Dt : 10/10/2022(Monday)

faq:

define Method Overloading process?

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

Case-1 : Constructor Overloading process

=>More than one constructor differentiated by their Para_list or Para_type is known as Constructor Overloading process

Ex:

PClass.java

```
package test;
public class PClass {
    public PClass(int x) {
     System.out.println("====PClass(x)====");
     System.out.println("x:"+x);
    }
}
```

Display.java

```
package test;
public class Display extends PClass{
       //Constructor Overloading
    public Display(int x,int y,int z) {
     this(x,y);//Con_with_2_para
     System.out.println("===Display(x,y,z)===");
     System.out.println("z:"+z);
    }
    public Display(int x,int y) {
     super(x);//PClass_Con_with_1_para
     System.out.println("===Display(x,y)====");
     System.out.println("y:"+y);
    }
```

}

DemoInheritance5.java(MainClass)

```
package maccess;
import test.Display;
public class DemoInheritance5 {
	public static void main(String[] args) {
	    Display ob = new Display(11,12,13);//Con_with_3_para
	}
}
```

o/p:

```
====PClass(x)====

x:11

===Display(x,y)====

y:12

===Display(x,y,z)===

z:13
```

---

faq:

wt is the diff b/w

(i)this()

(ii)super()

(i)this():

=>"this()" is used to interlink constructors from the same class for execution

(ii)super():

=>"super()" is used to interlink constructors from PClass and CClass for execution.

===

Case-2 : Instance method Overloading process

=>More than one instance method differentiated by their Para_list or Para_type is known as Instance method Overloading process.

Ex:

PClass.java

```
package test;
public class PClass {
     public int k=200;
     //Method Overloading
     public void m(int a,int b) {
          this.m(a);
           System.out.println("====m(a,b)====");
           System.out.println("b:"+b);
    }
    public void m(int a) {
     System.out.println("====m(a)====");
     System.out.println("a:"+a);
    }
}
```

CClass.java

```
package test;
public class CClass extends PClass{
     public int k=300;
      //Method Overloading
     public void m(int a,int b,int c,int d) {
      this.m(a,b,c);
      System.out.println("====m(a,b,c,d)====");
      System.out.println("d:"+d);
     }
     public void m(int a,int b,int c) {
      super.m(a, b);
```

```
         System.out.println("====m(a,b,c)====");
         System.out.println("c:"+c);
        }
        public void dis()//Non-Overloading method
        {
         System.out.println("====Variables===");
         System.out.println("PClass variable k : "+super.k);
         System.out.println("CClass variable k : "+this.k);

        }
}
```

DemoInheritance6.java(MainClass)

```
package maccess;
import test.*;
public class DemoInheritance6 {
        public static void main(String[] args) {
           CClass ob = new CClass();
           ob.m(11, 12, 13, 14);//method_with_4_para
           ob.dis();
        }
}
```

o/p:

```
====m(a)====

a:11

====m(a,b)====

b:12

====m(a,b,c)====

c:13

====m(a,b,c,d)====

d:14

====Variables===

PClass variable k : 200
```

CClass variable k : 300

--

faq:

wt is the diff b/w

(i)this

(ii)super

(i)this:

=>"this" keyword is used to access variables and methods from the Same class.

(ii)super:

=>"super" keyword is used to access Variables and methods from the Parent class or SuperClass

===

Case-3 : Static method Overloading process

=>More than one static method differentiated by their para_list or para_type is known as Static method Overloading process.

Note:

=>we cannot interlink static methods for execution using "this" and "super" keywords,because "this" and "super" are Non-static pre-defined variables.

=>we can also access static methods using "this" and "super" keywords,but these Keywords must be used in Non-Static methods.

Ex:

PClass.java

```
package test;
public class PClass {
     //Static Method Overloading
     public static void m(int a,int b) {
           System.out.println("====m(a,b)====");
           System.out.println("a:"+a);
           System.out.println("b:"+b);
    }
    public static void m(int a) {
     System.out.println("====m(a)====");
     System.out.println("a:"+a);
    }

}
```

CClass.java

```
package test;
public class CClass extends PClass{
     public int k=300;
      //Static Method Overloading
     public static void m(int a,int b,int c,int d) {
      System.out.println("====m(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 static void m(int a,int b,int c) {
      System.out.println("====m(a,b,c)====");
      System.out.println("a:"+a);
           System.out.println("b:"+b);
      System.out.println("c:"+c);
     }
     public void dis(int a,int b,int c,int d)
     {
          super.m(a);
          super.m(a, b);
          this.m(a, b, c);
          this.m(a, b, c, d);
      }
}
```

DemoInheritance7.java(MainClass)

```
package maccess;
import test.*;
public class DemoInheritance7 {
     public static void main(String[] args) {
        CClass ob = new CClass();
        ob.dis(11, 12, 13, 14);

     }
}
```

o/p:

```
====m(a)====

a:11

====m(a,b)====

a:11

b:12

====m(a,b,c)====

a:11

b:12

c:13

====m(a,b,c,d)====

a:11

b:12

c:13

d:14
```

===

Summary:

(i)Constructor Chaining process is available using "super()" and "this()".

(ii)Instance method Chaining process is available using "super" and "this" keywords

(iii)Static method Chaining process is Not-available using "super" and "this" keywords

==

Dt : 11/10/2022

faq:

Can we perform Overriding process for standard main() method?

=>No,we cannot perform Overriding process for Standard main() method,because main() method is static method.

faq:

Can we perform Overloading process for standard main() method?

=>Yes,we can perform Overloading process for standard main() method.

faq:

Can we pass parameters to Standard main() method?

=>Yes,we can pass parameters to Standard main() while execution Command.

syntax:

java Class_name arg1 arg2 arg3 ...

faq:

define CommandLine argument program?

=>The program in which we pass parameters to Standard main()

method is known as "CommandLine argument program".

Ex : DemoMain.java

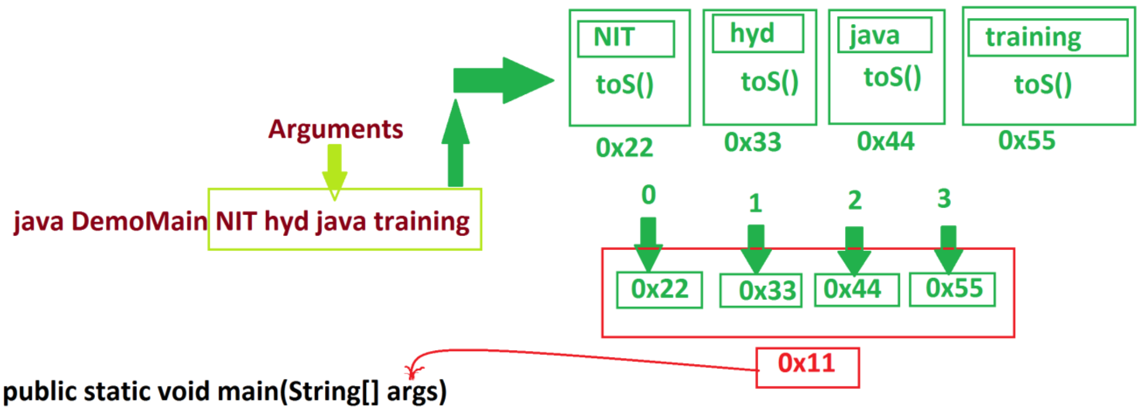
```
package maccess;
public class DemoMain {
      static int p=300;
      public static void main(String[] x)
      {
            DemoMain.main(p);//Method Call
            DemoMain.main(12.34F);//Method Call
          System.out.println("====Standard main()====");
            for(int i=0;i<x.length;i++)
            {
                 System.out.println(x[i].toString());
            }
      }
      public static void main(int k)
      {
          System.out.println("====main(int k)====");
          System.out.println("The value k:"+k);
      }
      public static void main(float z)
      {
          System.out.println("====main(float z)====");
          System.out.println("The value z:"+z);
      }
}
```

o/p:

```
D:\Demo138>javac DemoMain.java

D:\Demo138>java DemoMain NIT hyd java training
```

```
====main(int k)====

The value k:123

====main(float z)====

The value z:12.34

====Standard main()====

NIT

hyd

java

training


D:\Demo138>
```

Diagram:



==

faq:

wt is the diff b/w

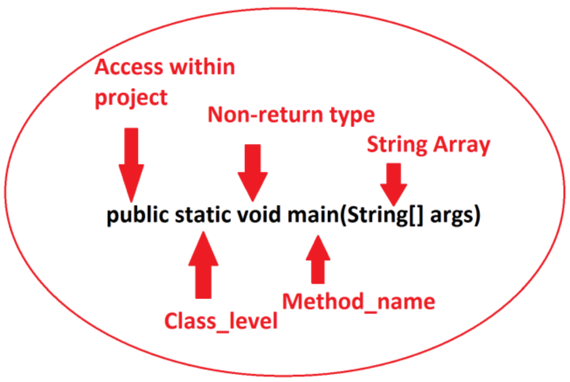
(i)Parameters

(ii)Arguments

=>Parameters specify Variables and Arguments Specify Values.

===

Diagram:



===

**imp*

Types of Inheritances:

=>Inheritances are categorized into the following:

1.Single Inheritance

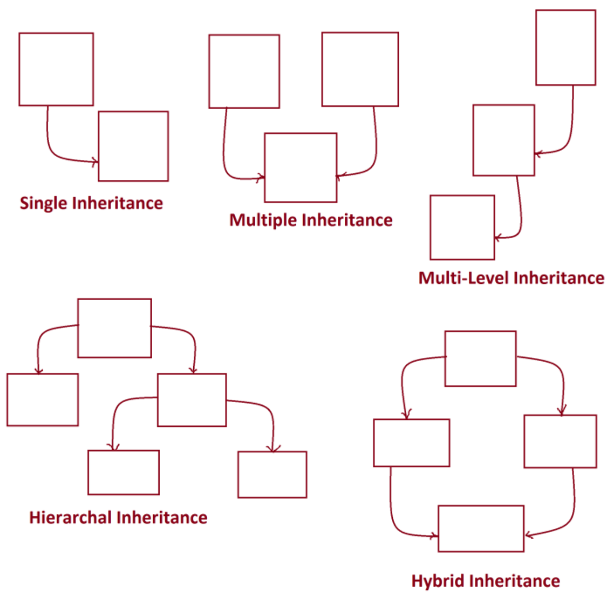
2.Multiple Inheritance

3.Multi-Level Inheritance

4.Hierarchal Inheritance

5.Hybrid Inheritance

Diagrams:



=>In realtime Inheritances are categorized into two types:

(a)Single Inheritance

(b)Multiple Inheritance

(a)Single Inheritance:

=>The process of extracting the features from one class at-a-time is known as Single Inheritance.

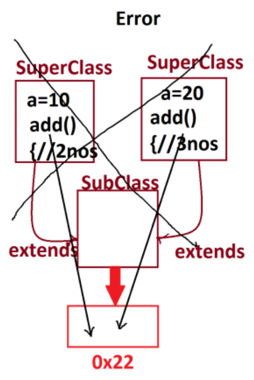
Ex:

above programs

(b)Multiple Inheritance:

=>The process of extracting the features from more than one class at-a-time is known as Multiple Inheritance.

Diagram:



--

Note:

=>Multiple Inheritance process using classes not available in Java,because which leads to replication of programming components and raises ambiguity,the ambiguity state applications will generate Wrong results.

=>We use Interfaces in Java to perform Multiple inheritance process.

==