Dt : 28/8/2023

ProjectName : App_Generalization1

packages,

p1 : PClass.java

```
package p1;
public class PClass {
    public void m1(int x)
    {
         System.out.println("***PClass m1(x)***");
         System.out.println("The value x:"+x);
    }
    public void m2(int y)
    {
         System.out.println("***PClass m2(y)***");
         System.out.println("The value y:"+y);
    }
}
```

p1 : CClass.java

```
package p1;
public class CClass extends PClass
{
    public void m1(int x)//Overriding method
    {
         System.out.println("***CClass m1(x)***");
         System.out.println("The value x:"+x);
    }
    public void m3(int z)//NonOverriding method
    {
         System.out.println("***CClass m3(z)***");
         System.out.println("The value z:"+z);
    }
}
```

p2 : DemoGeneralization1.java(MainClass)

```
package p2;
import p1.*;
public class DemoGeneralization1 {
    public static void main(String[] args) {
        PClass ob = (PClass)new CClass();
        ob.m1(11);
        ob.m2(12);
        //ob.m3(13);
        //CClass ob2 = (CClass)new PClass();
    }
}
```

o/p:

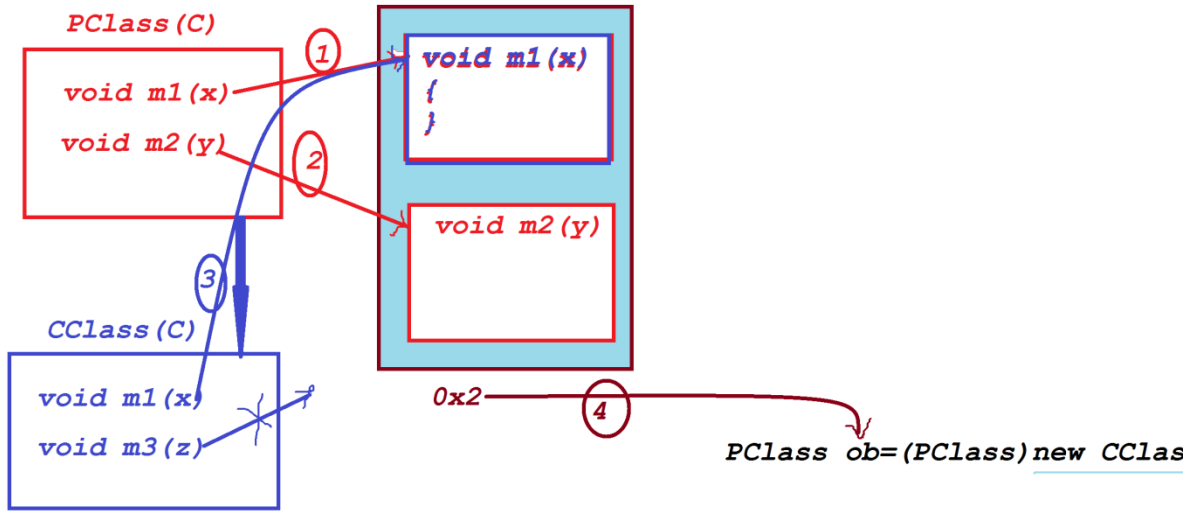
******CClass m1(x)******

The value x:11

******PClass m2(y)******

The value y:12

Diagram:



==========

ProjectName : App_Generalization2

packages,

p1 : ITest.java

```
package p1;
public interface ITest {
  public abstract void m1(int a);
  public default void m2(int b) {
      System.out.println("***ITest-default m2(b)****");
      System.out.println("The value b:"+b);
  }
}
```

p1 : IClass.java

```
package p1;
```

```
public class IClass implements ITest{
   public void m1(int a)//Overriding and implemented
method
    {
         System.out.println("====IClass m1(a)====");
         System.out.println("The value a:"+a);
    }
   public void m3(int c)//NonOverriding and
NonImplemented method
    {
         System.out.println("====IClass m3(c)====");
         System.out.println("The value c:"+c);
    }
}
```

p2 : DemoGeneralization2.java(MainClass)

```
package p2;
import p1.*;
public class DemoGeneralization2 {
     public static void main(String[] args) {
       ITest ob = (ITest)new IClass();
       ob.m1(11);
       ob.m2(12);
       //ob.m3(13);
     }
}
```

o/p:

====IClass m1(a)====

The value a:11

******ITest-default m2(b)*******

The value b:12

==========

Advantage of Generalization process:

=>Through Generalization process we can restrict CClasses and implementation classes to have only Overriding methods.

==========

Note:

=>Generalization process is also known as UpCasting process because Child-Object is Converted into Parent-Object.

=>This UpCasting process is also known Implicit TypeCasting process becuase which is performed automatically,which means manual conversion is not manditory.

==========

faq:

define Specialization process?

=>The process of constructing ChildClass by taking one feature from the ParentClass is known as Specialization process.

=>we use the following syntax to perform Specialization process:

```
CClass ob = (CClass)new PClass();
```

Coding Rule:

=>In Specialization process the ParentClass must be "Pre-defined class" from Java Library,if not it raises "java.lang.ClassCastException".

Note:

(i)we cannot perform Specialization process on Interfaces and AbstractClasses

(ii)Specialization process is also known as Down Casting process,because we convert PClass-Object into CClass-Object

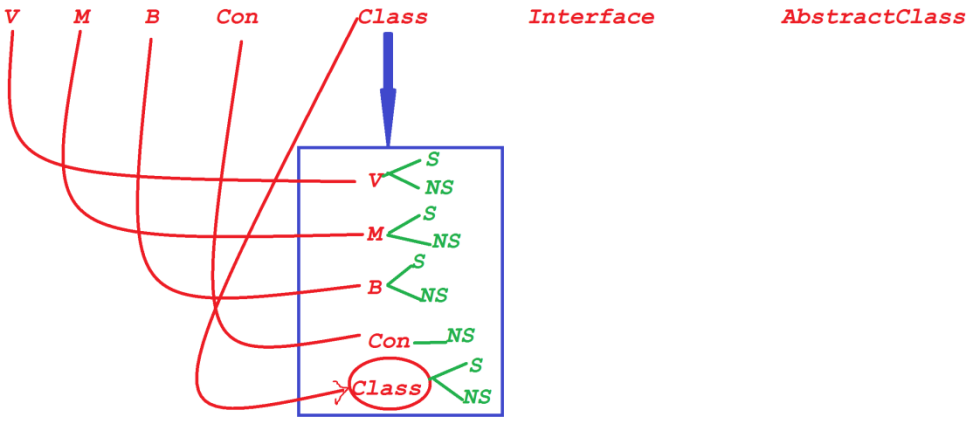
(iii)This DownCasting process is also known as Explicit TypeCasting process,because we have to perform conversion manually.

Ex for Specialization:

Cloning process

Serialization and DeSerialization process

==========



****imp***

InnerClasses in Java:

=>The process of declaring class inside the class is known as InnerClass or Nested Class.

=>These InnerClasses are categorized into two types:

1.Member InnerClasses

2.Anonymous InnerClasses

1.Member InnerClasses:

=>The InnerClasses which are declared as members of Class are known as Member InnerClasses.

=>Member InnerClasses are categorized into two types:

(a)Static member InnerClassrs

(b)NonStatic member InnerClasses

(a)Static member InnerClassrs:

=>The member InnerClasses which are declared with "static" keyword are known as Static member InnerClasses or Class member InnerClasses.

Coding Rules:

(i)Static member InnerClasses can be declared with both static and NonStatic members.

(ii)Instance methods of Static member InnerClasses can access static

variables of OuterClass directly,but cannot access Instance variables of OuterClass directly,because there is no relation b/w OuterClass and InnerClass Objects.

(iii)Static methods of InnerClass can access only static variables of InnerClass and OuterClass directly.

=>we use the following syntax to create object for Static member InnerClasses:

OuterClass_name.InnerClass_name ob = new OuterClass_name.InnerClass_name();

Ex:

ProjectName : App_InnerClass1

packages,

p1 : SubClass1.java

```
package p1;
public class SubClass1
{
    public int a=10;
    public static int b=20;
    public void m1()
    {
        System.out.println("****OuterClass-m1()****");
        System.out.println("The value a:"+a);
        System.out.println("The value b:"+b);
    }//OuterClass method
```

```
    public static class SubClass2
    {
        public int x=100;
        public static int y = 200;
        public void m2()
        {
            System.out.println("****InnerClass Instance
m2()****");
            System.out.println("The value x:"+x);
            System.out.println("The value y:"+y);
            //System.out.println("The value a:"+a);
            System.out.println("The value b:"+b);
        }
        public static void m22()
        {
             System.out.println("****InnerClass static
m22()****");
             //System.out.println("The value x:"+x);
             System.out.println("The value y:"+y);
             //System.out.println("The value a:"+a);
             System.out.println("The value b:"+b);
        }
    }//Static member InnerClass
}//OuterClass
```

p2 : DemoInnerClass1.java(MainClass)

```
package p2;
import p1.*;
public class DemoInnerClass1 {
    public static void main(String[] args) {
        SubClass1 ob1 = new SubClass1();//OuterClass
Object
        ob1.m1();//OuterClass method call
        SubClass1.SubClass2 ob2 = new
SubClass1.SubClass2();
                        //Static member InnerClass
Object
        ob2.m2();//InnerClass Instance method call
        SubClass1.SubClass2.m22();
```

```
        //InnerClass Static method call
    }
}
```

o/p:

*****OuterClass-m1()*****

The value a:10

The value b:20

*****InnerClass Instance m2()*****

The value x:100

The value y:200

The value b:20

*****InnerClass static m22()*****

The value y:200

The value b:20

==========