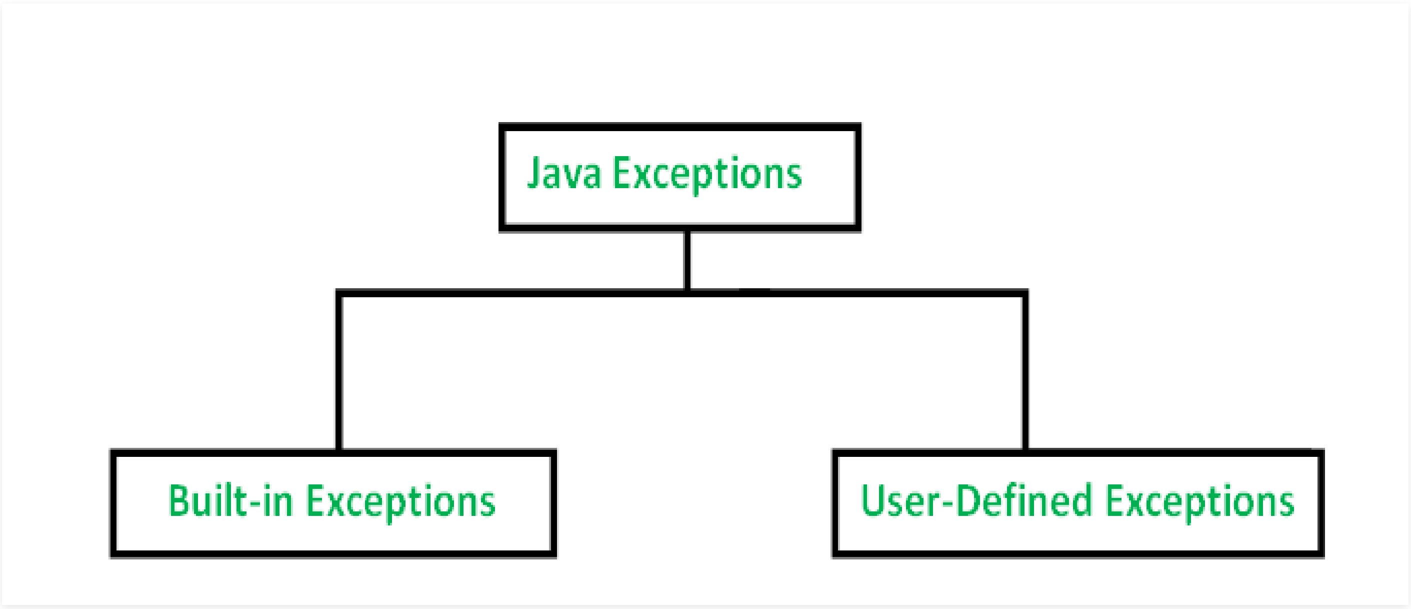
Types of Exception in Java with Examples

Difficulty Level : Medium Last Updated : 28 Jun, 2021

Java defines several types of exceptions that relate to its various class libraries. Java also allows users to define their own exceptions.



Built-in Exceptions

Built-in exceptions are the exceptions which are available in Java libraries. These exceptions are suitable to explain certain error situations. Below is the list of important built-in exceptions in Java.

1. **ArithmeticException**
   It is thrown when an exceptional condition has occurred in an arithmetic operation.
2. **ArrayIndexOutOfBoundsException**
   It is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.
3. **ClassNotFoundException**
   This Exception is raised when we try to access a class whose definition is not found
4. **FileNotFoundException**
   This Exception is raised when a file is not accessible or does not open.
5. **IOException**
   It is thrown when an input-output operation failed or interrupted
6. **InterruptedException**
   It is thrown when a thread is waiting, sleeping, or doing some processing, and it is interrupted.

7. **NoSuchFieldException**
   It is thrown when a class does not contain the field (or variable) specified
8. **NoSuchMethodException**
   It is thrown when accessing a method which is not found.
9. **NullPointerException**
   This exception is raised when referring to the members of a null object. Null represents nothing
10. **NumberFormatException**
    This exception is raised when a method could not convert a string into a numeric format.
11. **RuntimeException**
    This represents any exception which occurs during runtime.
12. **StringIndexOutOfBoundsException**
    It is thrown by String class methods to indicate that an index is either negative or greater than the size of the string

Examples of Built-in Exception:

- **Arithmetic exception**

```java
// Java program to demonstrate ArithmeticException
class ArithmeticException_Demo
{
    public static void main(String args[])
    {
        try {
            int a = 30, b = 0;
            int c = a/b;  // cannot divide by zero
            System.out.println ("Result = " + c);
        }
        catch(ArithmeticException e) {
            System.out.println ("Can't divide a number by 0");
        }
    }
}
```

Output:

```
Can't divide a number by 0
```

- **NullPointer Exception**

```
//Java program to demonstrate NullPointerException
class NullPointer_Demo
{
    public static void main(String args[])
    {
        try {
            String a = null; //null value
            System.out.println(a.charAt(0));
        } catch(NullPointerException e) {
            System.out.println("NullPointerException..");
        }
    }
}
```

Output:

```
NullPointerException..
```

- **StringIndexOutOfBound Exception**

```
// Java program to demonstrate StringIndexOutOfBoundsException
class StringIndexOutOfBound_Demo
{
    public static void main(String args[])
    {
        try {
            String a = "This is like chipping "; // length is 22
            char c = a.charAt(24); // accessing 25th element
            System.out.println(c);
        }
        catch(StringIndexOutOfBoundsException e) {
            System.out.println("StringIndexOutOfBoundsException");
        }
    }
}
```

Output:

```
StringIndexOutOfBoundsException
```

- **FileNotFound Exception**

```java
//Java program to demonstrate FileNotFoundException
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
 class File_notFound_Demo {

    public static void main(String args[])  {
        try {

            // Following file does not exist
            File file = new File("E://file.txt");

            FileReader fr = new FileReader(file);
        } catch (FileNotFoundException e) {
           System.out.println("File does not exist");
        }
    }
}
```

Output:

```
File does not exist
```

- **NumberFormat Exception**

```
// Java program to demonstrate NumberFormatException
class  NumberFormat_Demo
{
    public static void main(String args[])
    {
        try {
            // "akki" is not a number
            int num = Integer.parseInt ("akki") ;

            System.out.println(num);
        } catch(NumberFormatException e) {
            System.out.println("Number format exception");
        }
    }
}
```

Output:

```
Number format exception
```

- **ArrayIndexOutOfBounds Exception**

```
// Java program to demonstrate ArrayIndexOutOfBoundException
class ArrayIndexOutOfBound_Demo
{
    public static void main(String args[])
    {
        try{
            int a[] = new int[5];
            a[6] = 9; // accessing 7th element in an array of
                      // size 5
        }
        catch(ArrayIndexOutOfBoundsException e){
            System.out.println ("Array Index is Out Of Bounds");
        }
    }
}
```

Output:

```
Array Index is Out Of Bounds
```

User-Defined Exceptions

Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, user can also create exceptions which are called 'user-defined Exceptions'.

Following steps are followed for the creation of user-defined Exception.

- The user should create an exception class as a subclass of Exception class. Since all the exceptions are subclasses of Exception class, the user should also make his class a subclass of it. This is done as:

```
class MyException extends Exception
```

- We can write a default constructor in his own exception class.

```
MyException(){}
```

- We can also create a parameterized constructor with a string as a parameter.
  We can use this to store exception details. We can call super class(Exception) constructor from this and send the string there.

```
MyException(String str)
{
   super(str);
}
```

- To raise exception of user-defined type, we need to create an object to his exception class and throw it using throw clause, as:

```
MyException me = new MyException(“Exception details”);
throw me;
```

- The following program illustrates how to create own exception class MyException.
- Details of account numbers, customer names, and balance amounts are taken in the form of three arrays.
- In main() method, the details are displayed using a for-loop. At this time, check is done if in any account the balance amount is less than the minimum balance amount to be apt in the account.

- If it is so, then MyException is raised and a message is displayed “Balance amount is less”.

```java
// Java program to demonstrate user defined exception

// This program throws an exception whenever balance
// amount is below Rs 1000
class MyException extends Exception
{
    //store account information
    private static int accno[] = {1001, 1002, 1003, 1004};

    private static String name[] =
                {"Nish", "Shubh", "Sush", "Abhi", "Akash"};

    private static double bal[] =
         {10000.00, 12000.00, 5600.0, 999.00, 1100.55};

    // default constructor
    MyException() {    }

    // parameterized constructor
    MyException(String str) { super(str); }

    // write main()
    public static void main(String[] args)
    {
        try  {
            // display the heading for the table
            System.out.println("ACCNO" + "\t" + "CUSTOMER" +
                                               "\t" + "BALANCE");

            // display the actual account information
            for (int i = 0; i < 5 ; i++)
            {
                System.out.println(accno[i] + "\t" + name[i] +
                                               "\t" + bal[i]);

                // display own exception if balance < 1000
                if (bal[i] < 1000)
                {
                    MyException me =
                       new MyException("Balance is less than 1000");
                    throw me;
                }
            }
        } //end of try

        catch (MyException e) {
            e.printStackTrace();
        }
    }
}
```

RunTime Error

```
MyException: Balance is less than 1000
   at MyException.main(fileProperty.java:36)
```

Output:

```
ACCNO    CUSTOMER    BALANCE
1001    Nish    10000.0
1002    Shubh     12000.0
1003    Sush    5600.0
1004    Abhi    999.0
```

Related Articles:

- Checked vs Unchecked Exceptions in Java
- Catching base and derived classes as exceptions
- Quiz on Exception Handling

This article is contributed by **Nishant Sharma**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.
Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.