Interface

- An interface in Java is a specification of method prototypes. It has static constants and abstract methods.
- Inside interface every fields are by default ‘public static final’.
- Every method by default ‘public’ and ‘abstract’.
- From Java 8 version we can define ‘static’ and ‘default’ method inside interface.
- From Java 9 version we can also define private method.

Purpose of the interface

- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritance.
- It can be used to achieve loose coupling.

Facts About Interface

- A Java class can implement multiple Java Interfaces. It is necessary that the class must implement all the methods declared in the interfaces.
- The interface allows sending a message to an object without concerning which classes it belongs.
- An interface cannot be instantiated.
- An interface reference can point to objects of its implementing classes.
- An interface can extend from one or many interfaces. Class can extend only one class but implement any number of interfaces.

Facts About Interface

- An interface cannot implement another Interface. It has to extend another interface if needed.
- An interface which is declared inside another interface is referred as nested interface
- At the time of declaration, interface variable must be initialized. Otherwise, the compiler will throw an error.
- The class cannot implement two interfaces in java that have methods with same name but different return type.

Default Methods in Interface

- With the release of Java 8, methods with implementation (default methods) were introduced inside an interface. Before that, all the methods were abstract in Java.

- To declare default methods inside interfaces, we use the default keyword. For example :

```java
public interface Test {
  // default method declaration
  public default void defaultMethod(){
    // default method can hold default implementation
  }
}
```

Why Default Methods?

- Let's take a scenario to understand why default methods are introduced in Java.
- Suppose, we need to add a new method in an interface.
- We can add the method in our interface easily without implementation. However, that's not the end of the story. All our classes that implement that interface must provide an implementation for the method.
- If a large number of classes were implementing this interface, we need to track all these classes and make changes in them. This is not only tedious but error-prone as well.
- To resolve this, Java introduced default methods. Default methods are inherited like ordinary methods.

Marker Interface

- An interface that has no members (methods and variables) is known as marker interface or tagged interface or ability interface.

- In java whenever our class is implementing marker interface our class is getting some capabilities that are power of marker interface.
  Ex:- Serializable , Cloneable

- *Note: - user defined empty interfaces are not a marker interfaces only, predefined empty interfaces*
  are marker interfaces.

ENUMARATION (1.5 version)

- *Enumeration is used to declare group of named constants.*
- *You define an enum type by using the enum keyword.*
- *Every enum constant represents object of type enum.*
- *The enum constants are by default 'public static final'.*
- Compiler will generate .class file for enum.

ENUMARATION (1.5 version)

- *Enum constructor is by default private hence it is not possible to create the object in outside of the enum.*
- *Every enum constant contains index value & it starts from 0.*
- *Every enum is final by default hence other classes are unable to extends.*
- *Values() method used to retrieve the all the constants at a time.*
- *Ordinal() method is used to print the index numbers of constants & index starts from 0.*

ENUMARATION (1.5 version)

- *Inside the enum it is possible to declare the constructor in this case the group of constants must be first line must ends with semicolon.*
- *Every constant is object three times the constructor will be executed & it is a static hence it will be executed during .class file loading.*
- *Inside the enum it is possible to declare parameterized constructors.*
- *Constructor is initializing specific values enum constants.*
- *Inside the enum it is possible to declare the main method but to execute this main method just execute the .class file.*

ENUMARATION (1.5 version)

- *If we declare the enum outside of the class the applicable modifiers are public,<default>, strictfp.*
- *If we are declaring enum inside the class the applicable modifiers are : public , <default> , strictfp , private , protected, static*
- *It is not possible to declare the enum inside the methods it means locally.*
- *Every enum is final by default hence it is not possible to extends other classes.*
- *Every enum is a final by default hence it is not possible to create child enums.*
- *Every enum internally extends java.lang.Enum hence it is not possible to extends other classes.*