****
* Class Name : VJTech Academy , Maharashtra   *
* Author Name: Vishal Jadhav Sir   *
* Mobile No : 9730087674   *
****

==========
UNIT-III : Inheritance
==========

***Inheritance:

- The process of creating new class from old class is known as Inheritance.
- The mechanism of acquiring the properties of old class into the new class class is known as Inheritance.
- The newly created class is known as Subclass/child/derived class.
- Old class is known as Super class/Parent class/Base class.
- Inheritance will help us to achieve reusability feature.
- Because of Inheritance, our development time will get save and it will also impact on the project cost.

- Types of Inheritance:

1) Single Inheritance
2) Multi-level Inheritance
3) Multiple Inheritance
4) Hierarchical Inheritance
5) Hybrid Inheritance

- We use following syntax for creating the new class from old class.

```
class DerivedClassName extends BaseClassName
{
        //body of Derived Class
}
```

1) Single Inheritance:

- This is one of the types of inheritance.
- To create new class from only one base class is known as single inheritance.
- Syntax:

```
class DerivedClassName extends BaseClassName
{
        //body of Derived Class
}
```

- Program:

```
//Single Inheritance
import java.util.*;
class Student
{
        int rollno;
        String name;
```

```
        void get_stud_info()
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter Student Roll No:");
                rollno=sc.nextInt();
                System.out.println("Enter Student Name:");
                name=sc.next();
        }
        void disp_stud_info()
        {
                System.out.println("Student ROll No:"+rollno);
                System.out.println("Student Name:"+name);
        }
}
class Test extends Student
{
        int marks1,marks2;
        void get_stud_marks()
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter Student Test-1 Marks:");
                marks1=sc.nextInt();
                System.out.println("Enter Student Test-2 Marks:");
                marks2=sc.nextInt();
        }
        void disp_stud_marks()
        {
                System.out.println("Test-1 Marks:"+marks1);
                System.out.println("Test-2 Marks:"+marks2);
        }
}

class SingleInheritanceDemo
{
        public static void main(String args[])
        {
                Test t1=new Test();
                t1.get_stud_info();
                t1.get_stud_marks();
                System.out.println("*******STUDENT INFORMATION SYSTEM********");
                t1.disp_stud_info();
                t1.disp_stud_marks();
        }
}
/*
Enter Student Roll No:
1010
Enter Student Name:
James
Enter Student Test-1 Marks:
```

```
89
Enter Student Test-2 Marks:
78
*******STUDENT INFORMATION SYSTEM********
Student ROll No:1010
Student Name:James
Test-1 Marks:89
Test-2 Marks:78
*/
```

2) Multi-level Inheritance:

- The mechanism of deriving the class from another derived class is known as multi-level inheritance.
- To create new class from another derived class is known as multi-level inheritance.
- It is one of the types of inheritance.
- Syntax:

```
        class BaseClass1
        {
                        //body of BaseClass1 Class
        }
        class DerivedClass1 extends BaseClass1
        {
                        //body of DerivedClass1
        }
    class DerivedClass2 extends DerivedClass1
        {
                        //body of DerivedClass2
        }
```

-Example

```
//Multi-level Inheritance
import java.util.*;
class Student
{
        int rollno;
        String name;
        void get_stud_info()
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter Student Roll No:");
                rollno=sc.nextInt();
                System.out.println("Enter Student Name:");
                name=sc.next();
        }
        void disp_stud_info()
        {
```

```
                System.out.println("Student ROll No:"+rollno);
                System.out.println("Student Name:"+name);
        }
}
class Test extends Student
{
        int marks1,marks2;
        void get_stud_marks()
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter Student Test-1 Marks:");
                marks1=sc.nextInt();
                System.out.println("Enter Student Test-2 Marks:");
                marks2=sc.nextInt();
        }
        void disp_stud_marks()
        {
                System.out.println("Test-1 Marks:"+marks1);
                System.out.println("Test-2 Marks:"+marks2);
        }
}
class Result extends Test
{
        int total_marks;
        void get_total_marks()
        {
                total_marks=marks1+marks2;
        }
        void disp_total_marks()
        {
                System.out.println("Total Marks:"+total_marks);
        }
}

class MultilevelInheritanceDemo
{
        public static void main(String args[])
        {
                Result t1=new Result();
                t1.get_stud_info();
                t1.get_stud_marks();
                t1.get_total_marks();

                System.out.println("*******STUDENT INFORMATION SYSTEM********");
                t1.disp_stud_info();
                t1.disp_stud_marks();
                t1.disp_total_marks();
        }
}
/*
```

```
Enter Student Roll No:
1010
Enter Student Name:
James
Enter Student Test-1 Marks:
78
Enter Student Test-2 Marks:
90
*******STUDENT INFORMATION SYSTEM********
Student ROll No:1010
Student Name:James
Test-1 Marks:78
Test-2 Marks:90
Total Marks:168
*/
```

3) Multiple Inheritance:

- To create new class from more than one base class is known as multiple inheritance.
- But in java, we cannot derive multiple base classes properties in derived class.
- If you want to achieve this scenario then you can use alternate solution that is interface.
- Exmaple:

```
class A extends B extends C
{
        //body of A.
}
```

- Above scenario is not allowed in java language.

4) Hierarchical Inheritance:

- To create more than one derived classes from only one base class is known as Hierarchical inheritance.
- It is one of the types of inheritance.
- We use following syntax for creating the new class from old class.

```
class DerivedClassName extends BaseClassName
{
        //body of Derived Class
}
```

- Program:

```
//Hierarchical Inheritance
import java.util.*;
class Student
{
        int rollno;
        String name;
        void get_stud_info()
        {
```

```
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter Student Roll No:");
                rollno=sc.nextInt();
                System.out.println("Enter Student Name:");
                name=sc.next();
        }
        void disp_stud_info()
        {
                System.out.println("Student ROll No:"+rollno);
                System.out.println("Student Name:"+name);
        }
}
class Test extends Student
{
        int marks1,marks2;
        void get_stud_marks()
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter Student Test-1 Marks:");
                marks1=sc.nextInt();
                System.out.println("Enter Student Test-2 Marks:");
                marks2=sc.nextInt();
        }
        void disp_stud_marks()
        {
                System.out.println("Test-1 Marks:"+marks1);
                System.out.println("Test-2 Marks:"+marks2);
        }
}
class Sport extends Student
{
        float sport_wt;
        void get_sport_info()
        {
                sport_wt=8.9f;
        }
        void disp_sport_info()
        {
                System.out.println("Sport Weightage:"+sport_wt);
        }
}

class HierarchicalInheritanceDemo
{
        public static void main(String args[])
        {
                Test t1=new Test();
                System.out.println("*******Test Class Implementation********");
                t1.get_stud_info();
                t1.get_stud_marks();
```

```
                t1.disp_stud_info();
                t1.disp_stud_marks();

                Sport s1=new Sport();
                System.out.println("*******Sport Class Implementation********");
                s1.get_stud_info();
                s1.get_sport_info();
                s1.disp_stud_info();
                s1.disp_sport_info();

        }
}
/*
*******Test Class Implementation********
Enter Student Roll No:
1010
Enter Student Name:
James
Enter Student Test-1 Marks:
89
Enter Student Test-2 Marks:
78
Student ROll No:1010
Student Name:James
Test-1 Marks:89
Test-2 Marks:78
*******Sport Class Implementation********
Enter Student Roll No:
1010
Enter Student Name:
James
Student ROll No:1010
Student Name:James
Sport Weightage:8.9
*/
```

5) Hybrid Inheritance:

- The combination of more than one types inheritance is known as hybrid inheritance.
- Example:

```
//Hybrid Inheritance
import java.util.*;
class Student
{
        int rollno;
        String name;
        void get_stud_info()
        {
                Scanner sc=new Scanner(System.in);
```

```
                System.out.println("Enter Student Roll No:");
                rollno=sc.nextInt();
                System.out.println("Enter Student Name:");
                name=sc.next();
        }
        void disp_stud_info()
        {
                System.out.println("Student ROll No:"+rollno);
                System.out.println("Student Name:"+name);
        }
}
class Test extends Student
{
        int marks1,marks2;
        void get_stud_marks()
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter Student Test-1 Marks:");
                marks1=sc.nextInt();
                System.out.println("Enter Student Test-2 Marks:");
                marks2=sc.nextInt();
        }
        void disp_stud_marks()
        {
                System.out.println("Test-1 Marks:"+marks1);
                System.out.println("Test-2 Marks:"+marks2);
        }
}
class Result extends Test
{
        int total_marks;
        void get_total_marks()
        {
                total_marks=marks1+marks2;
        }
        void disp_total_marks()
        {
                System.out.println("Total Marks:"+total_marks);
        }
}
class Sport extends Student
{
        float sport_wt;
        void get_sport_info()
        {
                sport_wt=8.9f;
        }
        void disp_sport_info()
        {
                System.out.println("Sport Weightage:"+sport_wt);
```

```
        }
}
class HybridInheritanceDemo
{
        public static void main(String args[])
        {
                System.out.println("*******RESULT CLASS IMPLEMENTATION********");
                Result r1=new Result();
                r1.get_stud_info();
                r1.get_stud_marks();
                r1.get_total_marks();

                r1.disp_stud_info();
                r1.disp_stud_marks();
                r1.disp_total_marks();

                System.out.println("*******SPORT CLASS IMPLEMENTATION********");
                Sport s1=new Sport();
                s1.get_stud_info();
                s1.get_sport_info();
                s1.disp_stud_info();
                s1.disp_sport_info();
        }
}
/*
*******RESULT CLASS IMPLEMENTATION********
Enter Student Roll No:
1010
Enter Student Name:
James
Enter Student Test-1 Marks:
78
Enter Student Test-2 Marks:
99
Student ROll No:1010
Student Name:James
Test-1 Marks:78
Test-2 Marks:99
Total Marks:177
*******SPORT CLASS IMPLEMENTATION********
Enter Student Roll No:
1010
Enter Student Name:
James
Student ROll No:1010
Student Name:James
Sport Weightage:8.9
*/
```

Method Overriding:

==========
- Suppose, base class and derived class method names are same.
- When base class method derived in derived class then it got overide.
- It means base class method overriden by derived class method.
- To call overriden method, we can use super keyword.
- We use syntax for calling hidden method: super.methodName();

- Program:

```
//Method overriding and use of super keyword
class Base
{
        void display()
        {
                System.out.println("display() method of base class");
        }
}
class Derived extends Base
{
        void display()
        {
                super.display();
                System.out.println("display() method of derived class");
        }
}
class MethodOverriding
{
        public static void main(String args[])
        {
                Derived d1=new Derived();
                d1.display();
        }
}
/*
display() method of base class
display() method of derived class
*/
```

==========
How to invoke Base class Constructor
==========
- Base class constructor should not inherited in its sub class.
- Suppose, base class contain constructor then how we can call that constructor.
- In this case , we can super keyword.
- super keyword should be the first line of derived class constructor body.
- Syntax:

```
                super();                                //to invoke default
constructor
                or
                super(argument list)    //to invoke parameterized constructor
```

-Program1:

```
//use of super keyword for calling base class default constructor.
class Base
{
        Base()
        {
                System.out.println("Base class constructor called...!!!");
        }
}
class Derived extends Base
{
        Derived()
        {
                super();
                System.out.println("Derived class constructor called...!!!");

        }
}
class InvokeBaseClassConstructor
{
        public static void main(String args[])
        {
                Derived d1=new Derived();
        }
}
/*
Base class constructor called...!!!
Derived class constructor called...!!!
*/
-Program2:
//use of super keyword for calling base class parameterized constructor.
class Base
{
        int x;
        Base(int m)
        {
                x=m;
                System.out.println("Base class constructor called..m="+m);
        }
}
class Derived extends Base
{
        int y;
        Derived(int p,int q)
        {
                super(p);
                y=q;
                System.out.println("Derived class constructor called..q="+q);

        }
}
```

```
class InvokeBaseClassConstructor1
{
        public static void main(String args[])
        {
                Derived d1=new Derived(100,200);
        }
}
/*
Base class constructor called..x=100
Derived class constructor called..y=200
*/
```