Dt : 28/10/2022

***imp**

InnerInterfaces in Java:

(i)InnerInterfaces in Classes:

=>we can also declare InnerInterfaces in Classes and which can be Static member InnerInterfaces or NonStatic member InnerInterfaces.

(ii)InnerInterfaces in Interfaces:

=>we can also declare InnerInterfaces in Interfaces and which are automatically Static member InnerInterfaces.

(iii)InnerInterfaces in AbstractClasses:

=>we can also declare InnerInterfaces in AbstractClasses and which can static member InnerInterfaces or NonStatic member InnerInterfaces.

Ex:

SubClass.java

```
package test;
public class SubClass {
   public interface ITest2{
        public abstract void m2(int x);
   }//Instance member InnerInterface
   public static interface ITest22{
        public abstract void m22(int y);
   }//Static member InnerInterface
}//OuterClass
```

ITest.java

```
package test;
```

```
public interface ITest {
   public static interface ITest3{
        public abstract void m3(int a);
   }//Static member InnerInterface
}//OuterInterface
```

AClass.java

```
package test;
public abstract class AClass {
    public interface ITest4{
     public abstract void m4(int p);
    }//Instance member InnerInterface
    public static interface ITest44{
     public abstract void m44(int q);
    }//Static member InnerInterface
}//OuterAbstractClass
```

DemoInnerInterface.java(MainClass)

```
package maccess;
import test.*;
public class DemoInnerInterface {
     public static void main(String[] args) {
      System.out.println("****InnerInterface in Class****");
      SubClass.ITest2 ob2 = (int x)->
      {
     System.out.println("====m2(x)===");
     System.out.println("The value x:"+x);
      };
      ob2.m2(12);
      SubClass.ITest22 ob22 = (int y)->
      {
       System.out.println("====m22(y)===");
          System.out.println("The value y:"+y);
      };
      ob22.m22(13);
      System.out.println("****InnerInterface in Interface****");
      ITest.ITest3 ob3 = (int a)->
      {
       System.out.println("====m3(a)===");
        System.out.println("The value a:"+a);
      };
      ob3.m3(14);
```

```
        System.out.println("****InnerInterface in 
AbstractClass****");
        AClass.ITest4 ob4 = (int p)->
        {
         System.out.println("====m4(p)===");
            System.out.println("The value p:"+p);
        };
        ob4.m4(15);
        AClass.ITest44 ob44 = (int q)->
        {
         System.out.println("====m44(q)===");
            System.out.println("The value q:"+q);
        };
        ob44.m44(16);
       }
}
```

o/p:

```
****InnerInterface in Class****

====m2(x)===

The value x:12

====m22(y)===

The value y:13

****InnerInterface in Interface****

====m3(a)===

The value a:14

****InnerInterface in AbstractClass****

====m4(p)===

The value p:15

====m44(q)===

The value q:16
```

==

*Imp

InnerAbstractClasses in Java:

(i)InnerAbstractClasses in Class:

=>we can also declare InnerAbstractClasses in Class and which can be Static member InnerAbstractClass or NonStatic member InnerAbstractClass.

(ii)InnerAbstractClasses in Interfaces:

=>we can also declare InnerAbstractClasses in Interfaces and which are automatically Static member InnerAbstractClasses.

(iii)InnerAbstractClasses in AbstractClasses:

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

Ex:(Assignment)

===

Summary of Programming Components:

(a)Variables

1.Primitive DataType variables(hold Values)

(i)Static(Outside methods)

(ii)NonStatic

=>Instance(Outside methods)

=>Local(Inside methods)

2.NonPrimitive DataType variable(object references)

(i)Static

(ii)NonStatic

=>Instance

=>Local

(b)Methods

1.static methods

(i)Pre-defined

(ii)User defined

2.NonStatic methods(Instance methods)

(i)Pre-defined

(ii)User defined

(c)Blocks

1.Static blocks

2.NonStatic blocks(Instance blocks)

(d)Constructors

=>Non-Static Constructor

(e)Classes

1.Static Classes(Only InnerClasses)

2.NonStatic Classes

(f)Interfaces

1.Static Interfaces(Only InnerInterfaces)

2.NonStatic Interfaces

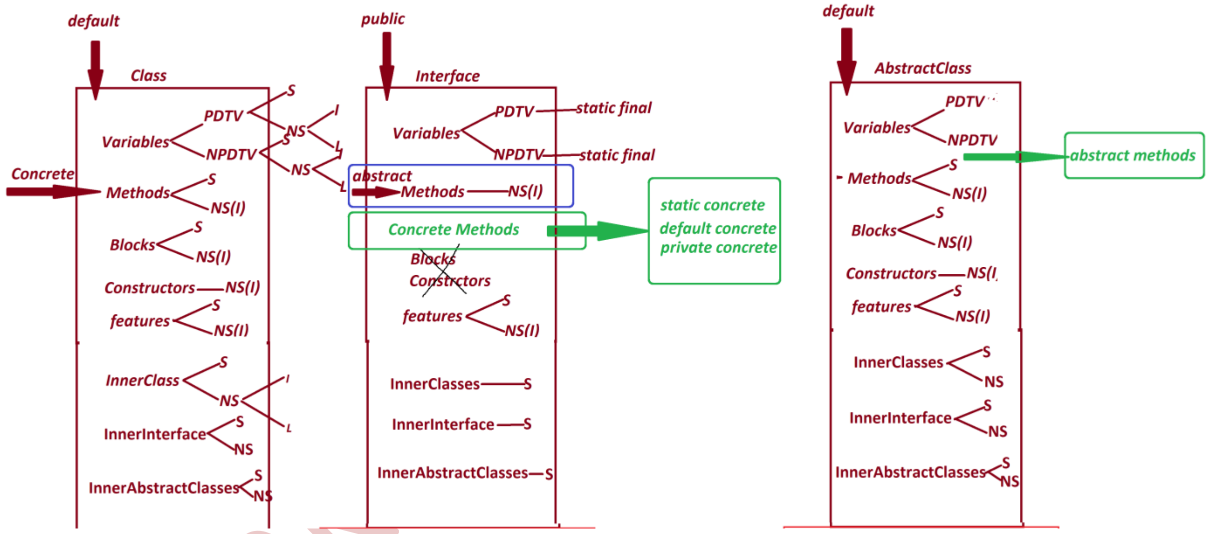
(g)AbstractClasses

1.Static AbstractClasses(Only InnerAbstractClasses)

2.NonStatic AbstractClasses

==

Comparision Diagram:



==

****imp***

Exception Handling process:

define Exception?

=>The disturbance which is occured from the application is known as

"exception"

define Exception Handling process?

=>The process which is used to handle the exception is known as Exception Handling Process.

=>we use the following blocks in Exception Handling process:

1.try block

2.catch block

3.finally block

=>These blocks are executed automatically when the exception is raised.

1.try block:

=>try block will hold the statements which are going to raise the exception.

syntax:

```
try
{
 //statements
}
```

behaviour of try block:

=>when exception raised from try block then one object is created for

Exception_type_class and object reference is thrown onto catch block.

2.catch block:

=>catch block will hold object reference thrown from the try block and the required msg generated from catch block.

syntax:

```
catch(Exception_type_class ref_var)
{
 //msg
}
```

3.finally block:

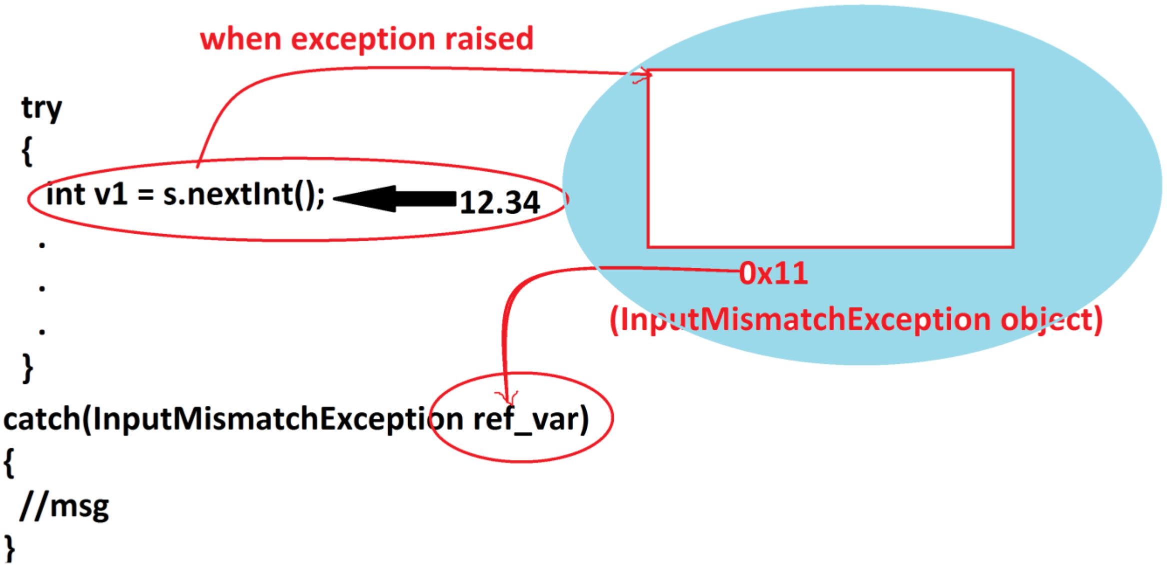
=>finally block is part of exception handling process,but executed automatically without depending on exception.

=>In realtime finally block will hold resource closing operations like IO close,File close,DB close,...

syntax:

```
finally
{
  //statements
}
```

Diagram:



```
 try
 {
  int v1 = s.nextInt();
  .
  .
  .
 }
catch(InputMismatchException ref_var)
{
 //msg
}
```

==