Experiment No. 10
Implement program on User Defined Exception
Date of Performance:
Date of Submission:

Aim: Implement program on User Defined Exception.

Objective:

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:

```
// Custom Exception Class
class InvalidAgeException extends Exception
{
   public InvalidAgeException(String message)
```

```
        {
        super(message);
        }
}

// Class to check age
class AgeValidator
{
   void validateAge(int age) throws InvalidAgeException
        {
      if (age < 18)
        {
        throw new InvalidAgeException("Age must be 18 or older.");
        }
        else
        {
        System.out.println("Age is valid: " + age);
        }
   }
}

// Main class to run the program
```

```
public class Main
{
   public static void main(String[] args)
{
      AgeValidator validator = new AgeValidator();

      // Test with an invalid age
      try
         {
         validator.validateAge(16); // This will throw the exception
         }
         catch (InvalidAgeException e)
         {
         System.out.println("Exception caught: " + e.getMessage());
           }

      // Test with a valid age
      Try
         {
         validator.validateAge(20); // This will not throw the exception
```

```
        }
        catch (InvalidAgeException e)
        {
       System.out.println("Exception caught: " + e.getMessage());
    }
  }
}
```

Conclusion:

In summary, mastering user-defined exceptions is essential for Java developers seeking to write clean, efficient, and user-friendly applications. By leveraging this feature, developers can create applications that handle unexpected situations gracefully, ultimately contributing to a more robust and reliable software experience..