Vidyavardhini's College of Engineering and Technology
Department of Artificial Intelligence & Data Science

Experiment No. 9
Name: Atharva Borse
Roll no: 07
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:

```
// Java program to demonstrate Exception Handling
public class JavaExceptionExample {

  // Method that can throw multiple exceptions
  public static void divide(int a, int b) throws ArithmeticException {
    if (b == 0) {
      // Throwing ArithmeticException for division by zero
      throw new ArithmeticException("Division by zero is not allowed.");
    } else {
      // Performing division if no exception
      int result = a / b;
      System.out.println("Result: " + result);
    }
  }
```

```
public static void main(String[] args) {
    // Try block to catch exceptions
    try {
        // Simulating a potential exception (division by zero)
        divide(10, 0);
    }
    // Catching specific exception (ArithmeticException)
    catch (ArithmeticException e) {
        System.out.println("Caught Exception: " + e.getMessage());
    }
    // Finally block that always executes
    finally {
        System.out.println("Finally block executed.");
    }

    // Using throw to manually throw an exception
    try {
        // Manually throwing an exception
        throw new NullPointerException("Manually thrown NullPointerException.");
    }
    catch (NullPointerException e) {
        System.out.println("Caught Exception: " + e.getMessage());
    }
    finally {
        System.out.println("Finally block executed after manual throw.");
    }
}
```

Output Generated Files

```
Caught Exception: Division by zero is not allowed.
Finally block executed.
Caught Exception: Manually thrown NullPointerException.
Finally block executed after manual throw.
```

}

Conclusion:

Comment on how exceptions are handled in JAVA.