

BASICS OF JAVA

OOPS CONCEPTS

STRING HANDLING

EXCEPTION HANDLING

JAVA MULTITHREADING

ADVANCED TOPICS

COLLECTION FRAMEWORK

JAVA GUI

REFLECTION API

RMI APPLICATION

INNER CLASS

WRAPPER CLASS

FILE HANDLING

LIST

SET

MAP

QUEUE & DEQUE

JDBC

LAYOUT MANAGERS

ADVERTISEMENT

# User defined Exception subclass in Java

ADVERTISEMENT

Java provides rich set of built-in exception classes like: ArithmeticException, IOException, NullPointerException etc. all are available in the java.lang package and used in exception handling. These exceptions are already set to trigger on pre-defined conditions such as when you divide a number by zero it triggers ArithmeticException.

Apart from these classes, Java allows us to create our own exception class to provide own exception implementation. These type of exceptions are called user-defined exceptions or **custom** exceptions.

You can create your own exception simply by extending java **Exception** class. You can define a constructor for your Exception (not compulsory) and you can override the `toString()` function to display your customized message on catch. Lets see an example.

## Example: Custom Exception

In this example, we are creating an exception class MyException that extends the Java Exception class and

```
class MyException extends Exception
{
    private int ex;
    MyException(int a)
    {
        ex = a;
    }
    public String toString()
    {
        return "MyException[" + ex + "] is less than zero";
    }
}

class Demo
{
    static void sum(int a,int b) throws MyException
    {
        if(a<0)
        {
```

OUTPUT

MyException[-10] is less than zero

## Example: Custom Exception

Lets take one more example to understand the custom exception. Here we created a class ItemNotFound that extends the Exception class and helps to generate our own exception implementation.

```
class ItemNotFound extends Exception
{
    public ItemNotFound(String s) {
        super(s);
    }
}

class Demo
{
    static void find(int arr[], int item) throws ItemNotFound
    {
        boolean flag = false;
        for (int i = 0; i < arr.length; i++) {
            if(item == arr[i])
                flag = true;
        }
        if(!flag)
        {
```

OUTPUT

ItemNotFound: Item Not Found

## Points to Remember

1. Extend the Exception class to create your own exception class.
2. You don't have to implement anything inside it, no methods are required.
3. You can have a Constructor if you want.
4. You can override the toString() function, to display customized message.

← Prev

Next →

ADVERTISEMENT

Java MCQ Tests

Prepare for Java Interview in TCS, Infosys, etc. companies.

Explore

Java Programs

Java programs with code and output for practice.

Explore

Spring Framework

Learn the most widely used Java framework in the World.

Explore

ADVERTISEMENT

ADVERTISEMENT

- Learn Coding (for beginners)

Tutorial Library

Interview Tests

Curious

Practice Coding

- Coding Courses

Learn HTML

Learn CSS

Learn JavaScript

Resources

C Language

C++/STL

Java

DBMS

Python

PHP

Android

Game Development

Data Structure & Alog.

Operating System

Computer Network

Computer Architecture

Docker

GO Language

GIT Guide

Linux Guide

More...

- Interview Tests

Java Interview Tests

Python Interview Tests

DBMS Interview Tests

Linux Interview Tests

Aptitude Tests

GATE 2022 Tests

More...

Projects/Programs

Python Projects

C Projects

Python Programs

C Programs

C++ Programs

Java Programs

ADVERTISEMENT