Dt : 24/8/2023

(a)static concrete methods(Java8 - 2014):

=>From Java8 version onwards the interfaces can be declared with static Concrete methods.

=>These static concrete methods will get the memory within the interface while interface loading and can be accessed with Interface_name.

Coding Rule:

=>Static concrete methods of Interface are not available to implementation classes,which means Implementation classes cannot access Static concrete methods of Interfaces.

(b)default concrete methods(Java8 - 2014)

=>From Java8 version omwards the interfaces canbe declared with default concrete methods.

=>default concrete methods means the methods declared with "default" keyword and which are NonStatic methods.

Coding Rule:

=>default concrete methods are available to implementation classes,which means default concrete methods can be accessed with implementation object name.

ProjectName : Interface_App4

packages,

p1 : ITest.java

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

p1 : IClass.java

```
package p1;
public class IClass implements ITest{
    public void m1(int a) {
         System.out.println("****Implemented m1(a)****");
         System.out.println("The value a:"+a);
    }
}
```

p2 : DemoInterface4.java(MainClass)

```
package p2;
import p1.*;
public class DemoInterface4 {
     public static void main(String[] args) {
        ITest.m2(123);//Static method call
        IClass ob = new IClass();//Implementation Object
```

```
        ob.m1(124);//Implemented method call
        //IClass.m2(125);//Static method Call
        ob.m3(126);//default method call
    }
}
```

o/p:

*******static m2(b)*******

The value b:123

*******Implemented m1(a)*******

The value a:124

*******default m3(c)*******

The value c:126

Note:

=>Static Concrete methods will be at Interface level and default concrete methods are at implementation Object level.

=>Static concrete methods are used in multiple inheritance applications and default concrete methods are used in NonMultiple inheritance applications.

(c)private concrete methods(Java9 - 2017)

=>From Java9 version onwards the interfaces can be declared with private concrete methods.

=>private concrete methods means the concrete methods which are declared with private keyword and which are available in two types:

(i)static private concrete methods

(ii)NonStatic private concrete methods

Coding Rule:

=>private concrete methods are accessed only inside the interface,which means accessed by the NonPrivate concrete methods of Interface.

ProjectName : Interface_App5

packages,

p1 : ITest.java

```
package p1;
public interface ITest {
    public abstract void m1(int a);
    private static void m2(int b) {
         System.out.println("****static private m2(b)****");
         System.out.println("The value b:"+b);
    }
    private void m3(int c) {
         System.out.println("****NonStatic private m3(c)****");
         System.out.println("The value c:"+c);
    }
    public default void access(int b,int c) {
         ITest.m2(b);
         this.m3(c);
    }
}
```

p1 : IClass.java

```
package p1;
public class IClass implements ITest{
  public void m1(int a) {
       System.out.println("****Implemented m1(a)****");
       System.out.println("The value a:"+a);
  }
}
```

p2 : DemoInterface5.java(MainClass)

```
package p2;
import p1.*;
public class DemoInterface5 {
    public static void main(String[] args) {
      IClass ob = new IClass();
      ob.m1(11);
      ob.access(13, 14);
    }
}
```

o/p:

*******Implemented m1(a)*******

The value a:11

*******static private m2(b)*******

The value b:13

*******NonStatic private m3(c)*******

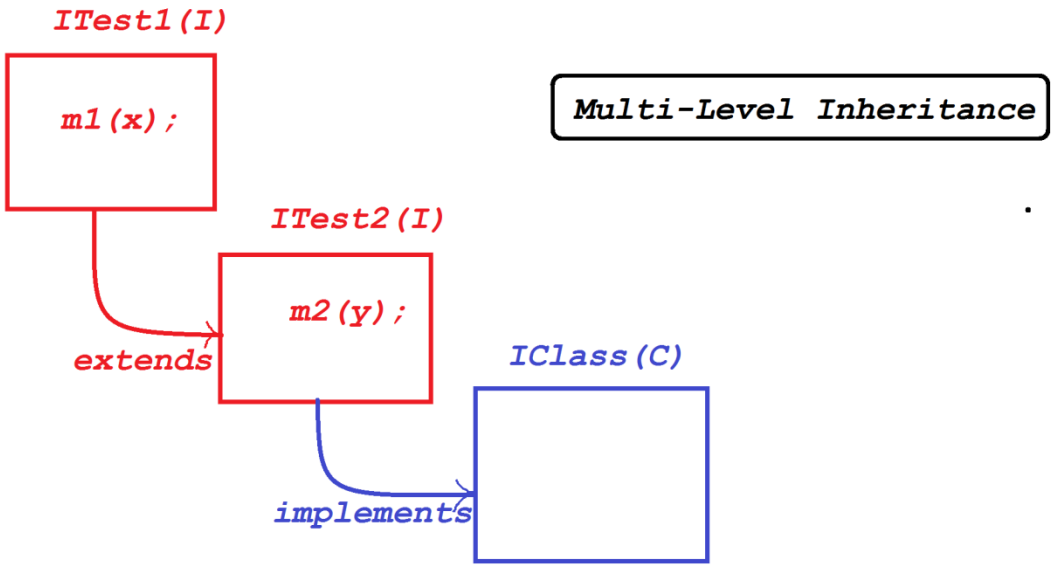
The value c:14

==========

Rule-12 : Interfaces cannot be declared with Blocks and Constructors

Rule-13 : Interface can take the features from another interface using "extends" keyword.

Diagram:



ProjectName : Interface_App6

packages,

p1 : ITest1.java

```
package p1;
public interface ITest1 {
  public abstract void m1(int a);
}
```

p1 : ITest2.java

```
package p1;
public interface ITest2 extends ITest1{
   public abstract void m2(int b);
}
```

p1 : IClass.java

```
package p1;
public class IClass implements ITest2{
   public void m1(int a) {
        System.out.println("===m1(a)===");
        System.out.println("The value a:"+a);
   }
   public void m2(int b) {
        System.out.println("===m2(b)===");
        System.out.println("The value b:"+b);
   }
}
```

p2 : DemoInterface6.java(MainClass)

```
package p2;
import p1.*;
public class DemoInterface6 {
    public static void main(String[] args) {
      IClass ob = new IClass();
      ob.m1(11);
      ob.m2(12);
   }
}
```

o/p:

===m1(a)===

The value a:11

===m2(b)===

The value b:12

==========

**imp*

Abstract Classes in Java:

=>The classes which are declared with "abstract" keywork are known as "abstract classes"

=>Abstract Classes are collection of Variables,concrete methods,abstract methods,Blocks and Constructors

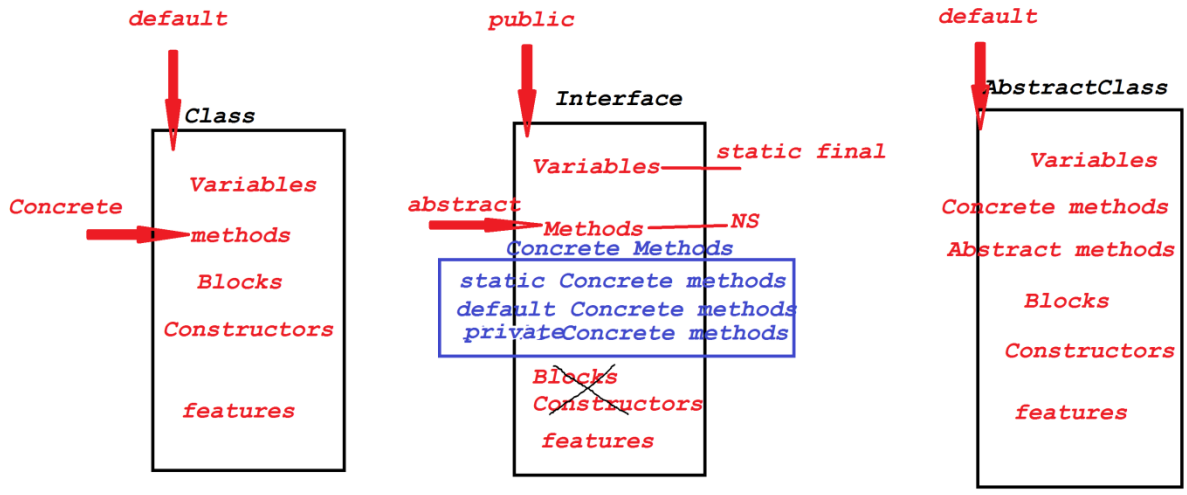
=>Abstract methods in abstract classes must be declared with "abstract" keyword.

=>Abstract classes cannot be instantiated,which means we cannot create object for abstract classes

=>These abstract classes are extended to classes using "extends" keyword and the classes are known as extention classes or implementation classes.

=>These extention classes must construct body for all abstract methods of abstract classes.

Diagram:



==========

====