| |
|---|
| Experiment No. 9 |
| Implement a program on Exception handling. |
| Date of Performance: |
| Date of Submission: |

**Aim:** Implement a program on Exception handling.

**Objective**: To able handle exceptions occurred and handle them using appropriate keyword

**Theory:**

The Exception Handling in Java is one of the powerful mechanisms to handle the runtime errors so that the normal flow of the application can be maintained.

Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

Java Exception Keywords

Java provides five keywords that are used to handle the exception. The following table describes each.

| Keyword | Description |
|---|---|
| try | The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally. |
| catch | The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later. |
| finally | The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not. |
| throw | The "throw" keyword is used to throw an exception. |
| throws | The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature. |

```
public class JavaExceptionExample{

 public static void main(String args[]){

  try{

    //code that may raise exception

    int data=100/0;
```

```
      }catch(ArithmeticException e){System.out.println(e);}

       //rest code of the program

       System.out.println("rest of the code...");

      }

}
```

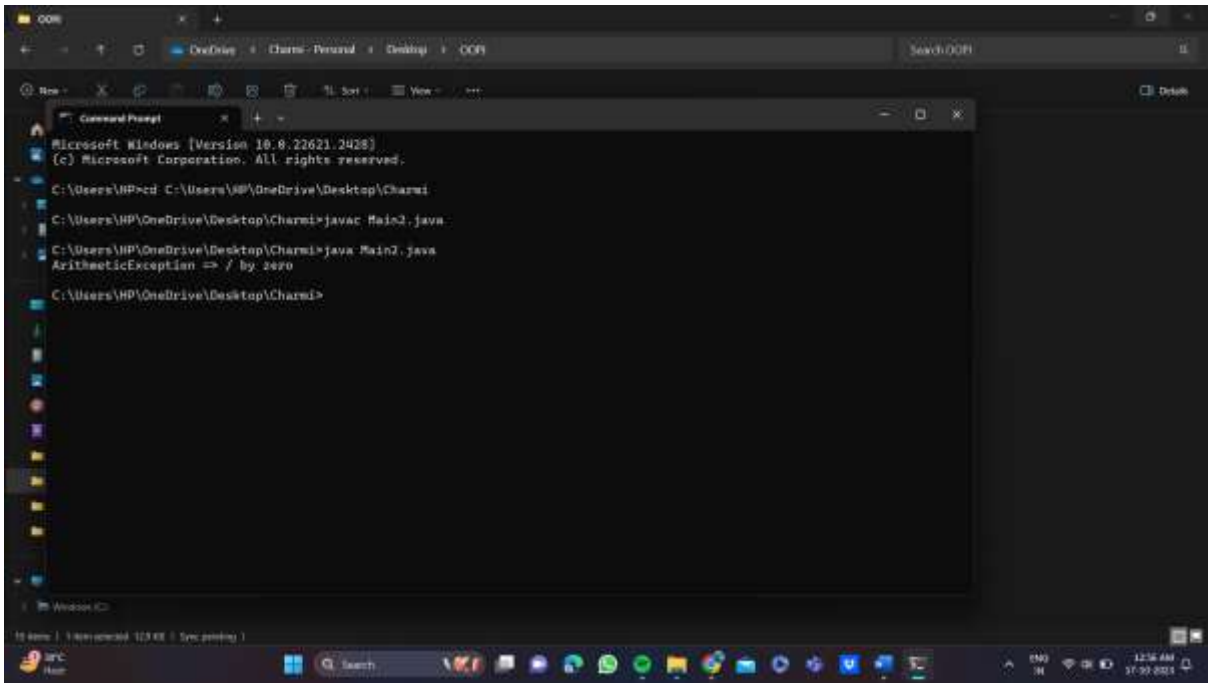
**Output:**

```
Exception in thread main java.lang.ArithmeticException:/ by zero
rest of the code...
```

**Code:**

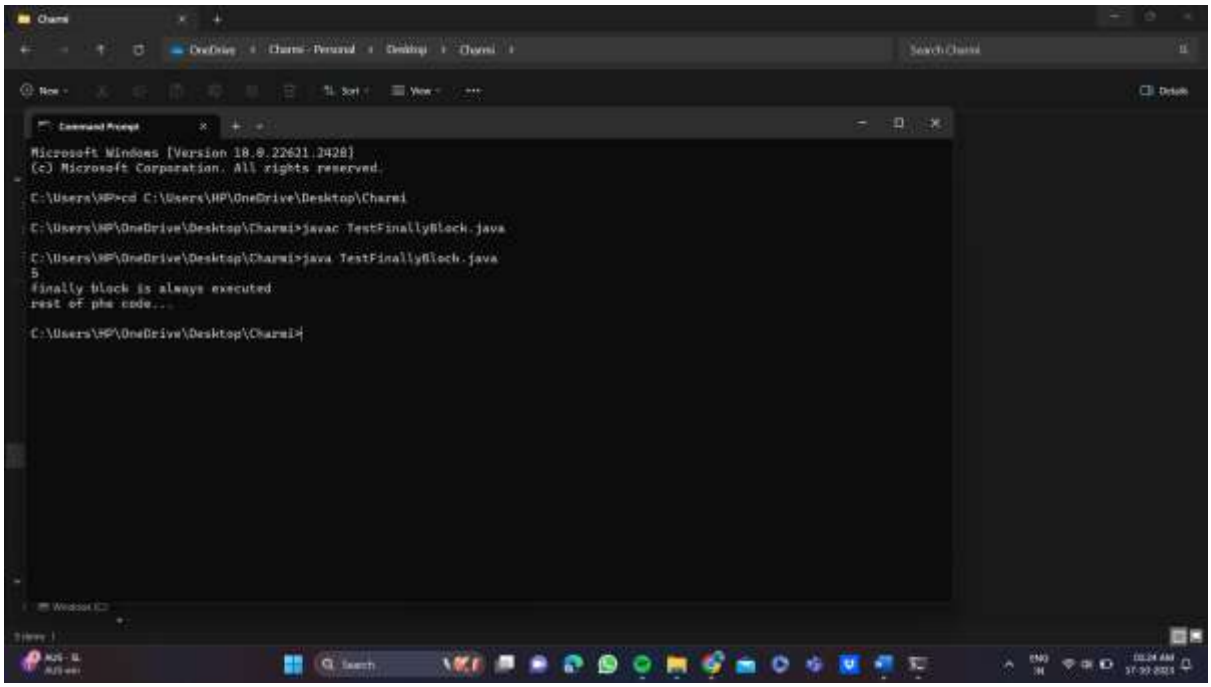
**1}** Try-catch

```
class Main2
{
public static void main(String args[])
{
try{
  int divideByZero = 8/0;
  System.out.println("Rest of code in try block");
    }

    catch (ArithmeticException e) {
      System.out.println("ArithmeticException => " + e.getMessage());
    }
  }
}
```



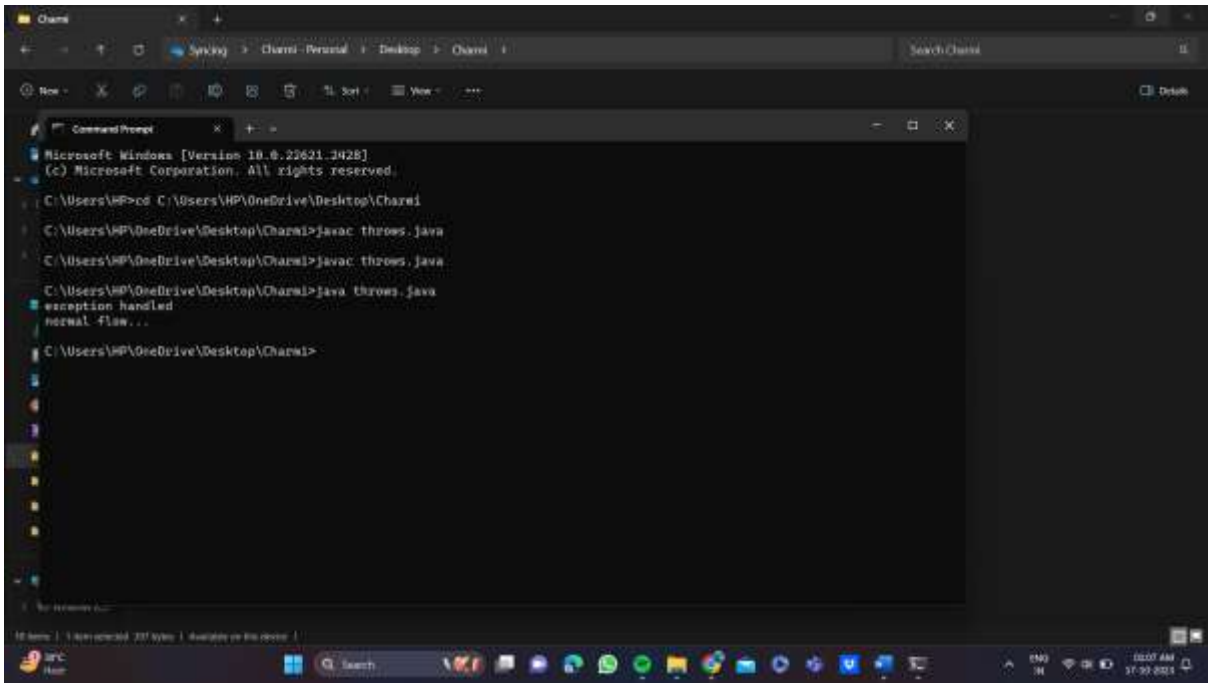

**2}** finally

```
class TestFinallyBlock {
  public static void main(String args[]){
  try{
   int data=25/5;
   System.out.println(data);
  }
  catch(NullPointerException e){
System.out.println(e);
}
 finally {
System.out.println("finally block is always executed");
}

System.out.println("rest of phe code...");
  }
}
```



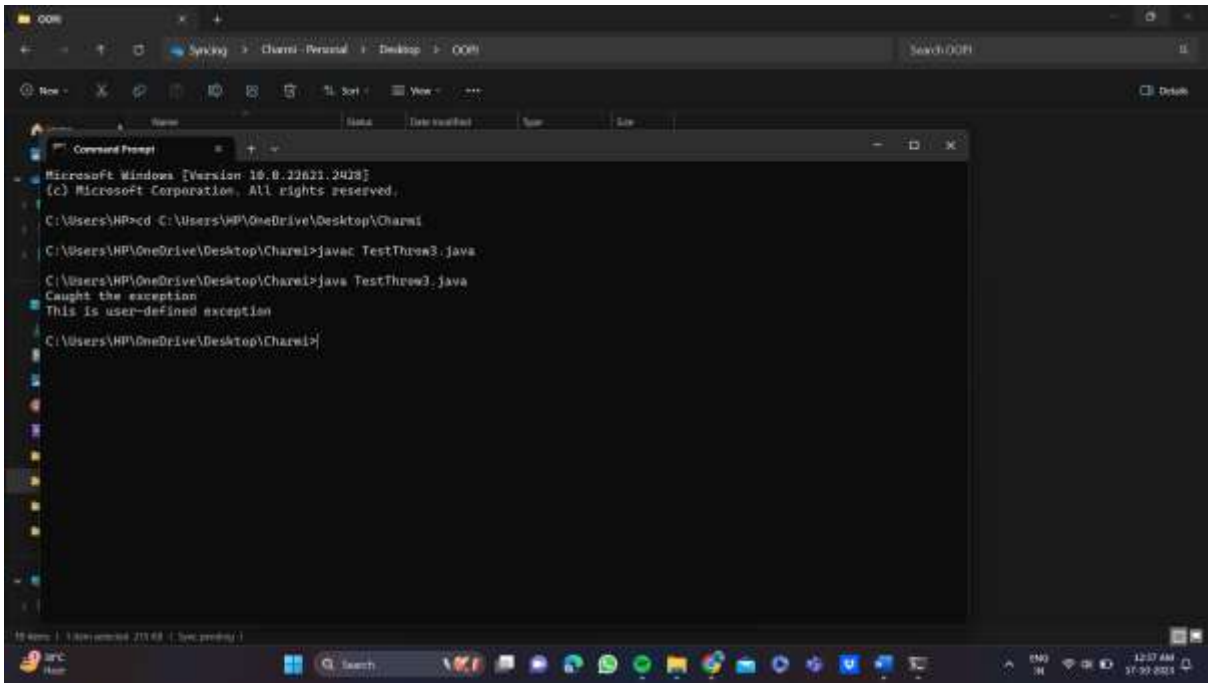

**3}**throws

```
import java.io.IOException;
 class Testthrows2{
   public static void main(String args[]){
    try{
     M m=new M();
     m.method();
    }catch(Exception e){System.out.println("exception handled");}

    System.out.println("normal flow...");
  }
}
class M {
    void method() throws IOException {
        throw new IOException("device error");
    }
}
```




**4}** throw

```
class TestThrow3
{
   public static void main(String args[])
   {
      try
      {
         throw new UserDefinedException("This is user-defined exception");
      }
      catch (UserDefinedException ude)
      {
         System.out.println("Caught the exception");
         System.out.println(ude.getMessage());
      }
   }
}
class UserDefinedException extends Exception
{
   public UserDefinedException(String str)
   {
      super(str);
   }
}
```




**Conclusion:**

In Java, exceptions are handled using a combination of the try, catch, finally, and throw keywords. The primary mechanism for handling exceptions is the try-catch block. Code that may potentially throw an exception is placed within a try block, and you provide one or more catch blocks to handle specific types of exceptions. You can also use a finally block after the try-catch blocks. Code within the finally block is executed regardless of whether an exception was thrown or not. It's typically used for cleanup actions (e.g., closing resources). You can use the throw keyword to explicitly throw an exception within your code. This is often done when you encounter an exceptional situation that your code can't handle, and you want to pass the control to an exception handler.