Dt : 5/9/2023

**imp*

Method References in Java:(Java8 - new feature)

=>The process in which body-of-method from a class,is attached with the abstract method of Functional Interface is known as Method Reference and in this process the class is not related to Functional Interface.

=>Method references are categorized into three types:

1.Reference to Constructor

2.Reference to Instance Method

3.Reference to Static Method

1.Reference to Constructor:

=>The process in which the abstract method of Functional Interface is attached with the body of Constructor,is known as Reference to Constructor.

syntax:

```
Func_Interface_name ob = Class_name :: new;
```

Ex:

```
IDisplay ob1 = SubClass :: new;
```

2.Reference to Instance Method:

=>The process in which the abstract method of Functional Interface is attached with the body of Instance method,is known as Reference to Instance method.

syntax:

Func_Interface_name ob = obj_name :: Instance_method_name;

Ex:

IDisplay ob2 = sb :: m1;

3.Reference to Static Method:

=>The process in which the abstract method of Functional Interface is attached with the body of Static method,is known as Reference to Static method

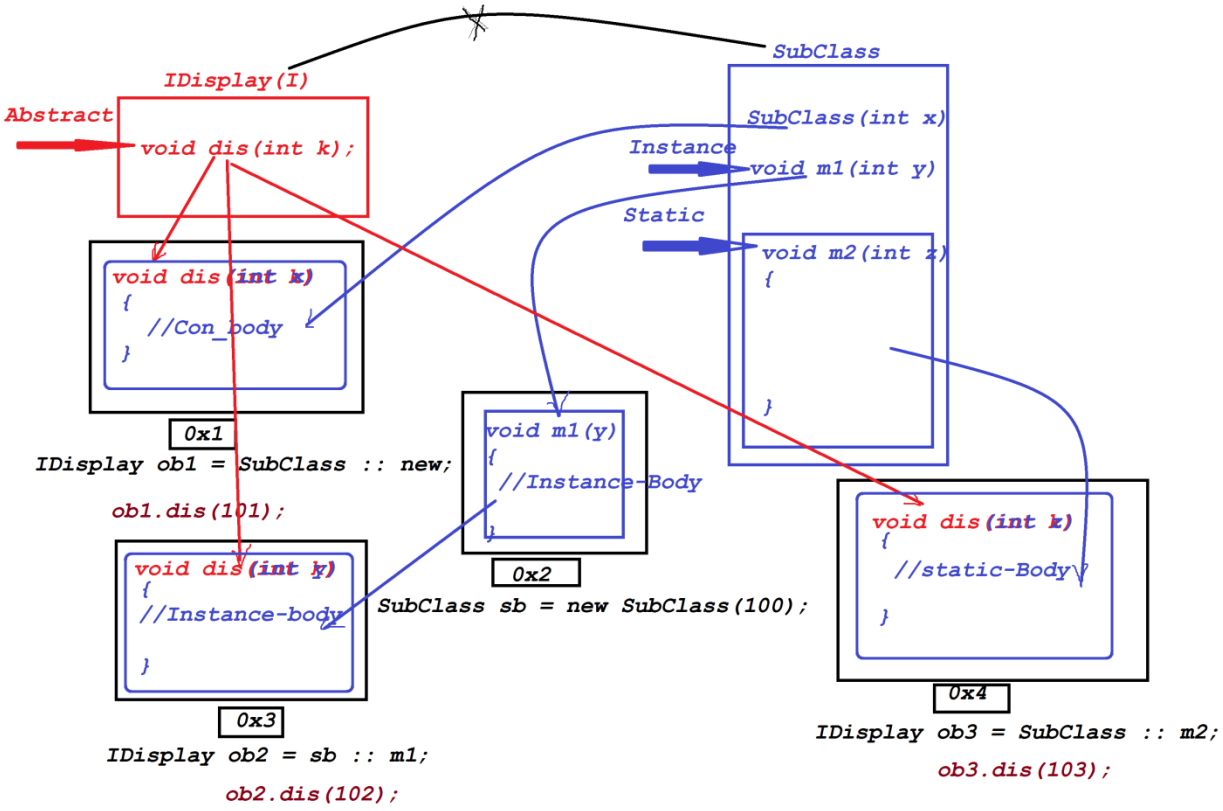
syntax:

Func_Interface_name ob = Class_name :: static_method_name;

Ex:

IDisplay ob3 = SubClass :: m2;

Layout:



Ex:

ProjectName : App_MethodReferences

packages,

p1 : IDisplay.java

```
package p1;
public interface IDisplay {
  public abstract void dis(int k);
}
```

p1 : SubClass.java

```
package p1;
```

```
public class SubClass {
    public SubClass(int x) {
         System.out.println("====Constructor-body====");
         System.out.println("The value x:"+x);
    }
    public void m1(int y) {
         System.out.println("====Instance method-
body====");
         System.out.println("The value y:"+y);
    }
    public static void m2(int z) {
         System.out.println("====Static method-
body====");
         System.out.println("The value z:"+z);
    }
}
```

p2 : DemoMethodReferences.java(MainClass)

```
package p2;
import p1.*;
public class DemoMethodReferences {
     public static void main(String[] args) {
       IDisplay ob1 = SubClass :: new;//Reference to
Constructor
       ob1.dis(101);
       SubClass sb = new SubClass(100);//Con_call
       IDisplay ob2 = sb :: m1;//Reference to Instance
method
       ob2.dis(102);
       IDisplay ob3 = SubClass :: m2;//Reference to
Static method
       ob3.dis(103);
     }
}
```

o/p:

====Constructor-body====

The value x:101

====Constructor-body====

The value x:100

====Instance method-body====

The value y:102

====Static method-body====

The value z:103

==========

****imp***

InnerClasses in Interfaces:

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

automatically static member InnerClasses.

****imp***

InnerClasses in AbstractCasses:

=>we can also declare InnerClasses in AbstractClasses and which canbe

Static member InnerClasses or NonStatic member InnerClasses.

Ex:

ProjectName : App_InnerClass4

packages,

p1 : ITest.java

```
package p1;
public interface ITest {
    public static class SubClass2{
        public void m2(int a) {
            System.out.println("****m2(a)****");
            System.out.println("The value a:"+a);
        }
    }//Static member InnerClass
}//OuterInterface
```

p1 : AClass.java

```
package p1;
public abstract class AClass {
    public class SubClass3{
        public void m3(int b) {
            System.out.println("****m3(b)****");
            System.out.println("The value b:"+b);
        }
    }//Instance member InnerClass
    public static class SubClass4{
        public void m4(int c) {
            System.out.println("****m4(c)****");
            System.out.println("The value c:"+c);
        }
    }//Static member InnerClass
}//OuterAbstractClass
```

p2 : DemoInnerClass4.java(MainClass)

```
package p2;
import p1.*;
public class DemoInnerClass4 {
     public static void main(String[] args) {
        System.out.println("-----InnerClass in
Interface-----");
        ITest.SubClass2 ob2 = new ITest.SubClass2();
        ob2.m2(12);
```

```
        System.out.println("----InnerClass in
AbstractClass---");
        AClass ob = new AClass()
        {
            //Anonymous_Class_body_with_Zero_methods
        };
        AClass.SubClass3 ob3 = ob.new SubClass3();
        ob3.m3(13);
        AClass.SubClass4 ob4 = new AClass.SubClass4();
        ob4.m4(14);
    }
}
```

o/p:

-----InnerClass in Interface-----

*****m2(a)*****

The value a:12

----InnerClass in AbstractClass---

*****m3(b)*****

The value b:13

*****m4(c)*****

The value c:14

-==