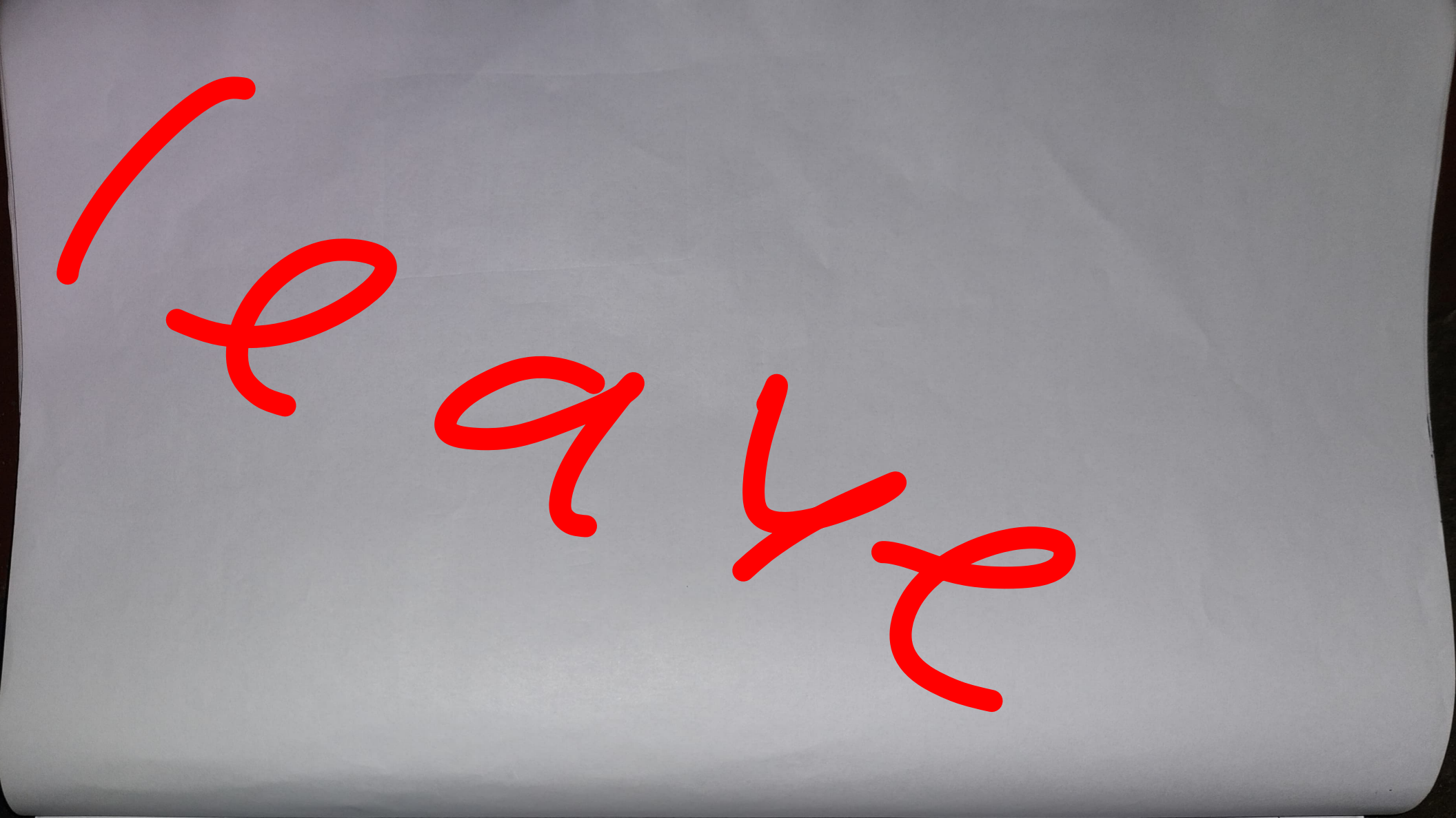
leave

leave


laye


leave

5/8/24

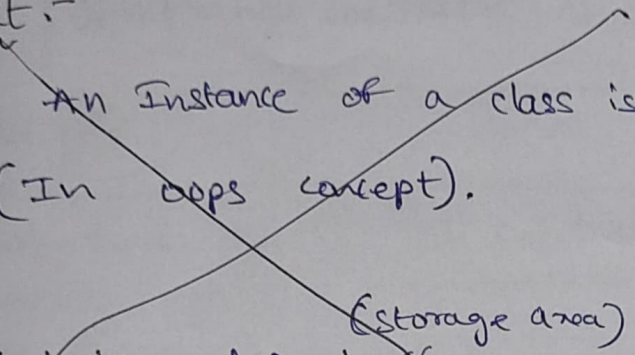
UNIT-2 :-

Class & Object :-

object :-

An Instance of a class is called object.

(In oops concept).



object is creates buffer area. This storage area consists variables & methods

Class :-

Class is a user defined blue print (or) prototype. which objects are created. Actually class is a logical view of entity. ones class has been defined, we can create no. of class related objects. It is way of binding to the variables and methods and its associated with together actually. in Java class is a user defined data type & it is ADT (Abstract data type).

Syntax :-

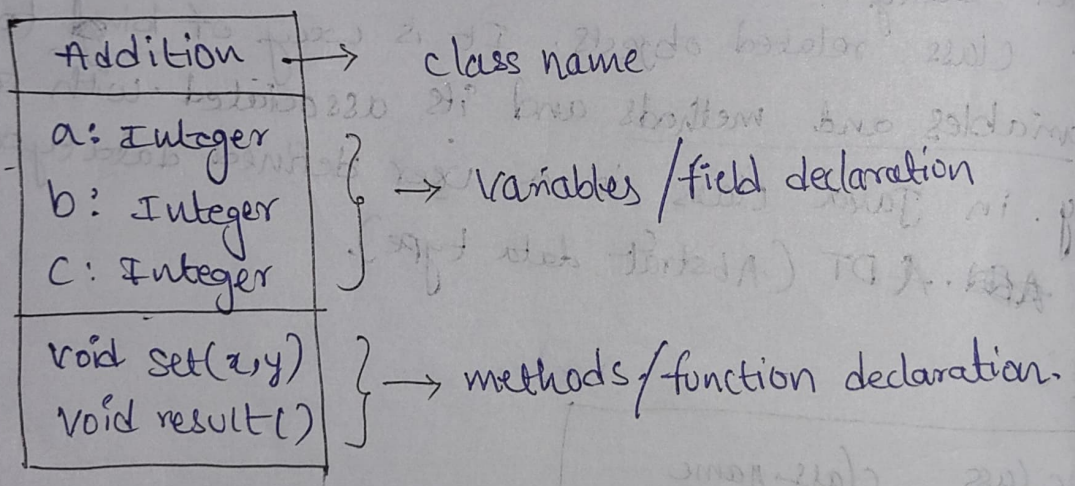
```
class class_name
{
    Variable Declaration
    Variable Declaration
}
```

Ex:-

```
Class Addition
{
  int a,b,c;
 void set (int x, int y)
 {
    a=x;
    b=y;
 }

 Void result ()
 {
    c=a+b;
    System.out.println("sum=" +c);
 }
}
```

Logical Architecture of class :-



Object :-

An Instance of a class is called object

Syntax:- ①

ⓐ

```
Class_name object;  // object declared.
object = new constructor(); // Instanticating
                            the object
```

Syntax:- ②

```
class_name object = new constructor();
```

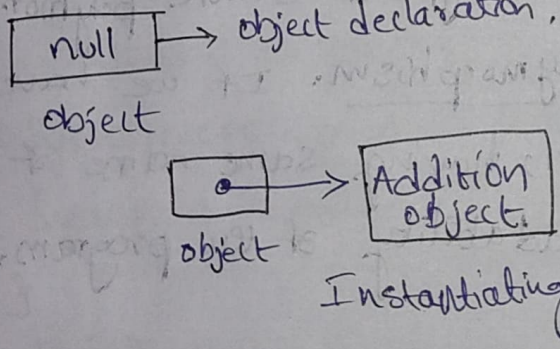
Example - ①

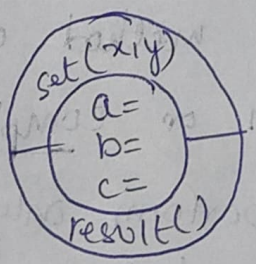
```
Addition obj;
Obj = new Addition()
```

Example - ②

```
Addition obj = new addition();
```

Memory Representation:-





* write any program using class & object.

```
Class Addition.
{
   int a,b,c;
  void set (int x, int y)
   {
      a=x;
      b=y;
   }
```

```
void result()
{
    c=a+b;
    System.out.println("sum="+c);
}
Public static void main (String args[])
{
    Addition obj = new Addition();
    obj.set(10,20);
    obj.result();
}
```

Method overloading:-

We can define multiple methods with the same name but with different parameters or parameter type. then this process is known as method overloading.

method overloading is a type of static polymorphism or compile time polymorphism. If we have to perform only one operation, having same name of the methods increases the readability of the program.

Advantages:-

method overloading increases the readability of program.

Different ways to method overloading:-

* By changing the number of parameters.
* By changing the datatype.

Program :-

```
class Rectangle
{ float l,b;
void rect (float l, float b)
{
    this . l=l;
    this . b=b;
    float ar= l*b;
    float p= 2*(l+b);
System.out.println ("Area of rectangle. =" +ar);
System.out.println ("parameter = "+p);
}
void rect (iut l, int b)
{
    this. l=l;
    this . b=b;
    float ar = l*b;
    float p= a*(l+b);
System.out.println ("Area of rectangle. =" +ar);
System.out.println ("perimeter = "+p);
}
void rect (float l)
{
    this l=l;
        b=l;
System.out.println ("Area of rectangle. =" + (l*b));
System.out.println ("perimeter =" + (a*(l+b));
}
public static void main (String args [])
```

```
{
    Rectangle V = new Rectangle();
    V.rect(4.2F);
    V.rect(4,5);
    V.rect(4.2F, 2.5F);
  }
}
```

Using Constructor :-

```
Class Rectangle
{
  float l,b;
  Rectangle (float l, float b)
  {
    this.l=l;
    this.b=b;
    float ar= l*b;
    float P= 2*(l*b);
  System.out.println("Area of rectangle = "+ar);
  System.out.println("perimeter = "+P);
  }
  Rectangle (int l, int b)
  {
    this.l=l;
    this.b=b;
    float ar= l*b;
    float P= 2*(l+b);
    System.out.println("Area of rectangle = "+ar);
    System.out.println("perimetter = "+P);
  }
```

```
Rectangle (float l)
{
    this.l = l;
      b = l;
    System.out.println("Area of rectangle = "+(l*b));
    System.out.println("perimeter = "+(2*(l+b));
}
public static void main (String args[])
{
    Rectangle r1 = new Rectangle (4,5);
    Rectangle r2 = new Rectangle (4.2F);
    Rectangle r3 = new Rectangle (4.2F, 5.2F);
}
}
```

Constructor :-

Constructor is a blocks of codes similar to the method at the time of calling constructor, memory is allocated for object that means constructor are instantiating for the object. It is special type of method used to initialise the object.

Every time an object is created using new operator (key word), atleast one constructor is called.

Rules for constructor :-

1. constructor name must be the same as its class name.
2. A constructor doesn't have any explicit return type
3. A java constructor can't be static, final, abstract.

Types of constructor:-

The constructor are classified into 2 types they are.

1. Parameter constructor.
2. Default constructor.

Paramiter constructor:-

A constructor has parameter then it is called parameter constructor.

Default constructor:-

A constructor doesn't have any parameters then it is called Default constructor.

Ex:-

```
Class circle
{
    final float Pi = 3.14F;
    float r;
    circle (float r)
    {
        this.r = r;
    System.out.println("Area of circle =" + (Pi * r * r));
    System.out.println("perimeter = " + (2* pi * r));
    }
circle ()
    {
        r = 5.0F;
System.out.println("Area of circle =" + (Pi * r * r));
System.out.println("perimeter=" + (2* Pi * r));
}
```

```
public static void main (String [] args)
  {
  Circle obj = new Circle (5.2F);
  Circle Obj2 = new Circle ();
  }
}
```

Static Members:-

Static is a key word in Java. This static keyword is used to declare to the member. than it is called static members. Here the members are include both properties (variables) and functions (methods). Static members are also known as class members. Static members are used in memory management.

Static Variable:-

Static variable are declared inside of the class with key word using "static". These variables are accessable inside of the entire class & methods. Static variables are create seperate memory location but not create in object. Static variable are also known as "class variable".

Syntax:-

Static <datatype> variable = <value>;

Ex:-

```
static int a=10;
```

Static Method :-

The static keyword declare to the method. then it is called static method. The static method can access only static variable and can't access non-static variables.

Syntax :-

```
Static <return type> method name (parameter list)
{
    method. Body
}
```

Ex :-

```
Static void rect (float x, float y)
{
    = =
    = =
}
```

Program :-

```
Class Rectangle
{ static float l,b;
  Static void rect (float x, float y).
  {
    l = x;
    b = y;
System. out. println ("Area of rectangle =" + (l*b));
System . out. println ("parameter =" + (2* (l+b));
}
```

```
public static void main(String args[])
{
    Rectangle.rect(4.2F, 5.2F)
}
}
```

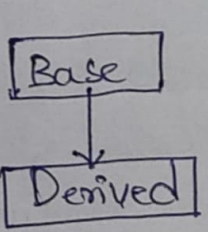
Inheritance :-

The processing of acquiring the qualities one class to another class is called inheritance. This qualities include both properties (variables) and functions (methods) of that class. Using inheritance, the "extends" the features of a class into another class. In Java, extend is a keyword. There are different types of inheritance

1. Single Inheritance.
2. Multilevel Inheritance.
3. Hierarchical Inheritance.
4. Multiple Inheritance

1. Single Inheritance

Deriving a new class based on only one base class. and the newly derived class is not other deriving new class. is called Single Inheritance.

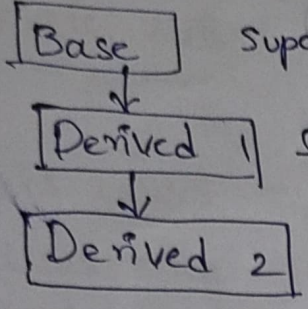


superclass / Existing / parent

Subclass / new / child.

2. Multilevel Inheritance :-

Deriving a class based on base class which is already derived from another class. is called multilevel inheritance.



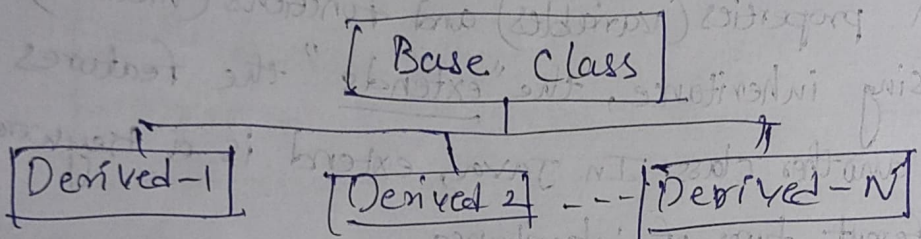
Super class / Existing / parent

sub class / new / child / parent / super.

sub class / new / child

Hierarchical Inheritance :-

Deriving a class based on existing class which is derived from "based class".



Single inheritence :-

```
class mysuper
{
 int a,b;
mysuper (int a, int b)
{
 this.a=a;
 this.b=b;
}
 void show()
{
  System.out.println("sum of two num="+(a+b));
 }
}
class mysub extends mysuper
{
 int c;
mysub (int a, int b, int c)
{
  super(a,b);
 this.c=c;
}
void show()
{
 System.out.println("sum of three num="+(a+b+c));
}
Public static void main (String args[])
{
  mysub obj= new mysub (10,5,6);
   obj.show();
  mysuper obj2 = new mysuper (20,5);
  obj2.show();
} }
```

Multilevel inheritence:-

```
class mysuper
{
int a,b;
mysuper(int a, int b)
{
this.a=a;
this.b=b;
}
void show()
{
System.out.println("sum of two num="+(a+b));
  }
}
class mysub extends mysuper
{
int c;
mysub(int a, int b, int c)
{
  super(a,b);
this.c=c;
}void show()
{
System.out.println("sum of three num="+(a+b+c));
  }
}
class mysub2 extends mysub
{
int d;
mysub2(int a, int b, int c, int d)
{
  super(a,b,c);
  this.d=d;
}
void show()
{
```

```
System.out.println("sum of four num="+(a+b+c+d));
}
public static void main(String args[])
{
  mysub2 obj= new mysub2(5,5,5,5);
  obj.show();
  mysub.obj2 = new mysub(10,5,6);
  obj2.show();
  mysuper obj3= new mysuper(20,5);
  obj3.show();
}
}
```

Over riding:-

Method overriding:-

We can define a method in sub class which is already defined in its superclass is called "method overriding".

In other words, when a method in subclass has the same name, same parameters and same return type as a method in its superclass. Then the method in the subclass is said to be override the method in the superclass.

Usage of method overriding:-

1. method overriding is used for runtime polymorphism
2. method overriding is used to specific implimentation of a method, that is already provided by its superclass.

Rules for method overriding:-

1. The method must have same name as in the superclass. (parent class)
2. The method must have same parameters as in the super-class.
3. The method must have same return type as in the superclass.
4. Must be a "IS-A" relationship

Program:-

```
class mysuper
{
  int a,b;
  mysuper(int a, int b);
  {
    This.a=a;
    This.b=b; }
```

```
void show()
{
  System.out.println("The sum of two num="+(a+b));
}
}

class mysub1 extends mysuper
{
  int c;
  mysub1(int a, int b, int c)
  {
    super(a,b);
    this.c=c;
  }
  void show()  //* This method already override
  {              in super class *//.
    System.out.println("The sum of three num="+(a+b+c));
  }
}
class mysub2 extends mysuper.
{
  int c,d;
  mysub2(int a, int b, int c, int d)
  {
    super(a,b);
    this.c=c;
    this.d=d;
  }
  void show()  //* This method already override in
  {                super class *//
    System.out.println("The sum of four num="+(a+b+c+d));
  }
  public static void main(String args[])
  {
    mysuper obj1 = new mysuper(5,7);
```

```
mysub1 obj2 = new mysub1 (5,7,9);
mysub2 obj3 = new mysub2 (6,5,7,8);
obj1.show();
obj2.show();
obj3.show();
}
}
```

Interface:-

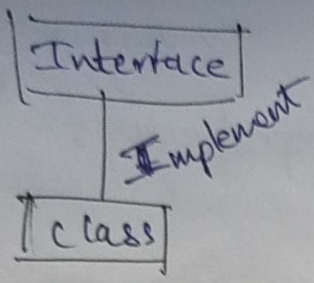
An interface in java is a blue print of a class. An interface is a mechanism to achieve data abstraction. It has final static variables and abstract method. there can be only abstract methods in an interface, not method body. This method body write in classes. It is used to achieve data abstraction and multiple inheritance in java.

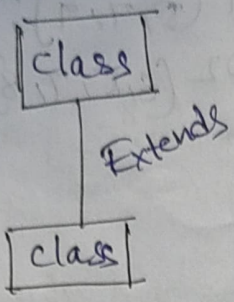
Program:-

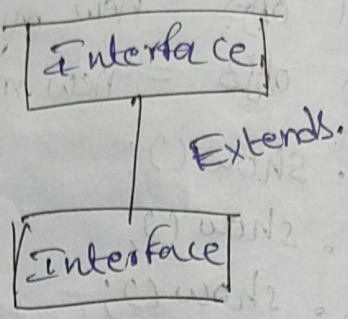
```
Interface myinterface.
{
    final static int a=4,b=2;
    public void add();
    public void mul();
```

Rules for interface:-

1. All the methods of an interface are public and abstract.
2. All the variables of an interface are final, static and public.
3. Interfacer method should not be static and final.
4. An interface can extend one or more other interfaces
5. An interface can't implement other interface.
6. An interface can implement no. of classes.







Syntax:-

```
interface interface_name
{
  [final static members]
  [Abstract members]
}
```

Implementing an Interface :-

It is a collection of abstract methods, we can't give a life.

It must be implemented in a class. Those classes, we can give life to the interfaces. there, uses "implement keyword" in class.

Syntax:-

```
class class_name implements interface_name
{
    class Definition
}
```

Program:-

```
interface myinterface.
{
  final static int a=4, b=2;
  Public void add();
  Public void mul();
}
Class compute implements myinterface
  {
    Public ~~static~~ void add ~~main~~ ().
    {
      System.out.println("The Sum of two values =" +(a+b));
    }
    Public void mul().
    {
    System.out.println("The mul of two values = "+(a*b));
    }
  Public static void main (String args[])
  {
     Compute obj = new compute();
     obj.add();
     obj.mul();
  }
}
```

Extending an interface :-

Like classes, interface can also be extended. An interface can extend the ~~futures~~ features one interface to another interface. Here features include abstract ~~method~~ methods and final static variables

Syntax:-

```
Interface interface_name2 Extends interface_num1
{
    Abstract methods.
    final static variables.
}
```

Program:-

```
Interface myinterface1
{
    final static int a=4, b=2;
}
Interface myinterface2 extends myinterface1
{
    public void add();
    public void mul();
}
Class compute implements myinterface2
{
    Public void add()
    {
        System.out.println("The sum of two values = "+(a+b));
    }
    Public void mul()
    {
        System.out.println("The mul of two values = "+(a*b));
    }
    Public static void main(String args)
    {
```

```
compute obj = new compute();
    obj.add();
    obj.mul();
}
}
```

Final Keyword :-

Final is a keyword in java. It is nothing but constants and Doesn't change during the execution of the programme. The Final keyword is a non-access modifier. used for classes, variables and methods.

Final keyword is used to declare in different ways.....

1. Used to variable (final variable)
2. Used to method (final methods)
3. Used to class. (final class)

Final variables :-

final is declare to variable and it's performs "constants". It is fixed value and doesn't change during the exeution of the programe.

Syntax :-

```
final Datatype variablename = value;
```

Example :-

```
final double Pi = 3.14F;
final int a = 4;
```

Final methods :-

Final keyword is used to declare methods. and it performs can't overwritten, ~~of the~~ (stop overriding technique)

Syntax:-

```
Final <return type> <methodname> (parameter lists)
{
  - - -
  - - -
}
```

Ex:-

```
final void show() //* stop overriding method in Subclass*//
{
  - - -

}
```

Final class:-

final keyword is used to declare class and it performs can't extends features one class to another class. that means stop inheritence technique.

~~Syntax~~ Example:-

```
final class mysuper
{
  - - - - -

}
class mysub extends mysuper //* stop inheritance technique*//
```

{

- -
- -

}

Syntax:-

```
Final class class-name
{
    - -
    - -
}
```

Exception:-