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:

```
public class TestThrow1
{
   //function to check if person is eligible to vote or not
   public static void validate(int age)
        {
      if(age&lt;18)
        {
        //throw Arithmetic exception if not eligible to vote
        throw new ArithmeticException(&quot;Person is not eligible to vote&quot;);
       }
      else
```

```
        {
        System.out.println(&quot;Person is eligible to vote!!&quot;);
       }
    }
//main method
   public static void main(String args[])
  {
      //calling the function
      validate(13);
      System.out.println(&quot;rest of the code...&quot;);
 }
}
```

Conclusion:

Exception handling in Java is crucial for building robust applications. By using try-catch blocks, developers can manage errors effectively, preventing crashes and ensuring smooth program execution. The ability to define custom exceptions and the distinction between checked and unchecked exceptions enhances error management capabilities.
Overall, a solid understanding of exception handling enables developers to create more resilient and maintainable Java applications, making it an essential skill in software development.