Programming In Java For Web Application

CSA-0985

Name : S. Anitha.

Reg No : 192324046.

Dept : B.Tech (AI & DS).

Date : 27/07/24.

Day : Saturday.

Inheritance:

Inheritance is a fundamental concept of object oriented programming (OOP) that allows a new class to inherit fields and methods from an existing class.

Single Inheritance:

Single Inheritance is a type of inheritance in which a subclass inherits from only one superclass.

class A

↓

class B.

```
class A {
    int a;
    void display A()
    {
        system.out.println ("a:" +a);
    }
}
```

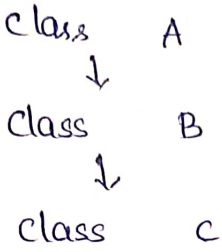
```
class B extends A {
        int b;
        void display B()
        {
          system.out.println ("b=" +b);
        }
      }

Public class Main {
      Public static void main (string[] args)
      {
        B obj = new B();
        obj.A = 10;
        obj.b = 20;
        obj. display A();
        obj. display B();
      }
}
```

10

20

Multi level Inheritance:

Multilevel inheritance is a type of inheritance in Java where a class is derived from a class that is also derived from another class.

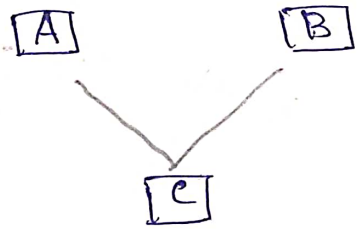



```
class A {
    Public void method A()
    {
        system . out. println ("Inside class A");
    }
}

class B extends A{
        Public void method B()
        {
            system . out . println ("Inside class B");
        }
}

class C extends B{
        Public void method c()
        {
            system . out println ("Inside class c");
```

```
Public class Main
{
    Public static void main (string[] args)
    {
        C.obj c : new c();
        obj c . method A();
        objc . method B();
        objc . method c();
    }
}
```

Multiple Inheritance:

Multiple inheritance refers to column features in some object oriented programming languages when a class can inherit characteristics from more that one percent class.

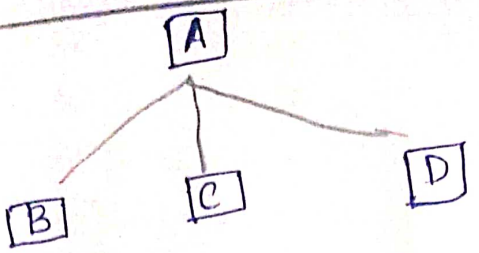



```
Class A
{
  void display A()
    {
      system.out.println ("Inside class A");
    }
}
```

```
class B
  {
     void displayB()
      {
        System.out.println("Inside class B")
      }
  }
class C extends A()
    {
      void display A()
       {
         super display();
         System.out.println("Inside class c");
       }
       void display c()
        {
          system.out.println("Method of class c");
        }
    }
```

Hierarchial Management:

It occurs when multiple sublcales inherit from a single superclass. This means that a single parent class can have multiple child cases.



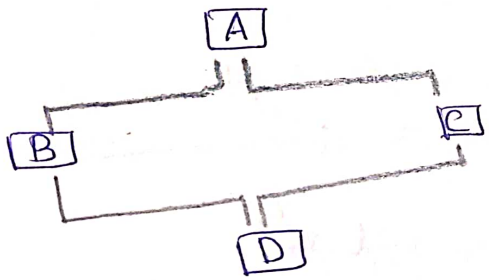

```
class A {
   Public void displayA()
    {
      System.out.println ("Inside class A");
    }
}
class B extends A{
     Public void displayB()
      {
        System.out.println ("Inside class B");
      }
     }
class C extends A {
     Public void displayC()
     {
       System.out.println ("Inside class D");
     }
    }
Public class Main
 {
    Public static void main (String[] args)
     {
        B objB = new B();
        C objC = new C();
        D objD = new D();
```

}

```
obj B. display A();
obj B. display B();
System. out. println();

obj c . display A();
obj c . display C();
System . out . println();

obj D. display A();
obj D. display B();
obj D. display D();
}
}
```

Hybrid Inheritance:

Condition of any inheritance.




```
class A
{
Public void display A()
{
system. out. println ("Inside class A");
}
}
```

```
class B
 {
  Public void display B()
    {
     system. out. println ("Inside class B")
    }
 }
class C extends A {
     Public void display C()
      {
       system. out. println ("Inside class C")
      }
     }
class D extends C
  {
   Public static void main (String[] args)
     {
        C objC  =new C() ;
        D objD  = newD():
        obj C .displayA();
        obj C . displayC();
        System. out. println();

        obj D . display();
        obj D. display C();
        obj D . diplay D();
      }

  }
```

Exception Handling:

An exception is an error during the execution of a program.

Key components of Exception Handling:

Try
Catch
Throw
Finally
Throws

Nested catch:

```
class main {
    Public static void main (string[] args)
    {
        try
        {
            int a = 5/10;
        system. out. println ("Rest or code in key block");
        }
        catch (Arithmetic exception e)
        {
            system.out. println ("Arithmetic expression" +e.
                                    get message());
        }
        catch (exception e)
        {
            system. out. println ("Exception =" +e.getmessage());
        }
```

```
    }
}
```

Arithmetic exception by zero.

Try & catch:

```
class Main {
    Public static void main (string[ ]  args)
    {
      try
       {
          int  b=1/0;
       }
      catch exception.e
      {
      system. out. println ("Exception known" +e. get Message());
       }
      system. out. println  (a[4]);
    }
    catch ( Array  Index  order  bound  Exception)
       {  system. out. println ("Exception thrown" +e. getMessage());
        }
      system. out. println  ("Out  of  Blocks")
      }
}
```

Exception thrown by 0.

throw :

```
Public class Main {

    static void checkage (int age) throws Arithmetic exception
    {
        if (age < 19)
     }
    throw new arithmetic exception ("Access defined -
                                    You must be at-least
                                    18 year old");

    else
      {
        system.out.println ("Access granted - You are
                                    old enough");
      }
    }
    Public static void main (String[] args)
      {
        try
        {
          check age (16)
        }
        catch (arithmetic exception e)
          {
            system.out.println (e.get message());
          }
        }
      }
```

You are not eligible.

Finally:

Try → block of code to test the error being executed.

Catch → Block of code to be executed if an error occurs in try block.

Finally → Code that always executes.

The Finally block is a section of code that is executed regardless of whether an exception is thrown or not.

```
Public        Main {

Public static void main (String[] args)
 {
   try
   {
    int [] = {1,2,3}
     system.out.println ("Rest of code in the
                          try block");
     }
   catch (Array index out of Bound exception")
  {
  system.out.println ("Array index out of
                       exception:"+e.Messagel));
  }
  Finally {
     system.out.println ("This is the finally block");
     }
   }
```

Arithmetic ⇒ / zero.