

## Interface

It is a component in java which is used to achieve 100 % abstraction and multiple inheritance.

Syntax to create an interface

**[Access Modifier] interface InterfaceName**

{

**// declare members**

}

When an interface is compiled we get a class file with an extension .class only

ASUS Vivobook

**Case 1:**

```
interface Demo1
{
    public static void main(String[] args)
    {
        System.out.println("Hello World...!!!");
    }
}
```

**Case 2:**

```
interface Demo1
{
    int a; //CTE : Variable a is by default public, static, final. A final variable must be
           initialized.
}
```

ASUS Vivobook

### **EXAMPLE :**

```
interface Demo
{
    void m1();
    void m2();
    public void m3();
    static final int y = 7;
}
```

**The Demo is an interface .**

**In the interface all the members by default public in nature.**  
**And abstract methods are by default public and abstract.**

ASUS Vivobook

## What all are the members the can be declared in an interface?

MEMBERS	CLASS	INTERFACE
Static variables	Yes	Yes, but only final static variables
Non-static variables	Yes	No
Static methods	Yes	Yes, From JDK 1.8 v. <b>NOTE:</b> They are by default public in nature
Non-static methods	Yes	Yes but we can have only abstract non-static methods <b>NOTE :</b> Non-static methods are by default <ul style="list-style-type: none"><li>public</li><li>abstract</li></ul>
Constructors	Yes	No
Initializers (Static & non-static block)	Yes	No

### NOTE:

In interface, all the members are by default have public access modifier.

### **Why do we need an interface ?**

- To achieve 100% abstraction.Concrete non-static methods are not allowed.
- To achieve multiple inheritances.

**What all the members are not inherited from an interface?**

Only static members of an interface are not inherited to both class and Interface.

ASUS Vivobook

```
interface Demo2
{
    public void test() // CTE
    {
    }
}
```

Note : inside the interface non static concrete method is not allowed.

ASUS Vivobook

## **INHERITANCE WITH RESPECT TO INTERFACE:**

An Interface can inherit any number of interfaces with the help of an extends keyword.

**EXAMPLE: Single level inheritance**

```
interface I1
{
}
interface I2 extends I1
{
}
```

**NOTE :**

The interface which is inheriting an interface should not give implementation to the abstract methods. It should be given by any of the child class.

Note : we can't create the object for the interface , but we can create the reference variable of the interface and store the child type object reference.

ASUS Vivobook

## Extends and Implements keyword

### **Extends:**

We can achieve inheritance in between class and class ,or interface and interface by extends keyword.

### **Implements :**

To achieve inheritance In between interface and class we are using implements keywords.

ASUS Vivobook

## Single level Inheritance

I1

```
Void t1( );
Void t2( );
static void t3( )
{ }
```



I2 extends I1

```
Void t4( );
```

## **EXAMPLE 1 :**

- Interface I1 have 3 methods
  - 2 - non static abstract (**t1()** , **t2()**)
  - 1 - static ( **t3()** )
- Interface I2 have 3 methods
  - 2 - inherited non static abstract methods (**t1()** , **t2()**)
  - 1 - declared non static abstract methods ( **t4()** )

## Multi level inheritance

I1

```
Void t1( );
Void t2( );
static void t3( )
{ }
```

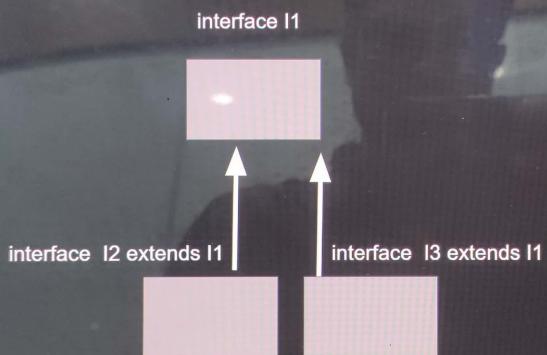
I2 extends I1

```
Void t4( );
```

I3 extends I2

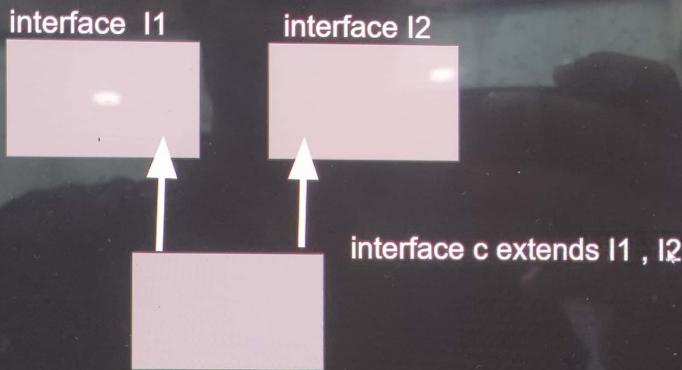
ASUS Vivobook

## Hierarchical inheritance

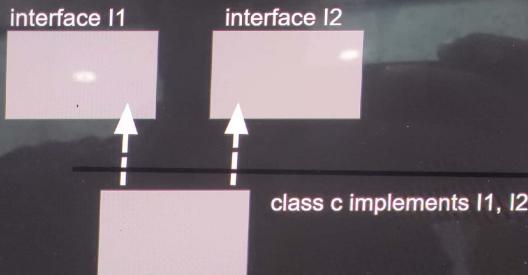


ASUS Vivobook

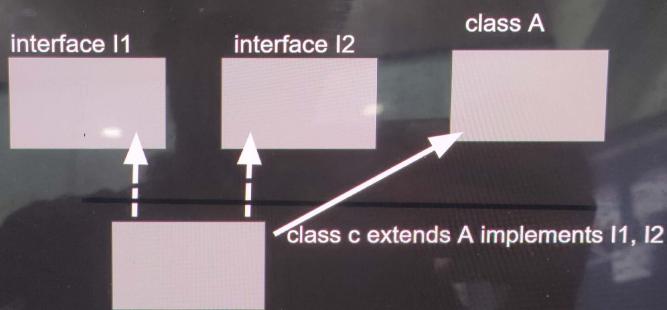
## Multiple inheritance with respect to interface



## Multiple inheritance with respect to class



## Multiple inheritance with respect to both class and interface



**NOTE:**

**With respect to interface there is no diamond problem.**

The reason,

- They don't have constructors.
- Non-static methods are abstract (do not have implementation)
- Static methods are not inherited.

**INHERITANCE OF AN INTERFACE BY THE CLASS :**

- Class can inherit an interface with the help of implements keyword.( class can be child of interface)
- Class can inherit more than one interface.(One class can have more than one interface as parent )
- Class can inherit a class and an interface at a time.

Note : A class can't be a Parent of interface.

ASUS Vivobook

**Class can't be the parent of interface.**



ASUS Vivobook

**Class can't be the parent of the interface because in every class by defaultly 11 non static concrete methods of object class are there. If we make class will be parent , then all methods will be inherited to the child but inside the interface concrete non static methods are not allowed.**

**NOTE:**

- If a class inherits an interface then it should give implementation to the abstract non-static methods of an interface.
- If the class is not ready to give implementation to the abstract methods of an interface then it is mandatory to make that class an abstract class.
- The next level of child class is responsible for giving implementation to the rest of the abstract methods of an interface.

ASUS Vivobook

## Type of Interface

There are 3 types of interface.

1. Regular Interface
2. Functional Interface
3. Marker Interface

## Regular Interface

Regular interface is a interface which contains more than one abstract method.

Ex :

```
public interface Imouse{  
    public void click();  
    public void rightClick();  
    public void doubleClick();  
}
```

ASUS Vivobook

## Functional Interface

Functional interface is an interface which contains only one abstract method.

Ex : Comparable, Comparator , Runnable etc.

In Comparable compareTo() method is there

In Comparator compare() method is there

In Runnable run() method is there

ASUS Vivobook

## Marker Interface

Marker interface is an interface which does not have any method in it.

It mainly used to indicate jvm about the certain activity.

We have some marker interface as follows :

1. Clonable
2. Serializable
3. RandomAccess etc.

**Note : After JDK 1.8 in interface static concrete and default concrete method are allowed .**

ASUS Vivobook

The Exception is an unexpected problem which occurs during the execution of the program ( i. E nothing but RunTime). When an exception occurs , the program execution stops unexpectedly( this is known abrupt Stop of the program).

Note:

1. Every Exceptions in java is one one of the child of Throwable class , that's why we can say that every exceptions in java 'Throwable Type'.
2. Every Exceptions occur because of one Statement.
3. A statement will throw an exception during unexpected problem ( i.e nothing but abnormal situation).
4. All the exception in java, is one one Class .(can be predefined or user defined).

## **What happens if exception Occurs**

The program will stop suddenly and rest part of the code after exception creation statements are not going to execute.

## Example

```
class Demo
{
    public static void main(String [] args)
    {
        int a = 6;
        int b = 0;
        System.out.println(5+7);
        System.out.println(a/b); //bcuz of one statement //Every Exception is java is one class and all are
        Throwable Type
    }
}
```

# Lists of Important Exceptions and Statements

## Statement:

1. a/b (if b = 0)
2. Obj\_reference\_var.object\_member
3. (ClassName) Obj\_reference\_var
4. Array\_ref\_var[index]
5. string\_ref\_var.charAt(index)

etc.

## Exception:

- ArithmaticException
- NullPointerException
- ClassCastException
- ArrayIndexOutOfBoundsException
- StringIndexOutOfBoundsException

## Exception Hierarchy





Scanned with CamScanner