Note:
Every method present inside interface is abstract but in abstract class also we can take only abstract methods then what is the need of interface concept ?
We can replace interface concepts with abstract class. But it is not a good programming practice. We are misusing the role of abstract class. It may create performance problems also.

JAVA8 Features over Interfaces:

1. Default Methods in Interfaces
2. Static Methods in Interfaces
3. Functional Interfaces

1. Default Methods in Interfaces:

- In general, if we declare abstract methods in an interface then we have to implement all that interface methods in more no.of classes with variable implementation part.
- In the above context, if we require any method implementation common to every implementation class with fixed implementation then we have to implement that method in the interface as default method.
- To declare default methods in interfaces we have to use the "default" keyword in method syntax like access modifier.

```
interface Interf{
       default void m1(){
              System.out.println("m1-A");
       }
}
class A implements Interf{

}
class Test{
      public static void main(String args[]){
             Interf i=new A();
             i.m1();
      }
}
```

NOTE:
It is possible to provide more than one default method within a single interface.

Example:

```
interface Interf{
        default void m1(){
                        ---------
        }
        default void m2(){
                        --------
        }
}
```

In JAVA8,it is possible to override default methods in the implementation classes.

```
interface Interf{
        default void m1(){
               System.out.println("m1-A");
        }
}
class A implements Interf{
        public void m1(){
              System.out.println("m1-A");
        }
}
class Test{
        public static void main(String args[]){
              Interf i=new A();
                    i.m1();
        }
}
```

2. Static Methods in Interfaces:

- Upto JAVA7 version, static methods are not possible in interfaces but from JAVA8 version static methods are possible in interfaces in order to improve sharability.
- If we declare static methods in the interfaces then it is not required to declare any implementation class to access that static method,we can use directly interface name to access static method.

NOTE:
If we declare static methods in an interface then they will not be available to the respective implementation classes,we have to access static methods by using only interface names not even by using interface reference variable

Example:

```
interface I{
       static void m1(){
             System.out.println("m1-1");
       }
}
class Test{
       public static void main(String args[]){
             I.m1();
       }
}
```

Note:
In JAVA8 version, interfaces will allow concrete methods along with either "static" keyword or "default" keyword.

3. Functional Interface:

- If any Java interface allows only one abstract method then it is called a "Functional Interface".
- To make any interface as Functional Interface then we have to use the following annotation just above of the interface. @FunctionalInterface

EX:
java.lang.Runnable
java.lang.Comparable

NOTE:
In Functional Interfaces we have to provide only one abstract method but we can provide any no.of default methods and any no.of static methods.

```
@Functional Interface
interface Interf{
	void m1();

	default void m3(){
		System.out.println("m3-I");
	}
	static void m4(){
		System.out.println("m4-I");
	}
}
class A implements Interf{
	public void m1(){
		System.out.println("m1-A");
	}
}
public class Test{
	public static void main(String args[]){
		Interf i=new A();
		i.m1();
		i.m3();
		Interf.m4();
	}
}
```

Output:
m1-A
m3-I
m4-I