Dt : 4/11/2022

****imp***

Handling Multiple Exceptons:

=>we use the following two ways to handle Multiple Exceptions:

(i)we declare multiple catch blocks to a try-block to handle Multiple Exceptions

syntax:

```
try
{
 //Exception1
 //Exception2
}
catch(Exception1 ob)
{
 //msg
}
catch(Exception2 ob)
{
 //msg
}
```

(ii)From Java7 version onwards we can use single catch block to hold

Multiple exceptions.

```
try
{
 //Exception1
 //Exception2
}
catch(Exception1 | Exception2 | ... ob)
{
 //msg
}
```

==========

faq:

define try-with-resource statement?

=>try-with-resource statement is introduced by Java7 version and in which we can declare resources with try.

syntax:

```
try(resource1;resource2;...)
{
 //statements
}
```

Ex:

try(Scanner s = new Scanner(System.in);)

{

//statements

}

Advantage:

=>The resources will be closed automatically in try-with-resource statement,which means no need to use finally block.

=>In try-with-resource statement catch is optional block.

==========

faq:

define Enhanced try-with-resource statement?

=>Enhanced try-with-resource statement introduced by Java9 version and in which we declare resources outside the try and resource-reference variables with try

syntax:

resource1;resource2;...

try(res1-var;res2-var;...)

{

//statements

}

Ex:

```
Scanner s = new Scanner(System.in);
try(s;)
{
  //statements
}
```

==========

Note:

=>The classes which are declared in "try-with-resource statement" must be implementations of "java.lang.AutoCloseable" interface.

==========

faq:

define "java.lang.NullPointerException"?

=>"java.lang.NullPointerException" is raised when we use NonPrimitive datatype variable assigned with null value.

Ex:

```
package maccess;
public class DemoException5 {
   public static String str = null;
        public static void main(String[] args) {
      int len = str.length();//NullPointerException is raised
      System.out.println("str:"+str.toString());
      System.out.println("len of str:"+len);
        }
```

```
}
```

o/p:

Exception in thread "main" java.lang.NullPointerException:

Cannot invoke "String.length()" because "maccess.DemoException5.str"

is null at maccess.DemoException5.main(DemoException5.java:5)

==========

Dt : 5/11/2022

Assignment-1:

Convert BankTransaction application with Anonymous classes into

Exception handling process.

Balance.java

```
package test;
public class Balance {
   public double bal=2000;
   public double getBalance() {
        return bal;
   }
}
```

CheckPinNo.java

```
package test;
public class CheckPinNo {
    public boolean verify(int pinNo) {
     return switch(pinNo) {
     case 1111 : yield true;
     case 2222 : yield true;
     case 3333 : yield true;
     default : yield false;
     };
    }
}
```

Transaction.java

```
package test;
public interface Transaction {
   public static final Balance b = new Balance();
   public abstract void process(int amt)throws Exception;
}
```

DemoException6.java(MainClass)

```
package maccess;

import java.util.*;

import test.*;

@SuppressWarnings("serial")

public class DemoException6 extends Exception

{

        public DemoException6(String msg)

        {

                super(msg);

        }

        public static void main(String[] args)

        {

                Scanner s = new Scanner(System.in);

      try(s;)//Java9

      {

      int count=0;

      xyz:

          while(true)
```

```
{
        try
        {
                System.out.println("Enter the pinNo:");
                int pinNo = s.nextInt();
                CheckPinNo cpn = new CheckPinNo();
                boolean k = cpn.verify(pinNo);
                if(!k)//Exception Condition
                {
                        DemoException6 de = new DemoException6("Invalid pinNo");
                        throw de;
                }
                System.out.println("====Choice====");
                System.out.println("1.WithDraw\n2.Deposit");
                System.out.println("Enter the choice:");
                switch(s.nextInt())
                {
                case 1:
                        System.out.println("Enter the amt:");
                        int a1 = s.nextInt();
                        if(!(a1>0 && a1%100==0))//Exception Condition
                        {
                                DemoException6 de = new DemoException6("Invalid
amt");
```

```
                throw de;
            }
            Transaction wd2 = new Transaction()
            {
              public void process(int amt)throws Exception
              {
                    try
                    {
                          if(amt>b.bal)//Exception condition
                          {
                                Exception wd = new Exception("Insufficient
fund");
                                throw wd;
                          }
                          System.out.println("Amt withDrawn:"+amt);
                          b.bal=b.bal-amt;
                          System.out.println("Balance
amt:"+b.getBalance());
                          System.out.println("Transaction Completed...");
                    }//end of try
                    catch(Exception wd)
                    {
                          throw wd;//re-throwing
                    }
```

```
            }
        };
        wd2.process(a1);//method Call
        break xyz;
    case 2:
        System.out.println("Enter the amt:");
        int a2 = s.nextInt();
        if(!(a2>0 && a2%100==0))//Exception Condition
        {
            DemoException6 de = new DemoException6("Invalid
amt");
            throw de;
        }
        Transaction dp = new Transaction()
        {
            public void process(int amt)
            {
                System.out.println("Amt deposited:"+amt);
                b.bal=b.bal+amt;
                System.out.println("Balance
amt:"+b.getBalance());
                System.out.println("Transaction completed...");
            }
        };
        dp.process(a2);
```

```
                break xyz;
            default:
                System.out.println("Invalid Choice...");
                break xyz;
            }//end of switch
        }//end of try
        catch(InputMismatchException ime)
        {
            System.out.println("Enter only Integer value...");
            break xyz;
        }
        catch(Exception de)
        {
            System.out.println(de.getMessage());
            if(de.getMessage().equals("Invalid pinNo"))
            {
                count++;
            if(count==3)//Nested Simple if
            {
                System.out.println("Transaction blocked...");
                break xyz;
            }
            }//end of if
            else
```

```
                    {
                        break xyz;
                    }
                }


        }//end of loop
    }//end of try-with-resource
        }
}
```

Note:

=>In the process of handling exception in Anonymous classes,we handle "java.lang.Exception" directly because the class name is not available.

==========

Assignment-2:

Convert BankTransaction application with LambdaExpressions into Exception handling process.

Balance.java

```
package test;
public class Balance {
   public double bal=2000;
   public double getBalance() {
        return bal;
   }
}
```

CheckPinNo.java

```
package test;
```

```
public class CheckPinNo {
    public boolean verify(int pinNo) {
     return switch(pinNo) {
     case 1111 : yield true;
     case 2222 : yield true;
     case 3333 : yield true;
     default : yield false;
     };
    }
}
```

Transaction.java

```
package test;
public interface Transaction {
   public static final Balance b = new Balance();
   public abstract void process(int amt)throws Exception;
}
```

DemoException7.java(MainClass)

```
package maccess;

import java.util.*;

import test.*;

public class DemoException7 extends Exception

{

        public DemoException7(String msg)

        {

                super(msg);

        }

        public static void main(String[] args)

        {

                Scanner s = new Scanner(System.in);
```

```
try(s;)//Java9
{
int count=0;
xyz:
   while(true)
   {
         try
         {
               System.out.println("Enter the pinNo:");
               int pinNo = s.nextInt();
               CheckPinNo cpn = new CheckPinNo();
               boolean k = cpn.verify(pinNo);
               if(!k)//Exception Condition
               {
                     DemoException7 de = new DemoException7("Invalid pinNo");
                     throw de;
               }
               System.out.println("====Choice====");
               System.out.println("1.WithDraw\n2.Deposit");
               System.out.println("Enter the choice:");
               switch(s.nextInt())
               {
               case 1:
                     System.out.println("Enter the amt:");
```

```
int a1 = s.nextInt();
if(!(a1>0 && a1%100==0))//Exception Condition
{
        DemoException7 de = new DemoException7("Invalid
amt");
        throw de;
}
Transaction wd2 = (int amt)->
  {
        try
        {
                if(amt>Transaction.b.bal)//Exception condition
                {
                        Exception wd = new Exception("Insufficient
fund");

                        throw wd;
                }
                System.out.println("Amt withDrawn:"+amt);
                Transaction.b.bal=Transaction.b.bal-amt;
                System.out.println("Balance
amt:"+Transaction.b.getBalance());
                System.out.println("Transaction Completed...");
        }//end of try
        catch(Exception wd)
```

```
                {
                        throw wd;//re-throwing
                }
              };
            wd2.process(a1);//method Call
            break xyz;
        case 2:
            System.out.println("Enter the amt:");
            int a2 = s.nextInt();
            if(!(a2>0 && a2%100==0))//Exception Condition
            {
                DemoException7 de = new DemoException7("Invalid
amt");
                throw de;
            }
            Transaction dp = (int amt)->
                {
                    System.out.println("Amt deposited:"+amt);
                    Transaction.b.bal=Transaction.b.bal+amt;
                    System.out.println("Balance
amt:"+Transaction.b.getBalance());
                    System.out.println("Transaction completed...");
                };
            dp.process(a2);
            break xyz;
```

```
        default:
                System.out.println("Invalid Choice...");
                break xyz;
        }//end of switch
}//end of try
catch(InputMismatchException ime)
{
        System.out.println("Enter only Integer value...");
        break xyz;
}
catch(Exception de)
{
        System.out.println(de.getMessage());
        if(de.getMessage().equals("Invalid pinNo"))
        {
                count++;
        if(count==3)//Nested Simple if
        {
                System.out.println("Transaction blocked...");
                break xyz;
        }
        }//end of if
        else
        {
```

```
                                    break xyz;
                                }
                            }

                    }//end of loop
                }//end of try-with-resource
                }
}
```

Note:

=>In the process of handling exception in LambdaExpressions,we handle "java.lang.Exception" directly because the Class and method_name is not available.

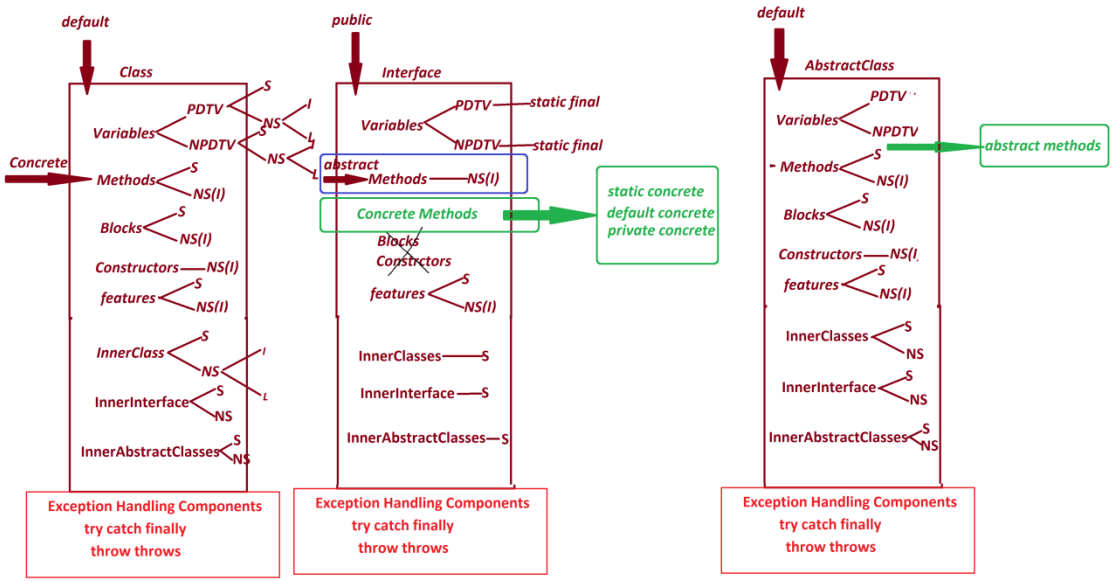
==========

faq:

define Encapsulation process?

=>The process of binding all the programming components into a single unit class is known as Encapsulation process.

Comparision Diagram:



==========

faq:

wt is the diff b/w

(i)function

(ii)member function

(iii)method

(i)function:

=>The part of program which is executed out of main-program in c-lang is known as function.

(ii)member function:

=>The functions which are declared part of classes in c++ lang are known

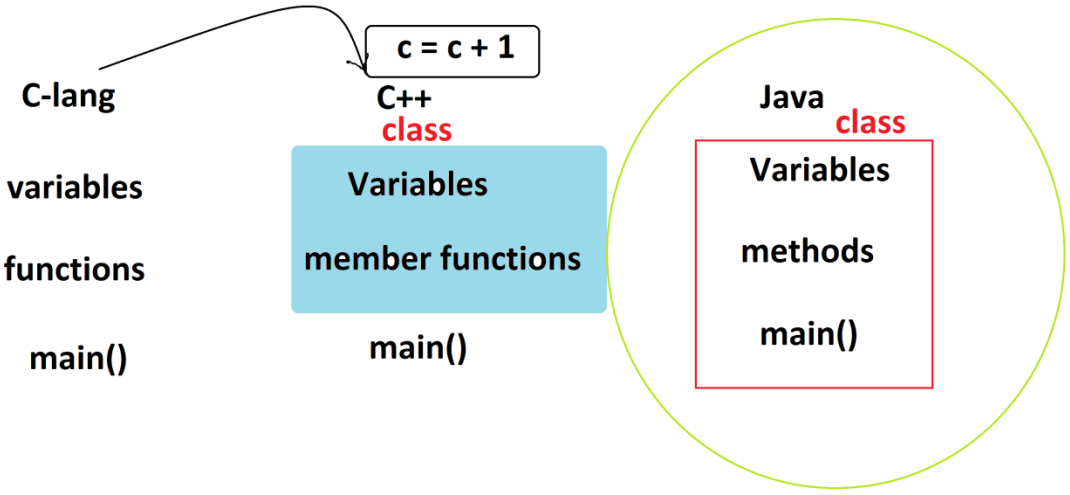
as member functions.

Note:

=>member functions can be declared inside the class and outside class.

(iii)method:

=>The functions which are declared only inside the class in Java-lang are known as methods.



==========

faq:

wt is the diff b/w classes in c++ and Classes in Java?

=>classes in c++ will hold Variables and functions,but cannot hold main().

=>classes in Java will Variables,methods and main()

==========

faq:

define Annotation?

=>The tag based information which is added to the programming component

like Interface,class,method and variable is known as Annotation.

=>we use "@" symbol to represent annotations

=>These Annotations will give information to compiler at compilation stage.

=>The following are two important annotations in CoreJava:

(i)@SuppressWarnings

(ii)@Override

(i)@SuppressWarnings:

=>@SuppressWarnings annotation will provide information to compiler,to close the raised Warnings.

(ii)@Override:

=>@Override annotation will provide information to compiler, to check the method is Overriding method or not

==========