Java Introduction.

1. It is introduced in 1990. Initially it is known as Oak Language. Then it is renamed as Java language.
2. From Java 2nd version onwards, java language distributed into 3 editions.
   1. Java SE (J2SE)
      1. Java SE stands for Java Standard Edition.
      2. All the basics of java concepts included in this edition and it is also known as Core Java.
      3. Can Develop Console Based and Desktop Application.
   2. Java EE (J2EE)
      1. Java EE stands for Java Enterprise Edition.
      2. It is a combination of multiple java technologies. It is also known as Advance Java.
      3. It is majorly used for developing a Web Application.
   3. Java ME (J2ME)
      1. Java ME stands for Java Micro Edition.
      2. Using this you can develop Embedded and Mobile application.
3. Initially java language was managed by Sun Microsystem which acquire by Oracle on 20s.

**Rules:**

1. **In a source file you can create more than one classes. But only one class must be public and the file name must be same as public class name.**
2. **In all the classes created in a source file can have a separate main method.**

**Important Components of Java**

1. JVM
   1. JVM stands for Java Virtual Machine.
   2. It is use to execute the java program, and also support the all activities required for execution such as stating up of you code, memory allocation, communicating with OS.
   3. To Execute the java program JVM must be present inside you system.
2. JRE
   1. JRE stands for Java Runtime Environment.
   2. It is consisting of JVM and APIs. Which create the runtime environment for the java program.
   3. On users system JRE must be available.
3. APIs
   1. APIs stands for Application Programming Interface.
   2. It is a set of predefine functionalities which is provided by java language.
4. JDK
   1. JDK stands for Java Development Kit.
   2. It is a combination of JRE, JVM, APIs and some development tools.
   3. You can develop the java code, compile it and execute the java code.
   4. JDK has present in Developers system.

|  |  |
| --- | --- |
| **Overloading** | **Overriding** |
| It is performed within the class | It is performed in sub class.  It occurs in two classes that is IS - A relationship(Inheritance) |
| Parameters must be different | Parameters must be same |
| It is the example of compile time polymorphism | It is the example of runtime polymorphism |
| In the overloading return data type can be same or it can be change. | Return data type must be same if it is primitive or void or it can be sub type. |
| Can overload static methods(main method also) | Can not override static methods |
| Can overload final methods | Can not final static methods |
| **Advantage**  You can improve the code readability.  Can provide easy way to access the functionalities to the user. | **Advantage**  You can change the implementation provided inside super class inside sub class. |
| In the overloading access modifier can be same or it can be change. | In the overriding access modifier can be same and can be changed but must be less restrictive |

|  |  |
| --- | --- |
| **Compiletime Polymorphism** | **Runtime Polymorphism** |
| The method call is resolved at the time of compilation and same will be followed at the time of execution | The method call is resolved at the time of execution, so different methods can be called at the runtime |
| Inheritance is not involved | Inheritance is involved |
| Eg: Overloading | Eg: Overriding |

| **Points** | **Static method** | **Non-static method** |
| --- | --- | --- |
| **Definition** | A **static method** is a method that belongs to a class, but it does not belong to an instance of that class and this method can be called without the instance or object of that class. | Every method in java defaults to a non-static method without a **static** keyword preceding it. **non-static** methods can access any **static** method and **static** variable also, without using the object of the class. |
| **Accessing members and methods** | In the **static** method, the method can only access only static data members and static methods of another class or same class but cannot access non-static methods and variables. | In the **non-static** method, the method can access static data members and static methods as well as non-static members and methods of another class or same class. |
| **Binding process** | The static method uses compile-time or early binding. | The non-static method uses runtime or dynamic binding. |
| **Overriding** | The static method cannot be overridden because of early binding. | The non-static method can be overridden because of runtime binding. |
| **Memory allocation** | In the **static** method, less memory is used for execution because memory allocation happens only once because the static keyword fixed a particular memory for that method in ram. | In the **non-static** method, much memory is used for execution because here memory allocation happens when the method is invoked and the memory is allocated every time when the method is called. |

**Data Type and Variable**

**DataType:**

1. Data Types are required to specify the type of values used in a program.
2. Data types are used to create a variable.
3. There are 2 types of data type in java
   1. Primitive Data Type
      1. Primitive Data type has a fixed sized which is predefine in java.
      2. Primitive data type are in non-object format.
   2. Non-Primitive Data Type
      1. Non-primitive data type doesn’t have any fixed size, its size will be decided as per it uses in the code.
      2. Non primitive data types are always in the Object format.

To see all methos and classes available

<https://docs.oracle.com/javase/7/docs/api/overview-summary.html>





