14-01-2025

Abstraction - hiding the details

0 % - 100 % abstraction - Abstract class

// only abstract methods - case 1

// only non abstract methods - case 2

// both methods - case 3

// empty abstract class - case 4

// abstract class must be inherited

// abstract class cannot be instantiated.

// an abstract class can define constructor.

// an abstract class can holds static method.

abstract - keyword

```
abstract class A
 {
    abstract void show ( );
    abstract void add ( int a , int b) ;
 }
class B extends A
 {
   void show ( )
     {
       Sopln (" I have completed the show ") ;
     }
    void add ( int a , int b )
     {
       int c ;
       c = a+b ;
       Soplin (" sum = " + c) ;
   }   }
class Main
 {
    public static void main ( String args [ ] )
     {
         B b = new B ( ) ;
         b.show ( ) ;
         b. add (12, 13) ;
     }
 }
```

* If child class is unable to implement all/some abstract method of its parent, then we declare the child class as abstract class.

```
abstract class B extends A
 {

 }
 // empty abstract class is also valid
```

* In this case, you must inherit the abstract class.

```
abstract class B extends A
 {

 }
class C extends A
 {
   void show ()
    {
     Sopln (" I have completed the show ");
    }
   void add (int a, int b)
    {
      int c;
      c = a+b;
      Soplr (" Sum = " + c);
 }  }
class Main
 {
   public static void main (String args[])
    {
      C ob = new C();
      ob.show();
      ob.add (12, 13);
    }
 }
```

// Output

I have completed the show

Sum = 25.

I12.java (based on 2nd case)

```
// abstract class only non abstract methods
abstract class A
{
    void show()
    {
        Sopln("I have completed the show");
    }
    void add(int a, int b)
    {
        int c;
        c = a+b;
        Sopln("sum = "+c);
    }
}
class B extends A
{
    void display()
    {
        Sopln("CSE");
    }
}
// Driver class
class Main
{
    public static void main(String ax[])
    {
        B ob = new B();
        ob.show();
        ob.add(12,13);
        ob.display();
    }
}
```

// Output

I have completed the show

Sum = 25

CSE

* we can't create object but ↓

```
class Main
{
    public static void main(String ax[])
    {
        A ob = new B();
        ob.show()
        ob.add(10,11);
        ob.display();
    }
}
```

Output: Error

↓ bcoz now child can't access parent

```
class Main
{
    public static void main(String ax[])
    {
        A ob = new B();
        ob.show()
        ob.add(10,11);
        // ob.display();
    }
}
```

Output: I have comp...

Sum = 25

Constructor of Abstract class :-

```
abstract class A
{
   A()
    {
      Sopln("I am constructor of abstract class A");
    }
     abstract void show();
     abstract void add(int a, int b);
}
  abstract class B extends A
   {
      B();
       {
         Sopln("I am constructor of abstract class B");
       }
  }
 class C extends B
   {
    C();
     {
        Sopln("I am constructor of class C");
      }
      void show()
       {
          Sopln("I have completed the show");
       }
       void add(int a, int b)
        {
           int c;
           c = a+b;
           Sopln("Sum = "+c);
         }
         class Main
          {
            public static void main(String as[])
             {
                C ob = new C();    // chaining of constructor.
                ob.show();
                ob.add(12,13);
             }
          }                  // chaining of the constructor
```

I am the constructor of abstract class A

I am the constructor of abstract class B

I am the constructor of class C

I have completed the show

Sum = 25

I10.java

* child class of an abstract class is abstract.
* we can take abstract variable of an abstract class.

```
abstract class A
 {
   A()
    {
      Sopln (" I am constructor of abstract class A");
    }
      Static void show ()
       {
         Sopln (" I have completed the show") ;
       }
      abstract void add (int a, int b);
  }
   abstract class B extends A
    {
      B();
       {
         Sopln ("I am constructor of abstract class B");
       }
    }
 class C extends B
    {
       B()
        {
          Sopln ("I am constructor of abstract class B");
        }
    }
 class C extends B
   {
    CC()
     {
        Sopln ("I am constructor of class C");
      }
        void add (int a, int b)
           {
              int c ;
               c= a+b;
               Sopln (" Sum = " + c);
           }
      }
     class Main
       {
          public static void main (String args[])
            {
               A ob = new C ();
               A.show ();
                ob. add (12, 13);
       }    }
```

A11.java

```
abstract class A
 {
   int a, b;
   static final int q = 25;
   A ( int x, int y)
    {
      a = x;
      b = y;
      Sopln (" I am the constructor of class A");
   }
     static void show ( )
      {
        Sopln (" I have completed the show");
        Sopln (" Value of q = "+q);
      }
        abstract void add();
    }
    class c extends A
     {
       int c;
       C ( int x, int y, int z)
        {
          super (x, y);
          c = z;
          Sopln (" I am the constructor of class C");
        }
          void add ()
           {
             int p;
             p = a+b+c;
             Sopln (" Sum = "+c);
           }
        }
  class Main
   {
     public static void main (String args [])
      {
          A c = new C (10, 20, 30);
          A.show ();
          c. add ();
       }
   }
```

Inheritance - code reusability

parent class (super class)
child class (sub class)

* child class can access data or methods of parent class but vice-versa isn't possible

Types of Inheritance:-

1) Single Level Inheritance

A ------> B (one parent → one child)

2) Multi Level Inheritance

A ----> B ----> C (grandparent included)

3) Hierarchical Inheritance

A -----> B

A -----> C

4) Multiple Inheritance (not supported by Java)

// Single Level Inheritance :-

```
class A
{
  void show ( )                    → method
   {
     Sopln ("I am a parent class");
   }
}
                                    → extend is a keyword
class B extends A                   → class B is the child
{
  void display ( )
   {
     Sopln ("I am the child of class A");
   }
}
  class Main
```

*Imp

```
    {
        public static void main (String as[])
        {
            B ob = new B();          → Created object of B.
            ob.show();
            ob.display();
        }
    }
```

// output :-

I am the parent class
I am the child of class A.

```
A ob = new A();   → wrong code bcoz parent have no
ob.show();          access to child.
ob.display();       (compile time error)
```

```
class A
{
  int a, b;
  A(int x, int y)
  {
    a = x;
    b = y;
  }
    void show()
    {
      Sopln (a+ "  " +b);
    }
}
  class B extends A
  {
    int c;
```

```
B(int p, int q, int r)
 {
   super(p,q);
   c=r;
 }
void display()                    → instance method
 {
   Sopln(a+"    "+b+"    "+c);
 }
}
 class Main
 {
   public static void main(String ar[])
    {
      B ob= new B(10,20,30);      // parameterised constr.
       ob.show();                 } calling show & display
       ob.display();
    }
 }
// Output
        10      20
        10      20      30
```

super is a keyword in java which is

- used to call immediate parent class constructor
- used to refer variable of parent class
- or method of parent class if their names are same.

```
class A
{
  int a;
  A(int x)
  {
    a=x;
  }
    void show()
    {
      Sopln(a);
    }
  }
  class B extends A
  {
    int a;
    B(int p, int q)
    {
      super(p);
      a = q;
    }
      void display()
      {
       Sopln( super.a + "   " + a);
      }
    }
      class Main
      {
       public static void main(String ar[])
        {
          B ob = new B(10,20);          // Output
          ob.show();                    10
          ob.display();                 10   20
        }
      }
```

Method Overriding :-

```
class A
 {
   void show ()
    {
      Sopln (" CSE");
    }
  }
   class B extends A
    {
     void show ()
     {
      Sopln (" KIIT UNIVERSITY ");
     }
}
  class Main
{
   public static void main ( String ar[])
    {
     B ob = new B();
     ob. show();
     ob. show();
    }
}
```

// Output :-

KIIT UNIVERSITY

KIIT UNIVERSITY

* method of parent will be overridden by child.

```
class A
 {
   void show ()
    {
      Sopln (" CSE");
    }
  }
   class B extends A
   {
     void show ()
     {
       super.show ();              // call parents method
       Sopln (" KIIT UNIVERSITY");
     }
  }
   class Main
  {
    public static void main ( String ar[])
     {
       B ob = new B();
       ob.show();
     }
  }
```

```
CSE
KIIT UNIVERSITY
```

Final Keyword :-

```
class A
{
   final void show ()
    {
      Sopln (" CSE");
    }
}
  class B extends A
  {
    void show ()
    {
     Sopln (" KIIT UNIVERSITY ");
    }
}
  class Main
{
   public static void main ( String ar[])
    {
     B ob = new B();
     ob. show();
     ob. show();
    }
}
```

final keyword :-
→ to avoid method overriding (final decision)

// Output : error
"overriding is not possible"

Use 2 — Using final Keyword, we can prevent inheritance.

(declare in class declaration).

```
final class A
  {
    void show ()
   {
    Sopln ("CSE")
   }
```

// Output :- error
can't inherit

Use 3 :- we can declare a variable as final.

```
class A
 {
   final int a=5;
   void show()
    {
      Sopln(a++);
    }
   }
     class B extends A
     {
       void show1()
        {
          Sopln(a--);
        }
      }
       class Main
       {
         public static void main(String ar[])
          {
            B ob = new B();
            ob.show();
            ob.show1();
          }
       }
```

// output :- 2 errors

we cant modify a=5 .

* it acts as a constant.

1 parent → many child (hierachical)

Dynamic Method Dispatch (run time polymorphism)

```
class A
  { void show() {
      Sopln ("Inside A's show method");
    } }
class B extends A
  { // Overriding show
    void show()
    {
      Sopln (" Inside B's show method");
    }
  }
class C extends A
  { // Overriding show
    void show()
    {
      Sopln (" Inside C's show method");
    }
  } // Driver class
    class Main
    {
      public static void main (String args[])
      {
        A a = new A(); // object of type A
        B b = new B();
        C c = new C();
        A ref ;          //obtain a reference of type A;
        ref = a ;
        ref.show(); // calling A's version of show()
        ref = b ;        // now ref refers to a B object
        ref.show();
        ref = c          // now ref refers to a C object.
        ref.show(); // calling C's version of show().
      }
    }
```

SUMMARY

1) Inheritance - code reusability
   - child class can access parent's method & variables but parent class can't access child's.

2) types of inheritance :- i) Single Level (one parent → one child)
   ii) Multi Level (parent → child → grandchild)
   iii) Heirarchical (one parent → multiple child)
   iv) Multi-Level Inheritance
   ↳ not supported in Java

3) • extends - used for inheriting a parent class

4) final keyword - prevents overriding
   • prevents inheritance
   • constant variable final int x = 10;

5) super
   • keyword to call immediate parent class constructor
   • used to refer variable of parent class or method of parent class if their names are same.

6) Dynamic Method Despatch
   * also known as Runtime Polymorphism
   * enables method calls to be resolved at runtime based on actual object type, not the reference type.
