Experiment No.9
Aim: Basic programming constructs like branching and looping
Name: Aman Mehtar
Roll no: 32
Date of Performance: 12/9/24
Date of Submission: 25/9/24

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 ExceptionHandling {
    public static void main(String args[]) {
        try {
            // Code that may raise an exception
            int data = 100 / 0; // This will cause an ArithmeticException (division by zero)
        } catch (ArithmeticException e) {
            // Handle the ArithmeticException
            System.out.println("Exception caught: " + e);
        } finally {
            // Code inside the 'finally' block will execute whether exception occurs or not
            System.out.println("This block (finally) is always executed.");
        }
        // Rest of the code
        System.out.println("Rest of the code...");
    }
}
```

Output:

```
Exception caught: java.lang.ArithmeticException: / by zero
This block (finally) is always executed.
Rest of the code…
```

```
Exception caught: java.lang.ArithmeticException: / by zero
This block (finally) is always executed.
Rest of the code...
```

Conclusion:

In Java, exceptions are handled using a combination of `try`, `catch`, and `finally` blocks.

- The `try` block contains the code that may cause an exception.
- The `catch` block handles the exception if it occurs, allowing the program to continue executing the rest of the code.
- The `finally` block always runs after the `try` and `catch`, ensuring that any necessary cleanup actions are executed regardless of whether an exception was thrown or caught. Thus, exception handling ensures that runtime errors do not disrupt the normal flow of the program