Dt : 25/8/2023

Ex-program:

ProjectName : AbstractClass_App

packages,

p1 : AClass.java

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

p1 : EClass.java

```
package p1;
public class EClass extends AClass{
   public void m1(int x) {
        System.out.println("====m1(x)====");
        System.out.println("The value x:"+x);
    }
}
```

p2 : DemoAbstractClass.java(MainClass)

```
package p2;
import p1.*;
public class DemoAbstractClass {
     public static void main(String[] args) {
        //AClass ob = new AClass();//Error
         EClass ob = new EClass();
         ob.m1(11);
         ob.m2(12);
     }
```

}

==
===

faq:

wt is the diff b/w

(i)Class

(ii)AbstractClass

=>Classes will hold only Concrete methods,but AbstractClasses can hold both Abstract methods and Concrete methods.

=>Classes can be Instantiated,but AbstractClasses cannot be instantiated.

==

faq:

wt is the diff b/w

(i)Interfaces

(ii)AbstractClasses

=>Components in Interfaces are automatically public,but components in AbstractClasses are automatically "default"

=>Variables in Interfaces are automatically "static" and "final",but variables in AbstractClasses are developer choice.

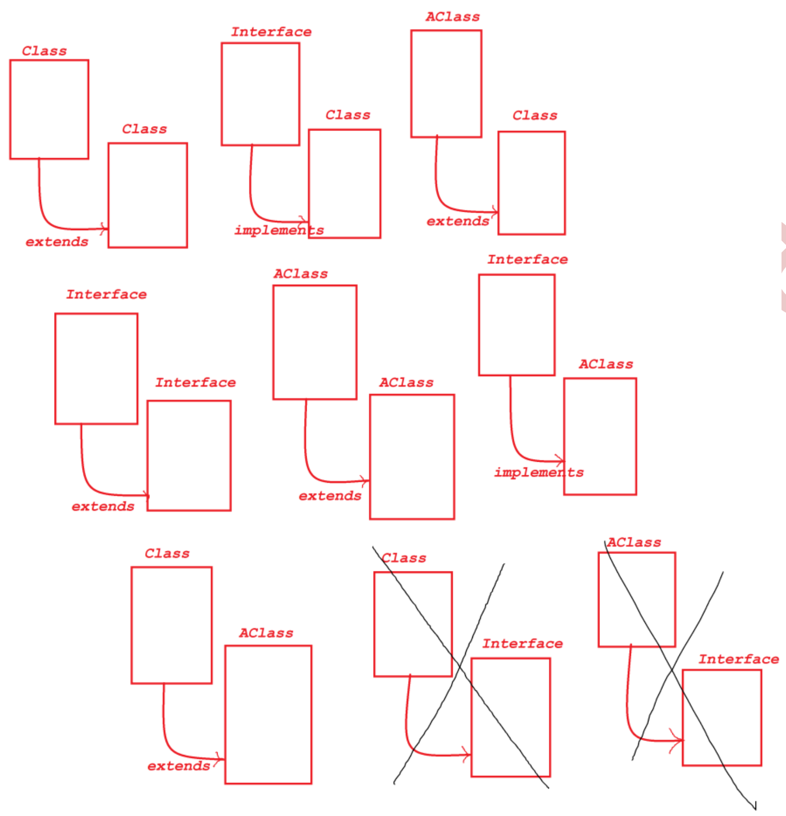
=>Interfaces cannot be declared with "blocks" and "constructors",but

AbstractClasses can be declared with "blocks" and "constructors"

===

****imp***

Summary of Single Inheritance models:



===

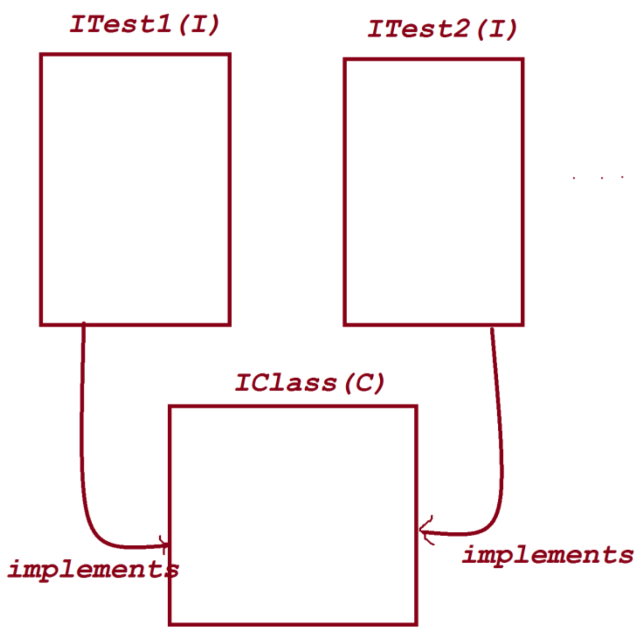
****imp***

Multiple Inheritance Models using Interfaces:

Model-1 : Extracting features from more than one interface into a class

(Class implementing from more than one Interface)

Diagram:



(i)Same abstarct method signatures in Multiple Inheritance process will not create any problem

(ii)Same static concrete method signatures in Multiple Inheritance process will not create any problem

(iii)Same default concrete method signatures in Multiple Inheritance process raises ambigutity at compilation stage

(iv)private Concrete methods of Interfaces will not create any problem

in multiple Inheritance process.

ProjectName : MultipleInheritance_App1

packages,

p1 : ITest1.java

```
package p1;
public interface ITest1 {
   public abstract void m1(int a);
   public static void m2(int b) {
        System.out.println("====ITest1-m2(b)====");
        System.out.println("b:"+b);
   }
   public default void m3(int c) {
        System.out.println("====ITest1-m3(c)====");
        System.out.println("c:"+c);
   }
}
```

p1 : ITest2.java

```
package p1;
public interface ITest2 {
     public abstract void m1(int a);
     public static void m2(int b) {
            System.out.println("====ITest2-m2(b)====");
            System.out.println("b:"+b);
        }
     public default void m33(int c) {
            System.out.println("====ITest2-m33(c)====");
            System.out.println("c:"+c);
        }
}
```

p1 : IClass.java

```
package p1;
public class IClass implements ITest1,ITest2{
   public void m1(int a) {
         System.out.println("====m1(a)===");
         System.out.println("a:"+a);
    }
}
```

p2 : DemoMultipleInheritance1.java(MainClass)

```
package p2;
import p1.*;
public class DemoMultipleInheritance1 {
     public static void main(String[] args) {
          IClass ob = new IClass();
          ob.m1(12);
          ITest1.m2(13);
          ITest2.m2(13);
          ob.m3(14);
          ob.m33(15);
      }
}
```

o/p:

```
====m1(a)===
a:12
====ITest1-m2(b)====
b:13
====ITest2-m2(b)====
b:13
====ITest1-m3(c)====
c:14

====ITest2-m33(c)====

c:15

---------------------------------------------------------------
```