CORE JAVA

Java is a high-level programming language originally developed by Sun Microsystems and released in 1995. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. This Coaching gives a complete understanding of Java.

For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

- **Differences among C, C++ and Java Programming Languages : C vs C++ vs Java**
- The purpose of learning a programming language is to become a better programmer i.e. to become more effective at designing and implementing new systems and at maintaining old ones.

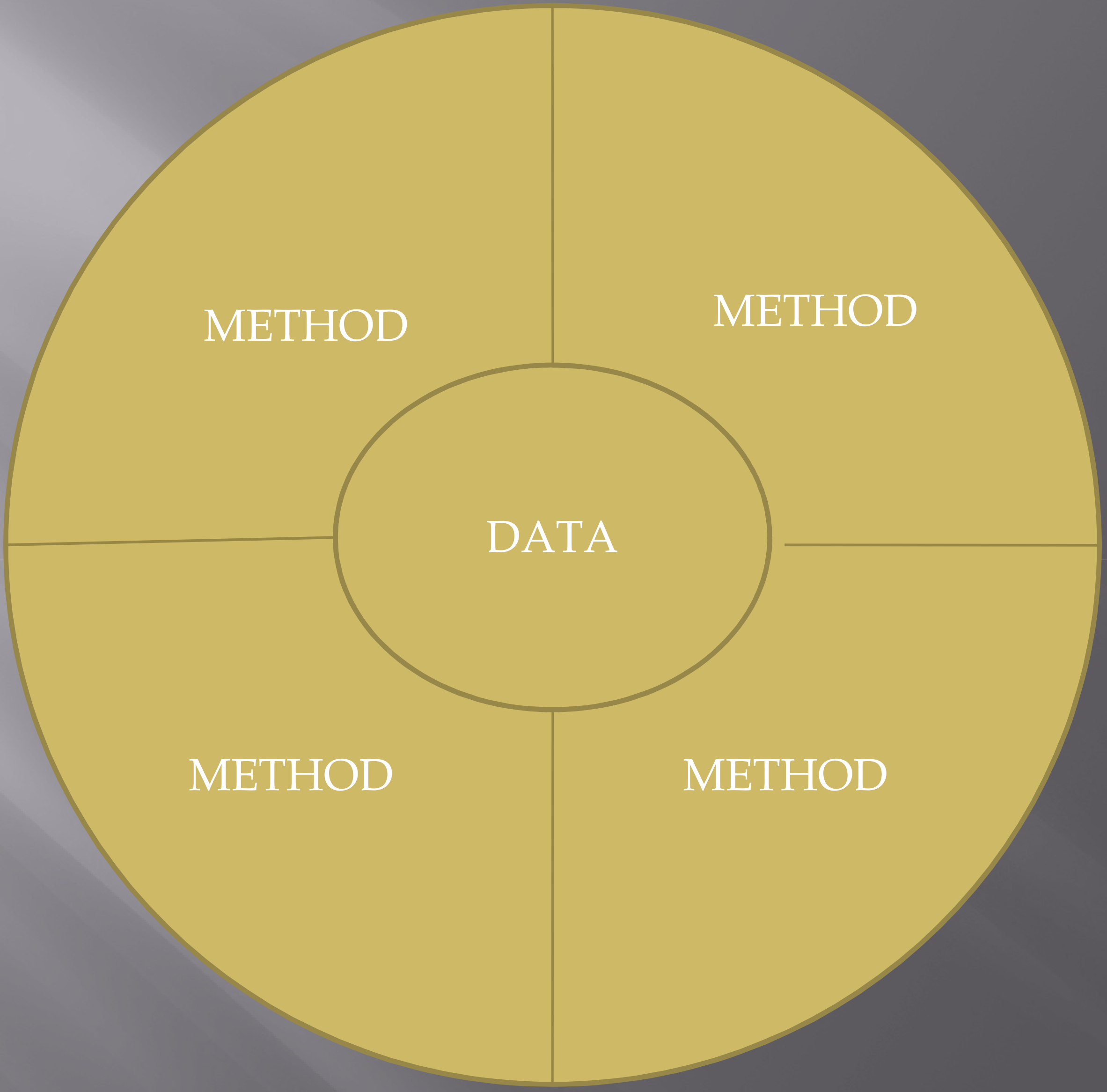
  C, C++ and Java are the most popular programming languages used today at a broad level. They have a pretty similar syntax for basic concepts. Most of the basic constructs like if statements, loops, function syntax, switch case statements and concepts like recursion are still valid. Many other concepts like the syntax for comments, and the idea of static class variables, also hold in both Java and C++.
  Java uses the syntax of C and structure of C++ language.

- **Introduction to C, C++ and Java :**

C is a general-purpose high level language that was originally developed by Dennis Ritchie in 1972 for the Unix operating system. C is a successor of B language which was introduced around 1970. C is a structured language which is easy to learn and produces efficient programs. it`s a top-down approach. It can handle low-level activities and can be compiled on a variety of computers. Today C is the most widely used System Programming Language.

C++ is a general purpose programming language developed by Bjarne Stroustrup starting in 1979 at Bell Labs,designed to make programming more enjoyable
for the serious programmers.C++ is a superset of the C programming language.
In addition to the facilities provided by C, C++ provides flexible and efficient facilities for defining new types.The key concept in C++ is class. A class is a user defined type.

Java is a programming language created by James Gosling from Sun Microsystems in 1991. The first publicly available version of Java (Java 1.0) was released in 1995. The Old name of Java was Oak. Java is now taken by oracle corporation.The acquisition of Sun Microsystems by Oracle Corporation was completed by Oracle in January 2010.The current version of Java is Java 1.7 (Java 7). Java is a Programming language as well as a Platform itself.


METHOD
METHOD
DATA
METHOD
METHOD

- Java is guaranteed to be **Write Once, Run Anywhere.**
- Java is −
- **Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform Independent** − 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.
- **Simple** − Java is designed to be easy to learn
- **Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

- **Architecture-neutral** − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on any platform.
- **Robust** − Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- **Multithreaded** − With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously.

- **Interpreted** − Java byte code is translated on the fly to native machine instructions and is not stored anywhere.
- **High Performance** − With the use of Just-In-Time compilers, Java enables high performance.
- **Distributed** − Java is designed for the distributed environment of the internet.
- **Dynamic** − Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment.

Tools You Will Need

- Pentium 200-MHz computer with a minimum of 64 MB of RAM (128 MB of RAM recommended).
- You will also need the following software –
- Linux 7.1 or Windows xp/7/8 operating system
- Java JDK 8
- Microsoft Notepad or any other text editor

◘ *First Java Program*

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

Local Environment Setup:

- Java SE is freely available from the link Download Java. You can download a version based on your operating system.
- Follow the instructions to download Java and run the **.exe** to install Java on your machine. Once you installed Java on your machine, you will need to set environment variables to point to correct installation directories −
- Setting Up the Path for Windows
- Assuming you have installed Java in *c:\Program Files\java\jdk* directory −
- Right-click on 'My Computer' and select 'Properties'.
- Click the 'Environment variables' button under the 'Advanced' tab.
- Now, alter the 'Path' variable so that it also contains the path to the Java executable. Example, if the path is currently set to 'C:\WINDOWS\SYSTEM32', then change your path to read 'C:\WINDOWS\SYSTEM32;c:\Program Files\java\jdk\bin'.

- Popular Java Editors
- To write your Java programs, you will need a text editor. There are even more sophisticated IDEs available in the market. But for now, you can consider one of the following −
- **Notepad** − On Windows machine, you can use any simple text editor like Notepad (Recommended for this tutorial), TextPad.
- **Netbeans** − A Java IDE that is open-source and free which can be downloaded from https://www.netbeans.org/index.html.
- **Eclipse** − A Java IDE developed by the eclipse open-source community and can be downloaded from https://www.eclipse.org/

- When we consider a Java program, it can be defined as a collection of objects that communicate via invoking each other's methods.
- **Object** − Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behavior such as wagging their tail, barking, eating. An object is an instance of a class.
- **Class** − A class can be defined as a template/blueprint that describes the behavior/state that the object of its type supports.
- **Methods** − A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
- **Instance Variables** − Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

- First Java Program

```
public class MyFirstJavaProgram {
/* This is my first java program. * This will print
   'Hello World' as the output */
 public static void main(String []args) {
   System.out.println("Hello World");
 // prints Hello World
}
}
```

- To save the file, compile, and run the program,Please follow the subsequent steps −
- Open notepad and add the code as above.
- Save the file as: MyFirstJavaProgram.java.
- Open a command prompt window and go to the directory where you saved the class. Assume it's C:\.
- Type 'javac MyFirstJavaProgram.java' and press enter to compile your code. If there are no errors in your code, the command prompt will take you to the next line (Assumption : The path variable is set).
- Now, type ' java MyFirstJavaProgram ' to run your program.
- You will be able to see ' Hello World ' printed on the window.

- Output
- C:\ > javac MyFirstJavaProgram.java
- C:\ > java MyFirstJavaProgram
  Hello World

- Basic Syntax
- About Java programs, it is very important to keep in mind the following points.
- **Case Sensitivity** − Java is case sensitive, which means identifier **Hello** and **hello** would have different meaning in Java.
- Class Names − For all class names the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.
- **Example:** *class MyFirstJavaClass*
- Method Names − All method names should start with a Lower Case letter. If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.
- **Example:** *public void myMethodName()*

- **Program File Name** − Name of the program file should exactly match the class name.
- When saving the file, you should save it using the class name (Remember Java is case sensitive) and append '.java' to the end of the name (if the file name and the class name do not match, your program will not compile).
- **Example:** Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as '*MyFirstJavaProgram.java‘*
- Package Name – Name of package should start with lowercase letter.
- Ex: ‘myJavaPrograms’
- **public static void main(String args[])** − Java program processing starts from the main() method which is a mandatory part of every Java program.

- Java Identifiers
- All Java components require names. Names used for classes, variables, and methods are called **identifiers**.
- All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_).
- After the first character, identifiers can have any combination of characters.
- A key word cannot be used as an identifier.
- Most importantly, identifiers are case sensitive.
- Examples of legal identifiers: age, $salary, _value, __1_value.
- Examples of illegal identifiers: 123abc, -salary.

- **Java Modifiers**
- Like other languages, it is possible to modify classes, methods, etc., by using modifiers. There are two categories of modifiers −
- **Access Modifiers** − default, public , protected, private
- **Non-access Modifiers** − final, abstract, strictfp
- We will be looking into more details about modifiers in the next section.
- **Java Variables**
- Following are the types of variables in Java −
- **Local Variables**
- **Class Variables (Static Variables)**
- **Instance Variables (Non-static Variables)**

- **Java Arrays**
- Arrays are objects that store multiple variables of the same type. However, an array itself is an object on the heap. We will look into how to declare, construct, and initialize in the upcoming sessions.

- **Java Enums**
- Enums were introduced in Java 5.0. Enums restrict a variable to have one of only a few predefined values. The values in this enumerated list are called enums.
- With the use of enums it is possible to reduce the number of bugs in your code.
- For example, if we consider an application for a fresh juice shop, it would be possible to restrict the glass size to small, medium, and large. This would make sure that it would not allow anyone to order any size other than small, medium, or large.
- Example
- class FreshJuice { enum FreshJuiceSize{ SMALL, MEDIUM, LARGE }
- FreshJuiceSize size; } public class FreshJuiceTest { public static void main(String args[]) {
- FreshJuice juice = new FreshJuice(); juice.size = FreshJuice.FreshJuiceSize.MEDIUM ;
- System.out.println("Size: " + juice.size); } }
- Output : Size: MEDIUM

- Java Keywords
- The following list shows the reserved words in Java. These reserved words may not be used as constant or variable or any other identifier names.
- Abstract
- assert
- Boolean
- Break
- Byte
- Case
- Catch
- Char
- Class
- Const
- Continue
- Default
- Do
- Double

- Else
- Enum
- Extends
- Finalfinally
- Float
- Forgotoif
- Implements
- Import
- Instanceof
- Intinterface
- Long
- Native
- New
- Package
- Private
- Protected
- Public
- Return
- Short

- Static
- Strictfp
- Super
- Switch
- Synchronized
- This
- Throw
- Throws
- Transient
- try
- Void
- Volatile
- while

- Comments in Java
- Java supports single-line and multi-line comments very similar to C and C++. All characters available inside any comment are ignored by Java compiler.
- Example
- public class MyFirstJavaProgram { /* This is my first java program. * This will print 'Hello World' as the output * This is an example of multi-line comments. */ public static void main(String []args) { // This is an example of single line comment /* This is also an example of single line comment. */ System.out.println("Hello World"); } }Output
- Hello World

- Inheritance
- In Java, classes can be derived from classes. Basically, if you need to create a new class and here is already a class that has some of the code you require, then it is possible to derive your new class from the already existing code.
- This concept allows you to reuse the fields and methods of the existing class without having to rewrite the code in a new class. In this scenario, the existing class is called the **superclass** and the derived class is called the **subclass**.
- Interfaces
- In Java language, an interface can be defined as a contract between objects on how to communicate with each other. Interfaces play a vital role when it comes to the concept of inheritance.
- An interface defines the methods, a deriving class (subclass) should use. But the implementation of the methods is totally up to the subclass.

- Java is an Object-Oriented Language. As a language that has the Object-Oriented feature, Java supports the following fundamental concepts −
- Polymorphism
- Inheritance
- Encapsulation
- Abstraction
- Classes
- Objects
- Instance
- Method
- Message Parsing

- *Thank You*