**Interface** : looks like a class but it is not a class. An interface can have **methods and variables** just like a class, but the methods declared in interface are by default **abstract** (only method signature, no body i.e. without code). Also, the variables declared in an interface are **public, static & final** by default. So, in general words, Interface is collection of declarations (only constants and abstract methods)

It is compulsory for a non abstract class to implement **all** methods of an interface that it is implementing.

If a class partially implement interface needs to be declared as **abstract**. Example in eclipse.

* You cannot instantiate an interface.
* An interface **does not** contain any **constructors** because interface is not a class.
* you **cannot** **instantiate** an interface
* **All** of the methods in an interface are abstract.
* An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
* From **Java 8** you can define **static** methods in interfaces in addition to default methods.
* An interface is not extended by a class; it is **implemented** by a class.
* An interface can extend **multiple** interfaces. Also, a class can implement multiple interfaces. (Java does not support multiple inheritance, but we can achieve it with multiple interfaces.)
* the **variables** declared in an interface are **public, static & final** by default
* A class can implement **more than one** interface **at a time**. (in this way we can achieve multiple inheritance in java)
* A class **CAN NOT** extend interface.
* A class can extend **only one** class, but implement **many** interfaces.
* An interface can extend another interface, in a similar way as a class can extend another class.
* Class can extend another class and implement more than one interface at a time

**Static method in interface** : Since it is static, we call it as : **interfacename.methodname**

