Dt : 21/8/2023

Ex-Application:(Demonstrating Instance Blocks in PClass and CClass)

ProjectName : Inheritance_App8

packages,

p1 : PClass.java

```
package p1;
public class PClass
{
   public int a;

   {
        System.out.println("****PClass-Instance
Block****");
        System.out.println("The value a:"+a);
   }
}
```

p1 : CClass.java

```
package p1;
public class CClass extends PClass
{
     {
          System.out.println("****CClass-Instance
Block***");
          System.out.println("The value a:"+a);
     }
}
```

p2 : DemoInheritance8.java(MainClass)

package p2;

```
import p1.*;
import java.util.*;
public class DemoInheritance8
{
        public static void main(String[] args)
        {
      Scanner s = new Scanner(System.in);
      CClass ob = new CClass();
      System.out.println("Enter the value for a:");
      ob.a = s.nextInt();
      System.out.println("****main()****");
      System.out.println("The value a:"+ob.a);
      s.close();
        }
}
```

o/p:

```
****PClass-Instance Block****
The value a:0
****CClass-Instance Block***
The value a:0
Enter the value for a:
120
```

*******main()*******

The value a:120

==========

Note:

=>Instance blocks from PClass are executed first,then Instance blocks from CClass are executed.

==========

faq:

define Method Overloading process?

=>More than one method with same method name but differentiated by their Para_list or Para_type is known as Method Overloading process.

Note:

(i)return_type of methods are not included in Method Overloading process.

(ii)To perform Method Overloading process,Inheritance is not manditory which means we can perform Method Overloading using single class also.

(iii)we can perform Instance method Overloading,Static method Overloading and Constructor Overloading.

Ex-1 : (Demonstrating Constructor Overloading process)

ProjectName : Inheritance_App9

packages,

p1 : PClass.java

```
package p1;
public class PClass {
     public PClass(int a)
     {
     System.out.println("****PClass(a)****");
     System.out.println("a:"+a);
     }
     public PClass(int a,int b)
     {
     this(a);
     System.out.println("****PClass(a,b)****");
     System.out.println("a:"+a);
     System.out.println("b:"+b);
     }
}
```

p1 : CClass.java

```
package p1;
public class CClass extends PClass
{
     public CClass(int a,int b,int c)
     {
     super(a,b);
     System.out.println("*****CClass(a,b,c)****");
     System.out.println("a:"+a);
     System.out.println("b:"+b);
     System.out.println("c:"+c);

     }
     public CClass(int a,int b,int c,int d)
     {
     this(a,b,c);//3_para_Con_call
     System.out.println("*****CClass(a,b,c,d)****");
```

```
        System.out.println("a:"+a);
        System.out.println("b:"+b);
        System.out.println("c:"+c);
        System.out.println("d:"+d);

        }
}
```

p2 : DemoInheritance9.java(MainClass)

```
package p2;

import java.util.*;

import p1.*;

public class DemoInheritance9 {

        public static void main(String[] args) {

      Scanner s = new Scanner(System.in);

      System.out.println("Enter the value a:");

      int a = s.nextInt();

      System.out.println("Enter the value b:");

      int b = s.nextInt();

      System.out.println("Enter the value c:");

      int c = s.nextInt();

      System.out.println("Enter the value d:");

      int d = s.nextInt();


      CClass ob = new CClass(a,b,c,d);//4_para_COn_call
```

```
        s.close();
          }
}
```

o/p:

```
Enter the value a:
11
Enter the value b:
12
Enter the value c:
13
Enter the value d:
14
****PClass(a)****
a:11
****PClass(a,b)****
a:11
b:12
*****CClass(a,b,c)****
a:11
b:12
c:13
```

*****CClass(a,b,c,d)****

a:11

b:12

c:13

d:14

==========

Note:

=>while Object creation process we can call only one constructor for execution,but other Constructors can also be executed using Constructor interlinking process or Constructor Chaining process.

super() - Interlinking of PClass and CClass Constructors

this() - Interlinking of Constructors from the same class

==========

Ex-2 :(Demonstrating Instance method Overloading process)

ProjectName : Inheritance_App10

packages,

p1 : PClass.java

```
package p1;
public class PClass {
    public void m1(int a)
    {
    System.out.println("****PClass m1(a)****");
    System.out.println("a:"+a);
    }
```

```
    public void m1(int a,int b)
    {
    this.m1(a);//1_para_method_call
    System.out.println("****PClass m1(a,b)****");
    System.out.println("a:"+a);
    System.out.println("b:"+b);
    }
}
```

p1 : CClass.java

```
package p1;
public class CClass extends PClass
{
    public void m2(int a,int b,int c)
    {
    super.m1(a, b);//2_para_method_call
    System.out.println("*****CClass m2(a,b,c)****");
    System.out.println("a:"+a);
    System.out.println("b:"+b);
    System.out.println("c:"+c);

    }
    public void m2(int a,int b,int c,int d)
    {
    this.m2(a, b, c);//3_para_method_call
    System.out.println("*****CClass m2(a,b,c,d)****");
    System.out.println("a:"+a);
    System.out.println("b:"+b);
    System.out.println("c:"+c);
    System.out.println("d:"+d);

    }
}
```

p2 : DemoInheritance10.java(MainClass)

package p2;

```
import java.util.*;
import p1.*;
public class DemoInheritance10 {
        public static void main(String[] args) {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter the value a:");
      int a = s.nextInt();
      System.out.println("Enter the value b:");
      int b = s.nextInt();
      System.out.println("Enter the value c:");
      int c = s.nextInt();
      System.out.println("Enter the value d:");
      int d = s.nextInt();

      CClass ob = new CClass();
      ob.m2(a, b, c, d);//4_para_method_Call
      s.close();
        }
}
```

o/p:

```
Enter the value a:
11
```

Enter the value b:

12

Enter the value c:

13

Enter the value d:

14

*****PClass m1(a)*****

a:11

*****PClass m1(a,b)*****

a:11

b:12

******CClass m2(a,b,c)*****

a:11

b:12

c:13

******CClass m2(a,b,c,d)*****

a:11

b:12

c:13

d:14

==========

Note:

=>we use "this" and "super" keywords to perform Instance methods interlinking process or Instance methods Chaining process.

==========

Ex-3:(Demonstrating Static method Overloading process)

faq:

Can we perform static method Interlinking process using "this" and "super" keywords?

=>No,we cannot Interlink static methods using "this" and "super" keywords because "this" and "super" keywords are NonStatic NonPrimitive datatype variables.

(static methods cannot access NonStatic variables)

faq:

can we access static methods using "super" and "this" keywords?

=>yes,we can access static methods using "super" and "this" keywords, but these keywords are declared within the Instance methods.

ProjectName : Inheritance_App11

packages,

p1 : PClass.java

```
package p1;
public class PClass {
    public static void m1(int a)
    {
    System.out.println("****PClass m1(a)****");
    System.out.println("a:"+a);
    }
    public static void m1(int a,int b)
    {
    System.out.println("****PClass m1(a,b)****");
    System.out.println("a:"+a);
    System.out.println("b:"+b);
    }
}
```

p1 : CClass.java

```
package p1;
public class CClass extends PClass
{
    public static void m2(int a,int b,int c)
    {
    System.out.println("*****CClass m2(a,b,c)****");
    System.out.println("a:"+a);
    System.out.println("b:"+b);
    System.out.println("c:"+c);

    }
    public static void m2(int a,int b,int c,int d)
    {
    System.out.println("*****CClass m2(a,b,c,d)****");
    System.out.println("a:"+a);
    System.out.println("b:"+b);
    System.out.println("c:"+c);
    System.out.println("d:"+d);

    }
    public void access(int a,int b,int c,int d)
    {
    super.m1(a);
```

```
        super.m1(a, b);
        this.m2(a, b, c);
        this.m2(a, b, c, d);
        }
}
```

p2 : DemoInheritance11.java(MainClass)

```
package p2;

import java.util.*;

import p1.*;

public class DemoInheritance11 {

        public static void main(String[] args) {

      Scanner s = new Scanner(System.in);

      System.out.println("Enter the value a:");

      int a = s.nextInt();

      System.out.println("Enter the value b:");

      int b = s.nextInt();

      System.out.println("Enter the value c:");

      int c = s.nextInt();

      System.out.println("Enter the value d:");

      int d = s.nextInt();


      CClass ob = new CClass();

      ob.access(a, b, c, d);//Instance method_call
```

```
        s.close();
          }
}
```

o/p:

```
Enter the value a:
10
Enter the value b:
20
Enter the value c:
30
Enter the value d:
40
****PClass m1(a)****
a:10
****PClass m1(a,b)****
a:10
b:20
*****CClass m2(a,b,c)****
a:10
b:20
c:30
*****CClass m2(a,b,c,d)****
```

a:10

b:20

c:30

d:40

==========

Types of Inheritances:

=>Inheritances are categorized into the following:

1.Single Inheritance

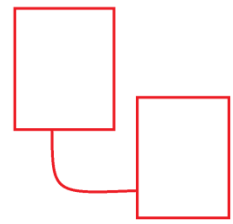
2.Multiple Inheritance

3.Multi-Level Inheritance

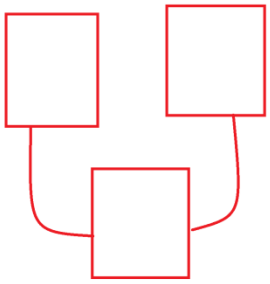
4.Hierarchal Inheritance

5.Hybrid Inheritance

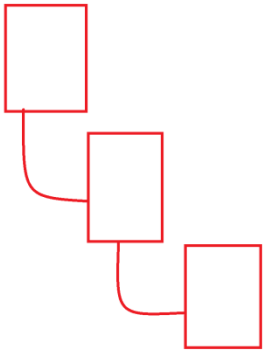
Diagrams:



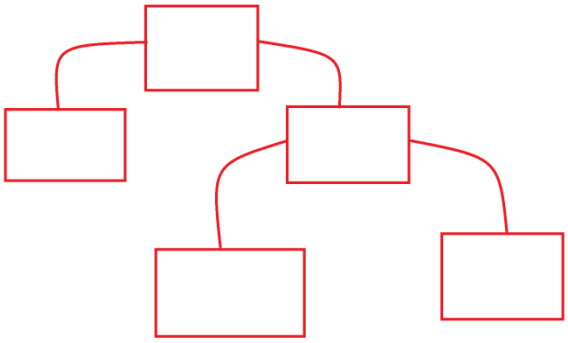
Single Inheritance



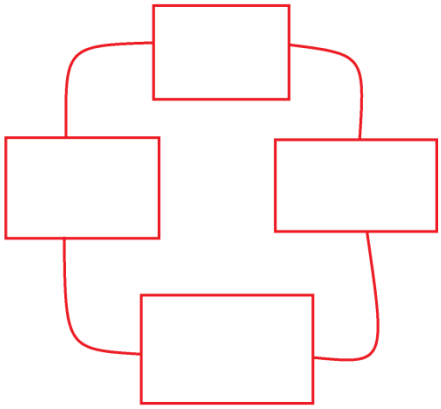
Multiple Inheritance



Multi-Level Inheritance



Hierarchal Inheritance



Hybrid Inheritance

==========