Interface -

*) Abstraction in java is acheived by two ways
   i) — Using Abstract class
   ii) — By using interface.

*) Interface are similar to Abstract class but having all the methods of abstract type. But in Abstract class all methods are either Abstract methods or concrete methods or both.

*) In case of Interface all methods are Abstract methods. so it is (100%) Abstraction.

*) It is just like a blue print of class which indicates what the class will do.

*) It is used to acheive Abstraction.

*) It support multiple inheritance.

*) It can be used to achieve loose coupling. i.e. change in one will not effect other.

Syntax:-

```
interface <interface name>
{
   // methods  --> must be Abstract
   // fields   --> All methods are public by default
}                  --> public static final by default compiler will add
```

e.g. interface i1
{
public abstract void show();
↑ by default compiler will add:

public static final int a=10;
}

*). In latest version i.e. version 8th default concrete method is added.

i.e.

interface i1
{ default void Creat()
{ // also possible.
}
}

*) Also we can creat static method.

e.g. public static void run()
{
}

must be public by default compiler will add.

*)

*) To inherits a interface "implements" keyword is used.

e.g.

```
interface I1
{ void show();
}
class test implements I1
{ void show()
  {
  }
P.S.V.M ( )
{ test ob = new test();
  ob.show();
}
}
```

All the body of interface must be declared in child class.

object is not created in interface. only object of child class is created.

Interface

*) Abstraction in java is acheived by two ways
   i) — Using Abstract class
   ii) — By using interface.

*) Interface are similar to Abstract class but having all the methods of abstract type. But in Abstract class all methods are either Abstract methods or concrete methods or both.

*) In case of Interface all methods are Abstract methods. so it is (100%) Abstraction.

*) It is just like a blue print of class which indicates what the class will do.

*) It is used to acheive Abstraction.

*) It supports multiple inheritance.

*) It can be used to acheive loose coupling. i.e. change in one will not effect other.

Syntax:-

```
interface <interface name>
{
    // methods  → must be Abstract
    // fields   → All methods are public by default
}                → public static final
                   by default compiler will add
```

e.g. interface i1
{
public abstract void show();
↑ by default compiler will add:

public static final int a=10;
}

*). In latest version i.e. version 8th default concrete method is added.

i.e.
interface i1
{ default void Creat()
{ // also possible.
}
}

*) Also we can create static method.

e.g. Public static void run()
{
}

must be public by default compiler will add.

*)

*) To inherits a interface "implements" keyword is used.

e.g.
```
interface I1
{ void show();
}
class test implements I1
{ void show()
  {
  }
p.s.v.m ( )
{ test ob = new test();
  ob.show();
}
}
```

All the body of interface must be declared in child class.

object of interface created in interface. only object of child class is created.

Multiple Inheritance using interface.

```
interface I1
{ public void show();
}
interface I2
{ public void show();
}
class multi implements I1, I2
{ public void show()
  { s.o.p("Hello");
  }
p.s.v.m ( )
{ multi ob = new multi();
  ob.show();
}
```

interface extends Interface

```
interface A
{
    void funA();
}
interface B extends A
{
    void funB();
}
class C implements B
{
    public void funA()
    {
        S.O.P("A interface");
    }
    public void B()
    {
        S.O.P("B interface");
    }
}
public class IntExtend
{
    P S.V.M()
    {
        IntExtend obj = new IntExtend();
        obj.funA();
        obj.funB();
    }
}
```