User-defined Custom Exception in Java

Difficulty Level : Easy Last Updated : 10 Feb, 2022

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 expception
```

```
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("GeeksGeeks");
        }
        catch (MyException ex) {
            System.out.println("Caught");

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

Output

```
Caught
GeeksGeeks
```

In the above code, the constructor of MyException requires a string as its argument. The string is passed to the parent class Exception's constructor using super(). The constructor of the Exception class can also be called without a parameter and the call to super is not mandatory.

```
// A Class that represents use-defined expception

class MyException extends Exception {
}

// A Class that uses above MyException
public class setText {
    // Driver Program
    public static void main(String args[])
    {
        try {
            // Throw an object of user defined exception
            throw new MyException();
        }
        catch (MyException ex) {
            System.out.println("Caught");
            System.out.println(ex.getMessage());
        }
    }
}
```

Output

```
Caught
null
```

This article is contributed by **Pranjal Mathur**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.