Dt : 27/10/2022

****imp***

Method References in Java:(Java8 - new feature)

=>The process in which abstract method of functional Interface is attached with the method_body from a class,where the class is not related to functional interface is known as "Method Reference Concept".

=>These method references are categorized into three types:

(a)Reference to Constructor

(b)Reference to Instance method

(c)Reference to Static method

(a)Reference to Constructor:

=>The process in which abstract method of functional interface is attached with the Constructor_body is known as "Reference to Constructor".

syntax:

```
Func_interface_name obj = Class_name :: new;
```

Ex:

```
ITest ob1 = Display :: new;
```

(b)Reference to Instance method:

=>The process in which abstract method of functional interface is attached with the Instance_method_body is known as "Reference to Instance

method".

syntax:

Func_interface_name obj = Object_name :: Instance_method_name;

Ex:

ITest ob2 = d :: dis1;

(c)Reference to Static method:

=>The process in which abstract method of functional interface is attached with the Static_method_body is known as "Reference to Static method".

syntax:

Func_interface_name obj = Class_name :: Static_method_name;

Ex:

ITest ob3 = Display :: dis2;

Ex-program:

ITest.java

```
package test;
public interface ITest {
   public abstract void m(int k);
}
```

Display.java

```java
package test;
public class Display {
    public Display(int x) {
     System.out.println("====Constructor body====");
     System.out.println("The value x:"+x);
    }
    public void dis1(int y) {
     System.out.println("====Instance method body====");
     System.out.println("The value y:"+y);

    }
    public static void dis2(int z) {
     System.out.println("====Static method body====");
     System.out.println("The value z:"+z);

    }
}
```

DemoMethodReferences.java(MainClass)

```java
package maccess;
import test.*;
public class DemoMethodReferences {
     public static void main(String[] args) {
          ITest ob1 = Display :: new;//Reference to Constructor
          ob1.m(121);//Constructor_body executed

          Display d = new Display(100);//Con_Call
          ITest ob2 = d :: dis1;//Reference to Instance method
          ob2.m(122);//Instance method_body executed

          ITest ob3 = Display :: dis2;//Reference to Static
method
          ob3.m(123);//Static method_body executed

     }
}
```

o/p:

```
====Constructor body====

The value x:121
```

====Constructor body====

The value x:100

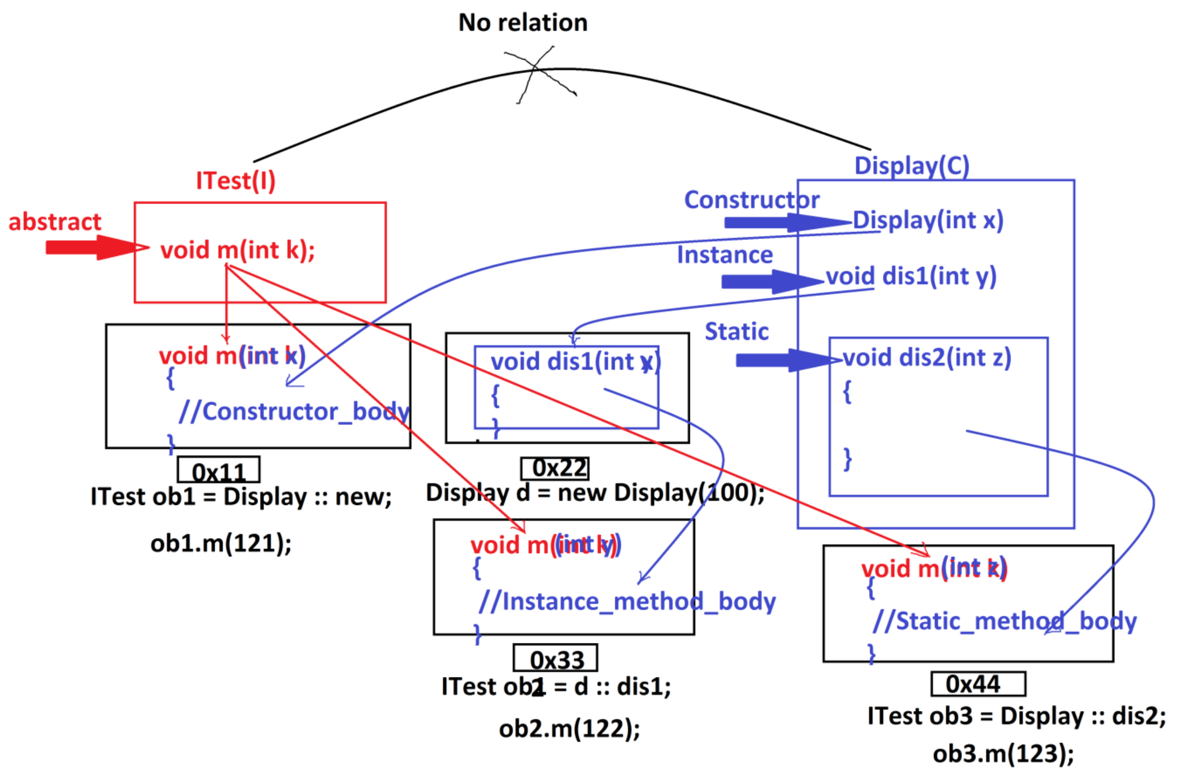
====Instance method body====

The value y:122

====Static method body====

The value z:123

Diagram:



===

**imp*

InnerClasses in Interfaces:

=>we can also declare InnerClasses in Interfaces and which are

automatically "Static member InnerClasses".

****imp***

InnerClasses in AbstractClasses:

=>we can also declare InnerClasses in AbstractClasses and which can be Static member InnerClasses or NonStatic member InnerClasses.

Ex:

ITest.java

```
package test;
public class ITest {
    public static class SubClass2{
     public void m2(int x) {
          System.out.println("===m2(x)===");
          System.out.println("The value x:"+x);
     }
    }//Static member InnerClass
}//OuterInterface
```

AClass.java

```
package test;
public abstract class AClass {
    public class SubClass3{
     public void m3(int p) {
          System.out.println("===m3(p)===");
          System.out.println("The value p:"+p);
     }
    }//Instance member InnerClass
    public static class SubClass33{
     public void m33(int q) {
          System.out.println("===m33(q)===");
          System.out.println("The value q:"+q);
     }
    }//Static member InnerClass
}//OuterAbstractClass
```

DemoInnerClasses4.java(MainClass)

```
package maccess;
import test.*;
public class DemoInnerClasses4 {
     public static void main(String[] args) {
        System.out.println("****InnerClass in Interface****");
        ITest.SubClass2 ob2 = new ITest.SubClass2();
        ob2.m2(11);
        System.out.println("****InnerClass in 
AbstractClass****");
        AClass ob = new AClass()
          {
          //Anonymous_InnerClass_with_0_members
        };
        AClass.SubClass3 ob3 = ob.new SubClass3();
        ob3.m3(12);
        AClass.SubClass33 ob33 = new AClass.SubClass33();
        ob33.m33(13);
     }
}
```

o/p:

```
****InnerClass in Interface****

===m2(x)===

The value x:11

****InnerClass in AbstractClass****

===m3(p)===

The value p:12

===m33(q)===

The value q:13
```

===