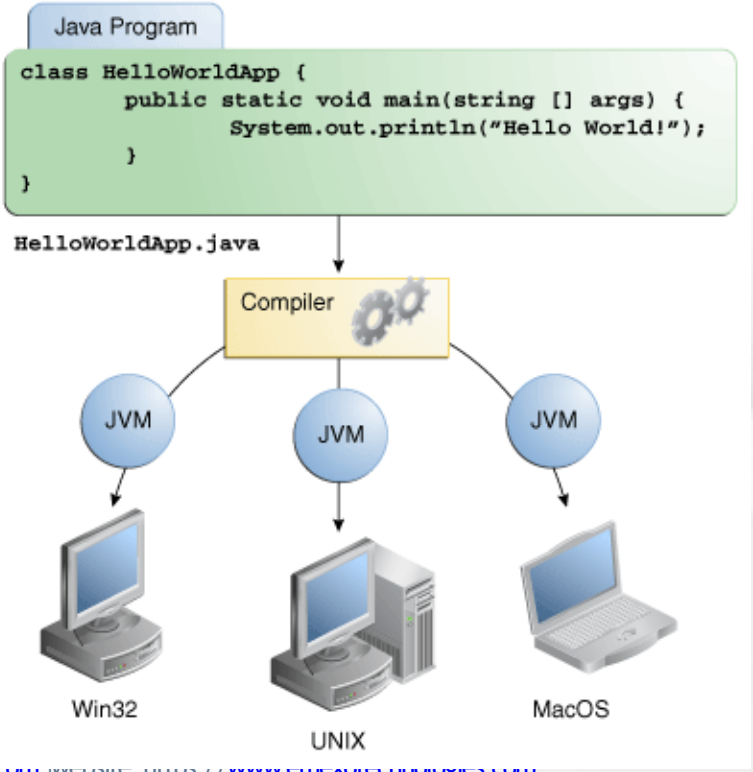
- All source codes are written in plain text files ending with .java extension
- Source files are then compiled into .class files by the java compiler, class files contains byte codes
- JVM translates byte code into machine code of the target operating system.

What is JVM ?

- JVM stands for Java Virtual Machine.
- Source files are then compiled into .class files by the java compiler, class files contains byte codes
- JVM translates byte code into machine code of the target operating system.

Java Virtual Machine (Contd..)

- **JVM is Platform independent**



What is JRE ?

- JRE stands for *Java Runtime Environment*.
- It provides the libraries, JVM and other components necessary for you to run applets and applications written in the Java programming language.
- The JRE contains standard tools such as java, keytool, policytool,… but it doesn't contain compilers or debuggers for developing applets and applications.
- When you deploy your Java applications on client's computer, the client needs a JRE to be installed.

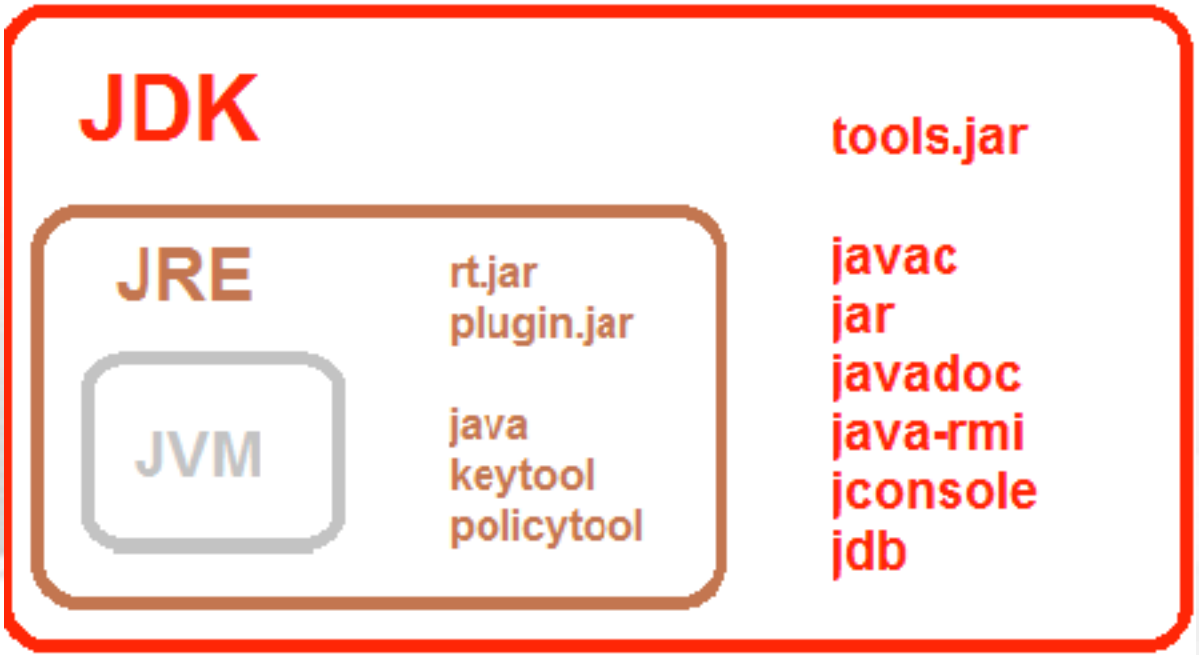
What is JDK ?

- JDK stands for *Java Development Kit*.
- It's a superset of JRE.
- The JDK includes the JRE plus command-line development tools such as compilers (javac) and debuggers (`jdb`) and others (jar, `javadoc`, etc) that are necessary or useful for developing applications.
- As a Java programmer, you have to install JDK as a minimum requirement for the development environment

JVM, JRE and JDK Summary

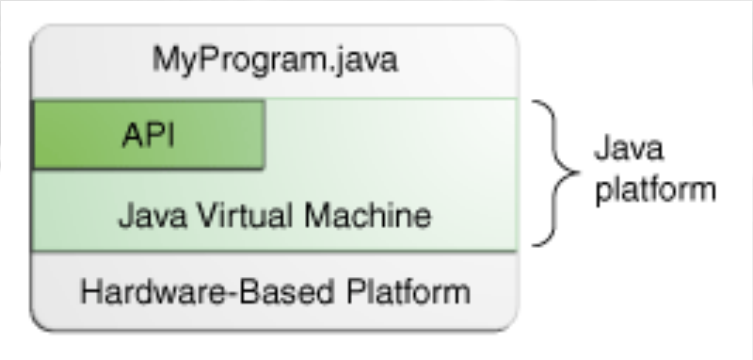
- **JVM:** is the virtual machine that runs Java applications. The JVM makes Java platform-independence.
- **JRE** = JVM + standard libraries: provides environment for executing Java applications.
- **JDK** = JRE + development tools for compiling and debugging Java applications.

JVM, JRE and JDK Summary (Contd..)

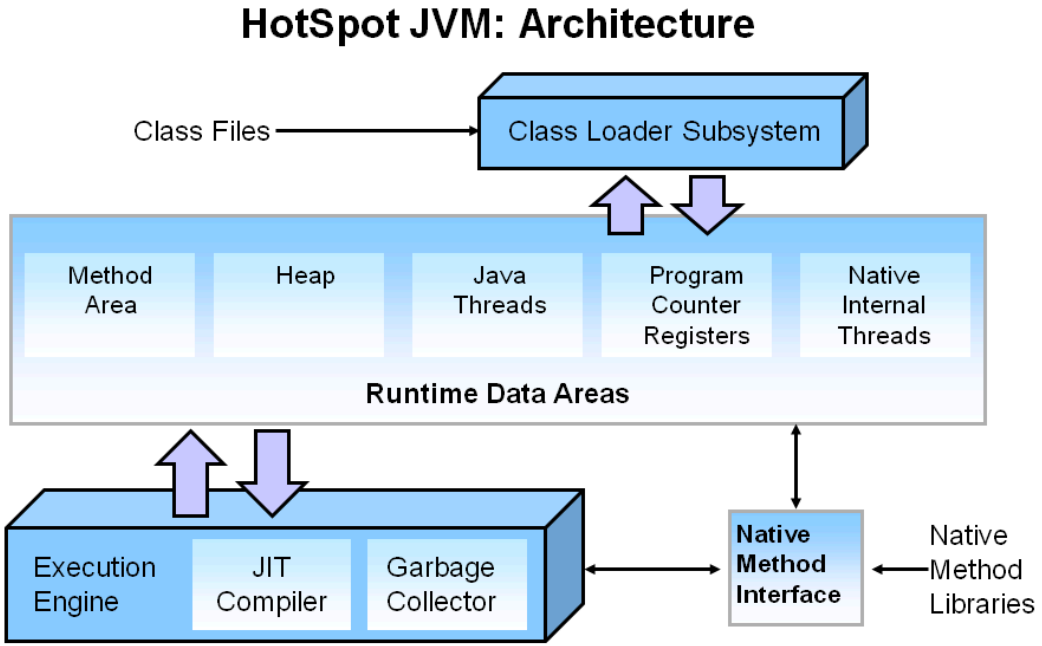


What's Platform ?

- A platform is a hardware or software environment in which a programs runs, Most popular platforms - Microsoft Windows, Linux, Solaris OS and Mac OS
- The java platform is differs from others, in that its software only platform that runs on the top of other hardware based platforms.



JVM Architecture



Java Project

- Java Project Structure
- Java File Structure

Access Modifier

- Access Modifier used to control accessibility to classes, interfaces, fields, constructors and methods.
- In other words, we can use access modifiers to protect data and behaviors from the outside world

Types of Access Modifier

- **public**
  - When applied to a class, the class is accessible from any classes regardless of packages
  - When applied to a member, the member is accessible from any classes

```
package animal;

public class Dog {
    public String name;

    public void bark() {
        System.out.print("Gow Gow!");
    }
}
```

Types of Access Modifier (Contd..)

- **private**
  - When a member is marked as private it is only accessible from within the enclosing class.
  - Nothing outside the class can access a private member.
  - It can be applied for members only. There is no ‘private’ class or interface

```
package animal;

public class Dog {

    private String breed;

    public Dog() {
        breed = "Bull";
    }
}
```

Types of Access Modifier (Contd..)

- **protected**
  - When a member of a class is marked as protected, it is accessible by only classes in the same package or by a subclass in different package.
  - It is applied for members only. There is no 'protected' class or interface.

```
package animal;

public class Dog {
    protected void waveTail() {
        System.out.print("Waving my tail...");
    }
}
```

Types of Access Modifier (Contd..)

- **default**
  - When a class or a member has default accessibility, it is accessible to only classes in the same package.
  - It doesn't have an associated keyword
  - When no explicit access modifier is specified, the types or members have default accessibility.

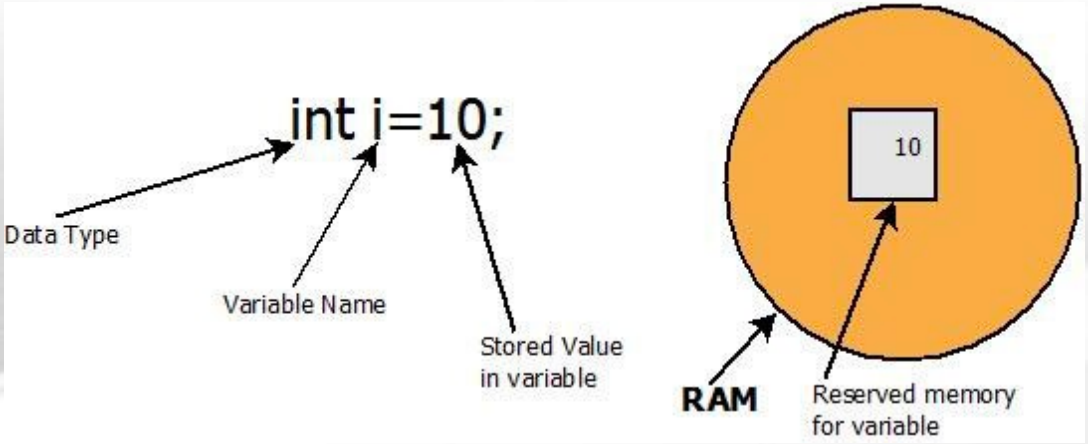
```
package animal;

class Wolf {
    public void play() {

    }
}
```

What is a Variable?

- In Java, a variable is a named reference to a memory area where the value of the variable is stored.



- **Static Variable**
  - Also know as *class variables*.
  - It is any field declared with the static modifier. It means that there is exactly one copy of this variable in existence, regardless of how many times the class has been instantiated..

```
package com.emexo.variable;

public class Bank {
    // static variable
    public static final String BANK_NAME = "JPMC";

    public static void main(String[] args) {
        // invoke static variable
        System.out.println(Bank.BANK_NAME);
    }
}
```

Types of Variable (Contd..)

- **Instance Variable**
  - Variables declared (in class) without **static** keyword.
  - Non-static fields are also known as instance variables because their values are unique to each instance of a class.

```
package com.emexo.variable;

public class Bank {
    // Instance variable
    public long accountNumber = 699771;
    public String accountName = "Gary";

    public static void main(String[] args) {
        // invoke instance variable
        Bank bank = new Bank();
        System.out.println(bank.accountNumber);
        System.out.println(bank.accountName);
    }
}
```

Types of Variable (Contd..)

- **Local Variable**
  - These are used inside methods as temporary variables exist during the method execution.
  - Local variables are only visible to the methods in which they are declared; they are not accessible from the rest of the class.

```
package com.emexo.variable;

public class Bank {

    public static void main(String[] args) {
        // Local variable
        long balance = 5000001;
    }
}
```

Constructor

- A constructor is a special code block of a class.
- It is always invoked when a new instance of the class is created.
- In other words, constructors are used to initialize state of an object when the object is being created.
- Constructors have same name as the class name.
- Constructors have a parameter list like methods but don't have a return type, nor even void.
- Types of Constructor
  - Default Constructor
  - Parameterized Constructor

Constructor (Contd..)

- **Default Constructor**
  - In case, programmer does not provide any constructor in class definition – JVM provides a default constructor to the class in runtime.
  - Programmer also can override default constructor in class. Let's look at the syntax.

Constructor (Contd..)

- **Parameterized Constructor**
  - A constructor that has parameters is known as parameterized constructor.
  - If we want to initialize fields of the class with our own values, then use a parameterized constructor.

Constructor (Contd..)

- **Constructor Rule**
  - Constructor name MUST be same as name of the class.
  - There cannot be any return type in constructor definition.
  - There cannot be any return statement in constructor.
  - Constructors can be overloaded by different arguments.
  - If you want to use **super()** i.e. super class constructor then it must be first statement inside constructor.

Methods

- Method is a collection of statements that perform some specific task and return the result to the caller.
- Methods in Java allow us to reuse the code without retyping the code. In Java, every method must be part of some class that is different from languages like C, C++, and Python.
- Types of Method
  - Static Method
  - Instance Method

Primitive Data Type

- In Java, the primitive data types are the predefined data types of Java. They specify the size and type of any standard values.
- Java has 8 primitive data types
  - byte - 8 bit
  - short - 16 bit
  - int - 32 bit
  - long - 64 bit
  - float - 32 bit
  - double - 64 bit
  - boolean - 1 bit
  - char - 16 bit

Wrapper Class

Primitive Data Type	Wrapper Class
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean

Autoboxing

- Autoboxing refers to the conversion of a primitive value into an object of the corresponding wrapper class is called autoboxing.
- For example, converting int to Integer class.

Unboxing

- Unboxing on the other hand refers to converting an object of a wrapper type to its corresponding primitive value.
- For example conversion of Integer to int.

Conditional Statement

- To execute the code based on the some condition
  - If Condition
  - If else Condition
  - If else If Condition
  - Switch Case

Looping Statement

- To execute the same set of code till the condition is false
  - For Loop
  - While Loop
  - Do While Loop

String

- String is a Immutable Class
- Cannot Change once creates
- There are 2 ways to create a new String
  - String Literal
  - Using new keyword