Dt : 13/9/2023
*imp
Handling Multiple Exceptions:
=>To handle multiple exceptions we use the following:
1.we can handle multiple exceptions by declaring multiple catch blocks to a try-block

```
Syntax:
 try
 {
    Exception1
    Exception2
    ….
 }
 catch(Exception1 ob)
 {
   //msg
 }
 catch(Exception2 ob)
 {
  //msg
 }
```

2.We can also handle multiple exceptions by declaring single catch-block to a try-block , Which is introduced by Java7 Version

```
Syntax:
 try
 {
    //Exception1
    //Exception2
    …
 }
 catch(Exception1 | Exception2 | … ob)
 {
  //msg
 }
```

==========

Faq:
Define try-with-resource?
=>try-with-resource statement introduced by Java7 version and in which try is declared with resource.

=>when we use try-with-resource statement the resources are closed automatically , which means no need to declare
Finally-block

Syntax:

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

Ex:

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

Note:
=>catch-block is not mandatory block in try-with-resource statement.

Ex:

```
package maccess;
import java.util.*;
@SuppressWarnings("serial")
public class DemoException3 extends Exception
{
	public DemoException3(String msg)
	{
		super(msg);
	}
	public static void main(String[] args)
	{

	try(Scanner s = new Scanner(System.in);)
	{
	  try
	  {
		System.out.println("Enter the Marks of CoreJava:");
		int mk = s.nextInt();//Exception for NonInteger input
		if(mk<0 || mk>100)//Exception
		{
			DemoException3 ob = new DemoException3("Invalid Marks...");
			throw ob;
		}
		if(mk>=0 && mk<=34)//Exception
		{
			DemoException3 ob = new DemoException3("Failed in CoreJava...");
			throw ob;
		}
		System.out.println("You passed in CoreJava....");
		System.out.println("CoreJava Marks:"+mk);

	  }
	  catch(InputMismatchException | DemoException3 ob)//Java7
	  {
		System.out.println(ob.getMessage());
		System.out.println(ob.toString());
	  }
	}//end of outer try
	 }

}
```

==========

Dt : 14/9/2023

Faq:
Define Enhanced try-with-resource statement?
  =>Enhanced try-with-resource statement introduced by Java9 version and in which the resources are declared
      Outside the try and the resource-ref-variables are declared with try.
syntax:
resource1;
resource2;
try(res1_ref_var;res2_ref_var;….)
{
  //statements
}

Ex:
Scanner s = new Scanner(System.in);
try(s;)
{
  //statements
}
==========
Faq:
Define java.lang.NullPointerException?
  =>java.lang.NullPointerException is raised when we perform operations on NonPrimitive DataType
    Variables holding null-value.

Program : DemoException4.java

```
package maccess;
import java.util.*;
public class DemoException4
{
      public static String compName;
      public static void main(String[] args)
      {
        Scanner s = new Scanner(System.in);
        try(s;)
        {
              try
              {
                    System.out.println("CustName:");
                    String custName = s.nextLine();
                    System.out.println("Enter the PhoneNo:");
                    long phNo = s.nextLong();//Exception for NonIntegerValue
                    System.out.println("****Details******");
                    System.out.println("CustName:"+custName);
```

```
                System.out.println("PhoneNo:"+phNo);
                int len = compName.length();
                System.out.println("CompanyName:"+compName);
                System.out.println("Characters in CompanyName:"+len);
            }
            catch(InputMismatchException ime)
            {
                System.out.println("Enter only Integer value....");
            }
        }//end of Enhanced try-with-resource
    }

}
```

==========

*imp

PolyMorphism in Java:

=>The Process in which programming Components having more than one form is known as Polymorphism.

Poly - Many

Morphism - Forms

Types of PolyMorphism:

=>PolyMorphism categorised into two types:

1.Dynamic PolyMorphism

2.Static PolyMorphism

1.Dynamic PolyMorphism:

=>The PolyMorphism as execution stage is known as Dynamic PolyMorphism or Runtime PolyMorphism.

Ex:

Method Overriding process

2.Static PolyMorphism:

=>The PolyMorphism at compilation stage is known as Static PolyMorphism or Compile Time PolyMorphism

Ex:

Method Overloading process