Comparison between Java and C++

Difference between Java and C++ are listed below:

Java	C++
Java does not support operator overloading.	C++ supports operator overloading.
Java does not support multiple inheritances.	C++ supports multiple inheritances.
It does not use pointer.	It uses pointer.
De-allocation of memory is done automatically with the help of the Garbage collection technique.	Requires explicit de-allocation of the Memory by the user.

Primitive – Primitive data types, also known as standard data types, are the data types that are built into the Java language Compiler. The compiler contains extensive instructions on the operations supported by the data type. There are eight primitive data types in Java. They are as follows:

- Boolean
- Char
- Byte
- Short
- Int
- Long
- Float
- Double

Abstract or Derived Data types– Abstract data types are based on primitive data types and have more functionality than primitive data types.

Constructor: -

- Constructor is a method, which initializes automatically when an object is created. It has the same name as the class name.
- Constructors do not possess a return type.
- There are two types of constructors:

Default
Parameterized: Has the capacity to accept arguments

- If the user has not defined a constructor then Java defines a default constructor with no parameter and an empty body

garbage collection : - When an object is no longer referred to by any variables, Java automatically reclaims the memory used by that object. This is known as **garbage collection**.

finalize () : --We should always remember that the **finalize()** method is called just before an object is reclaimed by the garbage collector. So, we cannot be sure when this particular method will be invoked. Therefore, we should not depend on the **finalize()** method for deallocating memory occupied by non-Java objects.

Method overloading*: -*In Java it is possible to define two or more methods within the same class that share the same name, as long as their parameter declarations are different. When this is the case, the methods are said to be *overloaded,* and the process is referred to as method overloading.

- If two or methods have the same name but have different number, sequence or data type of parameters in its argument list then such methods are called overloaded.
- Constructors can also be overloaded. It means multiple constructors in the class definition but with different signatures.

Method Overriding: - In a class hierarchy, when a method in a subclass has the same name and type signature as a method in its superclass, then the method in the subclass is said to *override* the method in the superclass.

Super:

- The first calls the super class' constructor.
- The second is used to access a member of the super class that has been hidden by a member of a subclass.

Inheritance

- Inheritance is the mechanism by which child classes are inherit from parent class.

- To inherit a class,the definition of one class is incorporated into the other using the extends keyword.
- The private member of the base class cannot be accessed from the derived class.
- The keyword super is used to refer to the immediate super class.
- When a class hierarchy is created the constructors are called in the order of derivation.

Static member

- The static members of a class can be accessed before an object of that class is created. Such members can be accessed with the help of the class name.

- A static method can only access other static methods and variables and cannot use super
- A static variable is initialized only once when the class is first loaded.

Final keyword

- The *final* keyword is used to signify that the variables behave as constants.
- A final class cannot be subclassed,
- A final variable cannot be modified .
- A final method cannot be overridden.

Abstract

- They are created by using the *abstract* keyword.
- A class is declared as abstract if it contains one or more abstract methods.
- An abstract method has only method declaration and no method body.
- An abstract class cannot be instantiated.
- The abstract must be overridden by the derived class or otherwise the derived must also be declared as abstract.

Interfaces

- Resembles an abstract class
- All the methods of the interface are abstract.
- Provides a means to define the protocol of a class without devoting any attention to implementation details.
- A class can implement an interface by overriding all the methods that have been defined in the interface.
- An interface can be implemented by a class with the help of the *implements* keyword.
- A class can implement multiple interfaces.

Packages

- Containers for groups of related classes.
- Avoids naming conflicts
- Syntax for creating a package:

  package package_name;
- The classes that make up a package should be stored in a directory with the same name as that of the package.
- In order to utilize the classes inside a package from outside that package the following syntax is required:

  import package_name.*;