Experiment No. 10
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 program on User Defined Exception.

Objective: To illustrate the creation and handling of custom exceptions in Java for enhanced error management.

Theory:

An exception is an issue (run time error) that occurred during the execution of a program. When an exception occurred the program gets terminated abruptly and, the code past the line that generated the exception never gets executed.

Java provides us the facility to create our own exceptions which are basically derived classes of Exception. Creating our own Exception is known as a custom exception or user-defined exception. Basically, Java custom exceptions are used to customize the exception according to user needs. In simple words, we can say that a User-Defined Exception or custom exception is creating your own exception class and throwing that exception using the 'throw' keyword.

For example, MyException in the below code extends the Exception class.

Why use custom exceptions?

Java exceptions cover almost all the general types of exceptions that may occur in the programming. However, we sometimes need to create custom exceptions.

Following are a few of the reasons to use custom exceptions:

- To catch and provide specific treatment to a subset of existing Java exceptions.
- Business logic exceptions: These are the exceptions related to business logic and workflow. It is useful for the application users or the developers to understand the exact problem.

In order to create a custom exception, we need to extend the Exception class that belongs to **java.lang package.**

Example: We pass the string to the constructor of the superclass- Exception which is obtained using the "getMessage()" function on the object created.

// A Class that represents use-defined exception

```
class MyException extends Exception {
        public MyException(String s)
        {
                // Call constructor of parent Exception
                super(s);
        }
}


// A Class that uses above MyException
public class Main {
        // Driver Program
        public static void main(String args[])
        {
                try {
                        // Throw an object of user defined exception
                        throw new MyException("UserDefined Exception");
                }
                catch (MyException ex) {
```

```
                System.out.println("Caught");



                // Print the message from MyException object

                System.out.println(ex.getMessage());

            }

        }

}
```

Output:

```
Caught
UserDefined Exception
```

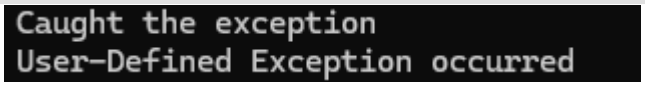
Code:

```
// A Class that uses the above MyException
public class UserDefinedException {
    // Driver Program
    public static void main(String args[]) {
        try {
            // Throw an object of user-defined exception
            throw new MyException("User-Defined Exception occurred");
        } catch (MyException ex) {
            // Exception is caught here
            System.out.println("Caught the exception");

            // Print the message from MyException object
            System.out.println(ex.getMessage());
        }
    }
```

```
}

// A Class that represents user-defined exception
class MyException extends Exception {
    // Constructor that accepts a string message
    public MyException(String s) {
        // Call the constructor of the parent Exception class
        super(s);
    }
}
```

Output:

```
Caught the exception
User-Defined Exception occurred
```



Conclusion:

User-defined exceptions in Java allow developers to create custom exceptions for specific scenarios, which aren't covered by built-in exceptions. By extending the `Exception` class, a custom exception can be thrown using the `throw` keyword and handled in a `try-catch` block. This is useful for providing detailed exception messages, specific business logic errors, or custom error handling, making the code easier to understand and maintain.