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:

```
import java.io.*;
class ExceptHandling
{
public static void main(String args[])
{
int a=0,b=0,result;
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String str;
System.out.println("Enter two numbers: ");
try
{
str=br.readLine();
a=Integer.parseInt(str);
str=br.readLine();
b=Integer.parseInt(str);
result = a/b;
System.out.println("The Quotient= "+result);
}
catch(ArithmeticException ae)
{
System.out.println("Exception has occurred. You have entered the divisor as
zero [0]");
}
catch(IOException ioe)
{
System.out.println("IOException occurred");
}
catch(NumberFormatException ne)
{
System.out.println("Invalid Number");
```

```
    }
    }
    }
```

Output:

```
C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>javac ExceptHandling.java

C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>java ExceptHandling.java
Enter two numbers:
10
2
The Quotient= 5

C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>java ExceptHandling.java
Enter two numbers:
-19
0
Exception has occurred. You have entered the divisor as zero [0]

C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>java ExceptHandling.java
Enter two numbers:
10.5
Invalid Number
```

Conclusion:

Comment on how exceptions are handled in JAVA.

- Exceptions are handled in Java using the try-catch-finally blocks. The try block contains the code that may throw an exception. The catch block contains the code that handles the specific exception. The finally block contains the code that is always executed, regardless of whether an exception occurs or not The ExceptionType can be any subclass of the Throwable class, such as IOException, ArithmeticException, NullPointerException, etc. We can also have multiple catch blocks for different types of exceptions, or use a single catch block with a generic Exception type to handle all exceptions. We can also use the throw keyword to explicitly throw an exception, and the throws keyword to declare that a method may throw an exception