Experiment No.____________ **Date___/___/2020**

TITLE OF EXPERIMENT: - **A program in Java to try and catch for Exception handling**

DIVISION:___________________________________ BRANCH: __________________________

BATCH:_____________________________________ ROLL NO.: _________________________

PERFORMED ON DATE: ____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 10

Aim: Write a java program which use try and catch for exception handling.

Objective: To learn try and catch for exception handling in Java

Software:

1. Eclipse
2. JDK 16

Theory:

Java Exceptions

When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things.

When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an **exception** (throw an error).

In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

Exception Handling in Java :

The **Exception Handling in Java** is one of the powerful *mechanism to handle the runtime errors* so that the normal flow of the application can be maintained. Try-catch block which is used for exception handling.

Advantage of Exception Handling

The core advantage of exception handling is **to maintain the normal flow of the application**. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions.

1. Try block

The try block contains set of statements where an exception can occur. A try block is always followed by a catch block, which handles the exception that occurs in associated try block. A try block must be followed by catch blocks or finally block or both.

Syntax of try block

```
try{
  //statements that may cause an exception
}
```

While writing a program, if you think that certain statements in a program can throw a exception, enclosed them in try block and handle that exception.

2. Catch block

A catch block is where you handle the exceptions, this block must follow the try block. A single try block can have several catch blocks associated with it. You can catch different exceptions in different catch blocks. When an exception occurs in try block, the corresponding catch block that handles that particular exception executes. For example if an arithmetic exception occurs in try block then the statements enclosed in catch block for arithmetic exception executes.

Syntax of try catch in java

```
try
{
    //statements that may cause an exception
}
catch (exception(type) e(object))
{
    //error handling code
}
```

3. Java try and catch

The try statement allows you to define a block of code to be tested for errors while it is being executed.

The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

The try and catch keywords come in pairs:

Syntax

```
try {

  //  Block of code to try

}

catch(Exception e) {

  //  Block of code to handle errors

}
```

4. Multiple catch blocks in Java

The few rules about multiple catch blocks.

1. As I mentioned above, a single try block can have any number of catch blocks.
2. A generic catch block can handle all the exceptions. Whether it is ArrayIndexOutOfBoundsException or ArithmeticException or NullPointerException or any other type of exception, this handles all of them.

```
catch(Exception e){
  //This catch block catches all the exceptions
}
```

3. If no exception occurs in try block then the catch blocks are completely ignored.
4. Corresponding catch blocks execute for that specific type of exception:
catch(ArithmeticException e) is a catch block that can handle ArithmeticException
catch(NullPointerException e) is a catch block that can handle NullPointerException
5. We can also use throw exception.

5. Finally block

Finally block executes whether an exception occurs or not. You should place those statements in finally blocks, that must execute whether exception occurs or not.

Program:

1. Program for lab11(main thread_source):

```java
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.Scanner;


public class Lab11 {


    public static void main(String[] args) {
        // TODO code application logic here
       Scanner input=new Scanner(System.in);
       int ch=0;
       System.out.println("1. Arithmetic Exception\n2.Null Pont 
Exception\n3.Array iNdex Out of Bound");
       System.out.println("4. File Handling Exception\n5.user 
defined Exception\n6. Number formating");
       System.out.println("enter choice:");
         ch=input.nextInt();

         switch(ch)
         {
             case 1:

                        /* To check Arithmetic exception*/
                      int a,b,c;
                      int result;
                  System.out.println("Enter value of a :\t");
                  a=input.nextInt();
                  System.out.println("Enter value of b:\t");
                  b=input.nextInt();
                  System.out.println("Enter value of c :\t");
                  c=input.nextInt();
                  try
                              {
                                    result = a / (b-c);
                              }
                              catch (ArithmeticException ae)
                               {
                                      System.out.println("Cannot 
divided by zero."+ae);
                              }
                  finally{
                      System.out.println("Finally Bock is excecuted 
Always");
                  }
                  break;
            case 2:
                    String s=null;
```

```
                     char c1;
                         try{
                              c1=s.charAt(1);

                         }
                     catch(NullPointerException ex)
                 {
                      System.out.println("Null pointer Exception");
                 }
                         break;

            case 3:
                     int arr[]={1,2,3,4};
                     try{
                         arr[7]=45;
                     }
                      catch(ArrayIndexOutOfBoundsException ai)
                     {
                     System.out.println("Required Array index is not 
available.");
                     }
                     break;
            case 4://file handling
                try{

                    FileInputStream f1=new 
FileInputStream("D:\\JAVAPROJECT\\FDP_FJP\\src\\fdp_fjp\\Saving1.jav
a");
                    DataInputStream in=new DataInputStream(f1);
                    System.out.println("File is found");
                }
                catch(FileNotFoundException ne){
                    System.out.println("File not found");
                }
                break;


         case 5://user defined exception
          try{
              Circle cr1= new Circle(10,10,0);
              System.out.println("Circle Created");
          }
          catch(InvalidRadiusException e)
          {
              e.printError();
          }
             break;
         case 6:
             //number formating
```

```
                try{
                    // String s1="PVGCOET";
                    String s2="123";
                     int x= Integer.parseInt(s2);
                   System.out.println(" Interger number is "+x);
                 }
                 catch(NumberFormatException e){
                     System.out.println(e.getMessage()+" is not an
interger");
                 }
                break;

               default:System.out.println("Wrong Choice");
          }

    }

}
```

2. Program for InvalidRadiusException (thread1):

```
class InvalidRadiusException extends Exception {
    private double r;

    public InvalidRadiusException(double radius)
    {
        r=radius;
    }
    public void printError()
    {
        System.out.println("Radius ["+r+"} is not valid");
    }
}
```

3. Program for Circle (thread2):

```
public class Circle {
    int x,y,r;
    public Circle(int centreX,int centreY,int radius)throws
InvalidRadiusException
    {
        if(radius<=0){
            throw new InvalidRadiusException(radius);
                  }
        else
        {
            this.x=centreX;
             this.y=centreY;
              this.r=radius;
        }
    }

}
```

Output: case1:

```
1.Arithmetic Exception
2.Null Pont Exception
3.Array iNdex Out of Bound
4. File Handling Exception
5.user defined Exception
6. Number formating
enter choice:
1
Enter value of a :
12
Enter value of b:
23
Enter value of c :
20
Finally Bock is excecuted Always
```

```
1. Arithmetic Exception
2.Null Pont Exception
3.Array iNdex Out of Bound
4. File Handling Exception
5.user defined Exception
6. Number formating
enter choice:
1
Enter value of a :
12
Enter value of b:
0
Enter value of c :
0
Cannot divided by zero.java.lang.ArithmeticException: / by zero
Finally Bock is excecuted Always
```

Case2:

```
1. Arithmetic Exception
2.Null Pont Exception
3.Array iNdex Out of Bound
4. File Handling Exception
5.user defined Exception
6. Number formating
enter choice:
2
Null pointer Exception
```

Case3:

```
1. Arithmetic Exception
2.Null Pont Exception
3.Array iNdex Out of Bound
4. File Handling Exception
5.user defined Exception
6. Number formating
enter choice:
3
Required Array index is not available.
```

Case4:

```
1. Arithmetic Exception
2.Null Pont Exception
3.Array iNdex Out of Bound
4. File Handling Exception
5.user defined Exception
6. Number formating
enter choice:
4
File not found
```

Case 5:

```
1. Arithmetic Exception
2.Null Pont Exception
3.Array iNdex Out of Bound
4. File Handling Exception
5.user defined Exception
6. Number formating
enter choice:
5
```

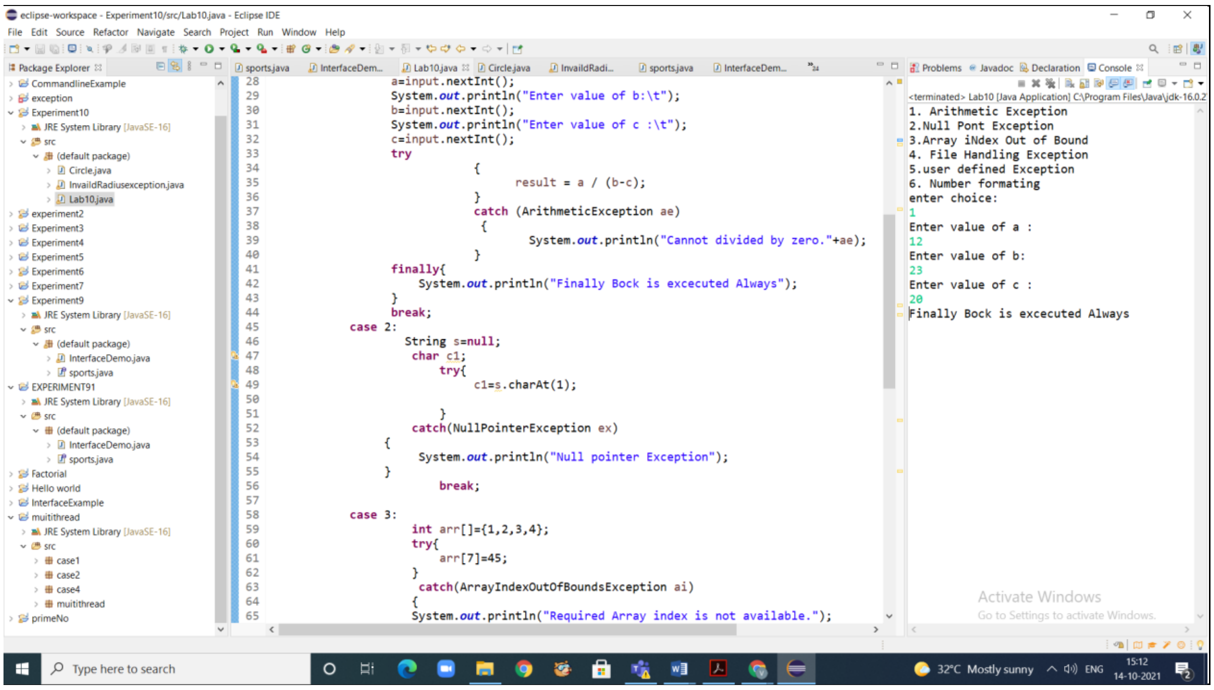
```
Radius [0.0} is not valid
```

Case 6:

```
1. Arithmetic Exception
2.Null Pont Exception
3.Array iNdex Out of Bound
4. File Handling Exception
5.user defined Exception
6. Number formating
enter choice:
6
 Interger number is 123
```

Conclusion:

Screenshot's of Program and Result:



```
                a=input.nextInt();
                System.out.println("Enter value of b:\t");
                b=input.nextInt();
                System.out.println("Enter value of c :\t");
                c=input.nextInt();
                try
                            {
                                 result = a / (b-c);
                            }
                            catch (ArithmeticException ae)
                             {
                                   System.out.println("Cannot divided by zero."+ae);
                            }
                finally{
                    System.out.println("Finally Bock is excecuted Always");
                }
                break;
          case 2:
                  String s=null;
                   char c1;
                       try{
                            c1=s.charAt(1);

                       }
                   catch(NullPointerException ex)
               {
                    System.out.println("Null pointer Exception");
               }
                       break;

          case 3:
                   int arr[]={1,2,3,4};
                   try{
                       arr[7]=45;
                   }
                    catch(ArrayIndexOutOfBoundsException ai)
                   {
                   System.out.println("Required Array index is not available.");
```

```
<terminated> Lab10 [Java Application] C:\Program Files\Java\jdk-16.0.2
1. Arithmetic Exception
2.Null Pont Exception
3.Array iNdex Out of Bound
4. File Handling Exception
5.user defined Exception
6. Number formating
enter choice:
1
Enter value of a :
12
Enter value of b:
23
Enter value of c :
20
Finally Bock is excecuted Always
```