ABSTRACTION

- It is a design process of hiding the implementation and showing only the functionality (only declaration) to the user is known as abstraction.

HOW TO ACHIEVE ABSTRACTION IN JAVA?

- In java, we can achieve abstraction with the help of abstract classes and interfaces.
- We can provide implementation to the abstract component with the help of inheritance and method overriding.

ABSTRACT MODIFIER :

- The abstract is a modifier, it is a keyword.
- It is applicable for methods and classes.

ABSTRACT METHOD

- A method that is prefixed with an abstract modifier is known as the abstract method.
- This is also said to be an incomplete method.
- The abstract method doesn't have a body (it has the only declaration)

Syntax to create abstract method :

abstract [access modifier] returnType methodName([For_Arg]) ;

NOTE :

Only child class of that class is responsible for giving implementation to the abstract method.

ABSTRACT CLASS :

ABSTRACT CLASS :

- If the class is prefixed with an abstract modifier then it is known as abstract class.
- We can’t create the object (INSTANCE) for an abstract class.

NOTE :

- **We can’t instantiate an abstract class**
- **An abstract class can have both abstract and concrete methods.**
- **If a class has at least one abstract method either declared or inherited but not overridden it is mandatory to make that class an abstract class.**

EXAMPLE :

```
abstract class Atm
{
abstract public double withdrawal();
abstract public void getBalance();
abstract public void deposit();
}
```

// hiding implementation by providing only functionality

Atm a = new Atm() // CTE

NOTE :

Only subclass of Atm is responsible for giving implementation to the methods declared in an Atm class.

Implementation of abstract method :

- If a class extend abstract class then it should give implementation to all the abstract method of the superclass.
- If inheriting class doesn't like to give implementation to the abstract method of superclass then it is mandatory to make subclass as an abstract class.
- If a subclass is also becoming an abstract class then the next level child class is responsible to give implementation to the abstract methods.

STEPS TO IMPLEMENT ABSTRACT METHOD:

STEP 1:

- Create a class.

STEP 2:

- Inherit the abstract class/ component.

STEP 3:

- Override the abstract method inherited (Provide implementation to the inherited abstract method).

CONCRETE CLASS:

- The class which is not prefixed with an abstract modifier and doesn't have any abstract method, either declared or inherited is known as a concrete class.

NOTE:

- **In java, we can create objects only for the concrete class.**

CONCRETE METHOD:

- The method which gives implementation to the abstract method is known as the concrete method.

EXAMPLE :

```
abstract class WhatsApp
{
abstract public void send();
}
class Application extends WhatsApp
{
public void send()
{
System.out.println("Send() method is implemented");
}
}
```

Creating object and calling the abstract method :

```
WhatsApp w = new Application();
w.send() // Send() method is implementeda
```

INTERFACES

INTERFACES :

- It is a component in java which is used to achieve 100% abstraction with multiple implementation.

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

EXAMPLE :

```
interface Demo
{
}
```

The Demo is an interface

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. NOTE: They are by default public in nature
Non-static methods	Yes	Yes but we can have only abstract non-static methods NOTE : Non-static methods are by default • public • abstract
Constructors	Yes	No
Initializers (Static & non-static block)	Yes	No

NOTE:

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

```
interface Demo1
{
int a; //CTE : Variable a is by default public, static, final. A final variable must be initialized.
}
interface Demo2
{
public void test() // CTE
{
}
}
```

Why do we need an interface ?

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 methods of an interface are not inherited to both class and Interface.

```
interface Demo3
{
public static void main(String[ ] a)
{
System.out.println(“Hello World..!!!”);
}
}
interface Demo1
{
Int a; //CTE : Variable a is by default public, static, final. A final variable must
be initialized.
}
interface Demo2
{
public void test() // CTE
{
}
}
interface Demo3
{
public static void main(String[] a)
{
System.out.println(“Hello World..!!!”);
}
```

INHERITANCE WITH RESPECT TO INTERFACE:

INHERITANCE WITH RESPECT TO INTERFACE:

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

EXAMPLE:

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

NOTE :

The interface which is inheriting an interface should not give implementation to the abstract methods.

EXAMPLE 1 :

Interface I1 have 3 methods

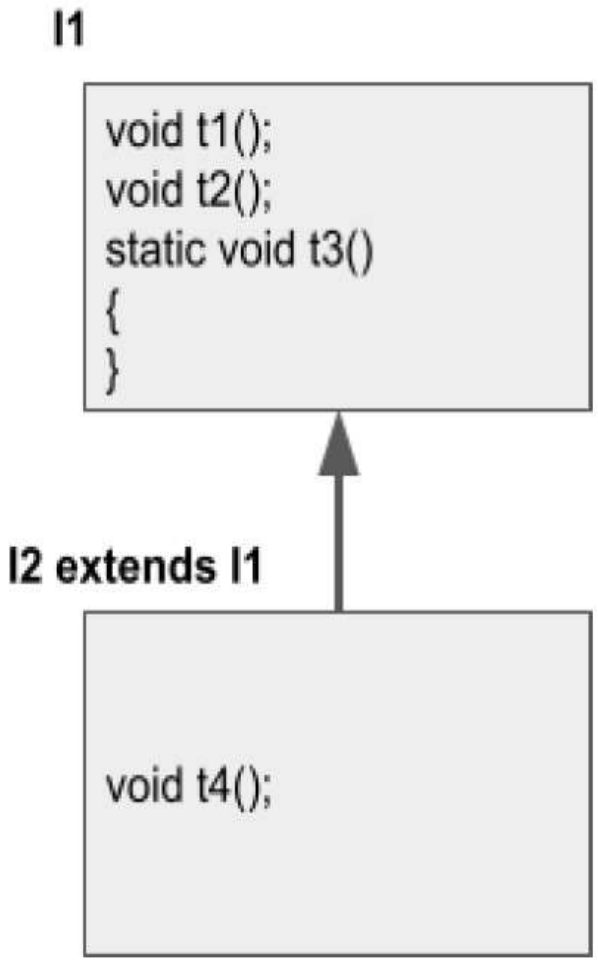
2 - non static (**t1() , t2()**)

1 - static (**t3()**)

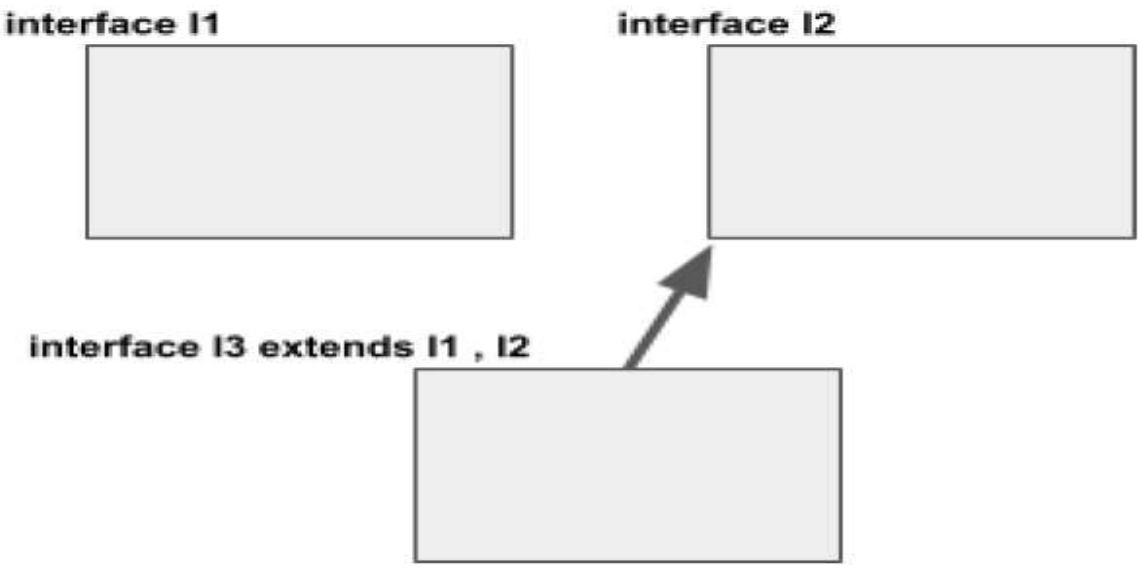
Interface I2 have 3 methods

2 - inherited non static method (**t1() , t2()**)

1 - declared non static methods (**t4()**)



EXAMPLE 2: Interface can inherit multiple interfaces at a time



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 :

INHERITANCE OF AN INTERFACE BY THE CLASS :

- Class can inherit an interface with the help of implements keywords.
- Class can inherit more than one interface.
- Class can inherit a class and an interface at a time.

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.**

EXAMPLE 1 : Class inheriting an interface

- Interface I1 have 3 methods

2 - non static (**t1() , t2()**)

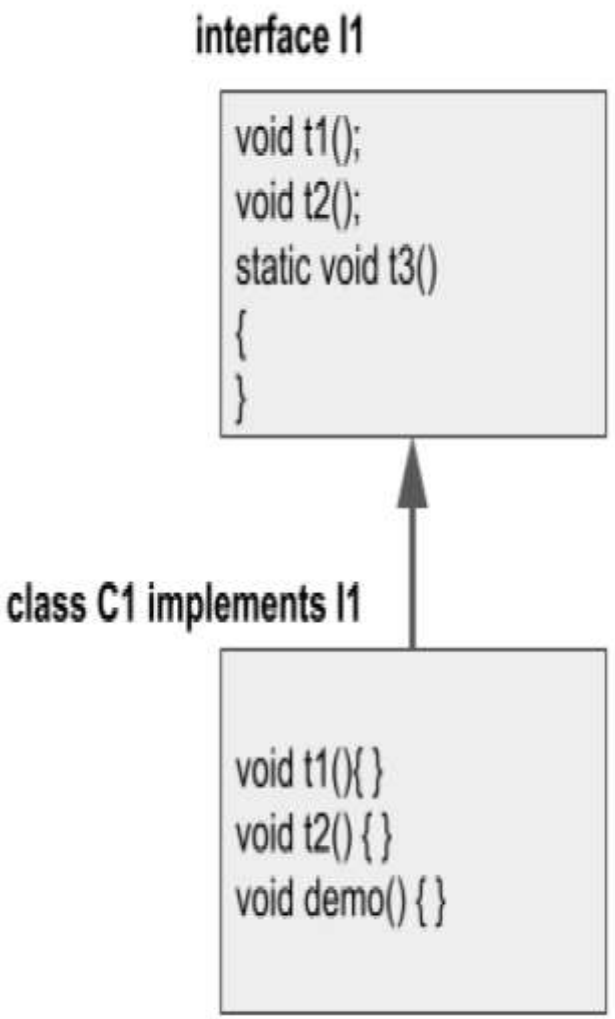
1 - static (**t3()**)

- Class c1 have 3 methods

2 - inherited and implemented

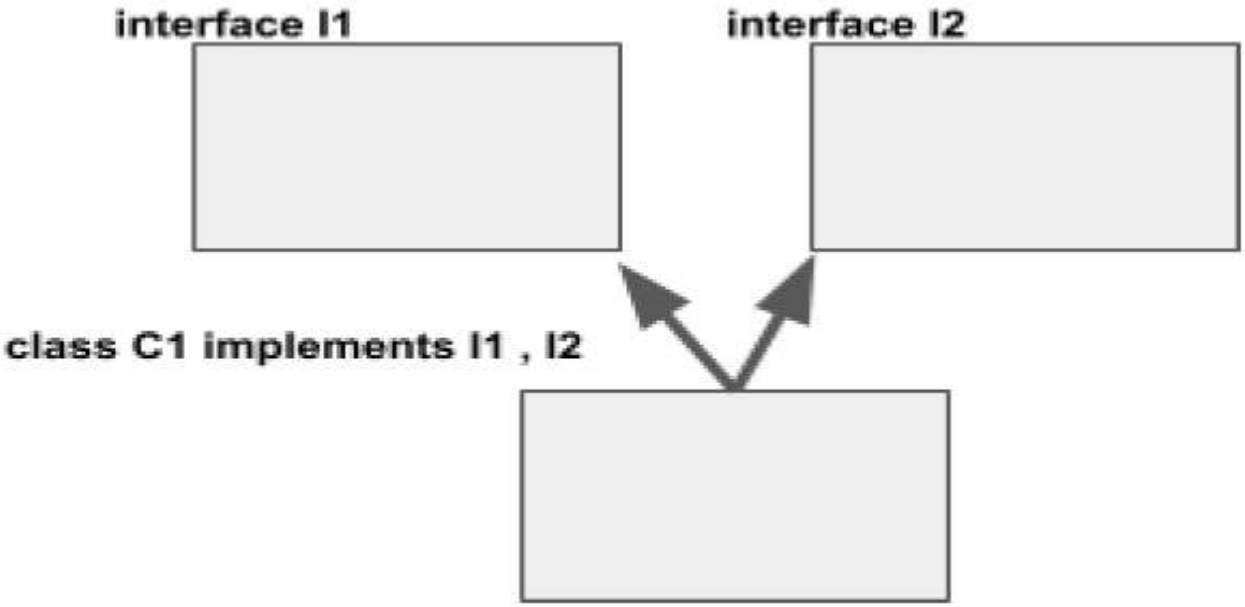
non static method (**t1() , t2()**)

1 - declared non static methods (**demo()**)



EXAMPLE 2: Class inheriting multiple interfaces

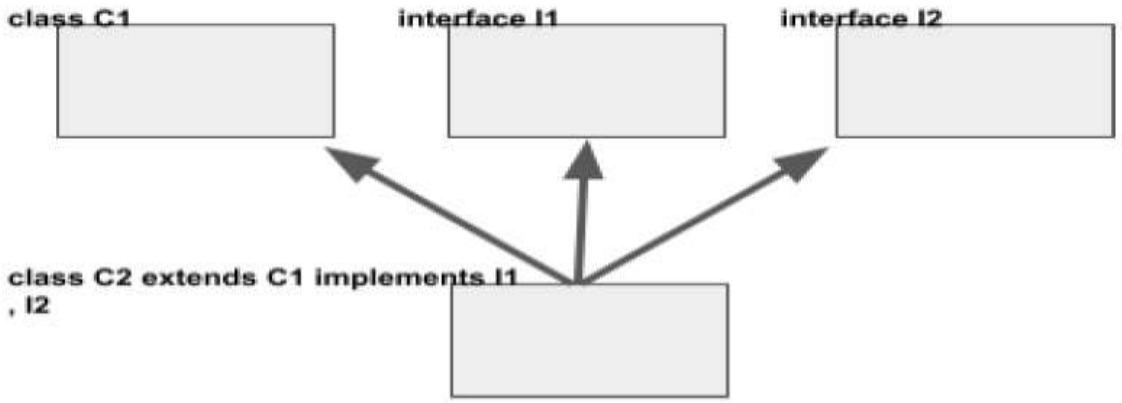
- The diamond problem is solved by implementing multiple interfaces by the class at a time.

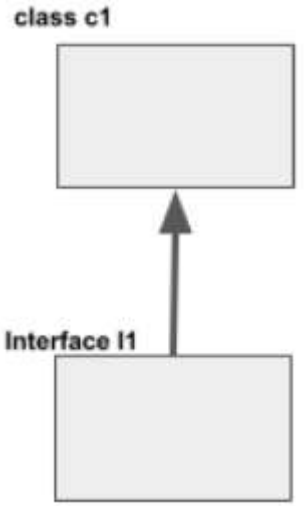


Reason,

- Non-static methods are not implemented in an interface.
- Static methods are not inherited.

EXAMPLE 3: Class can inherit interface and class at a time





RULE:

Use the extends first and then implements.

NOTE:

- **Class can inherit multiple interfaces but it can't inherit multiple classes at a time.**
- **Interface can't inherit a class. Interface can't inherit from the class**

Reason,

Class has concrete non-static methods, So class can't be a parent to the interface