Exception Hierarchy

Types of Exceptions in Java

Basically, there are two types of exceptions in java API. They are:

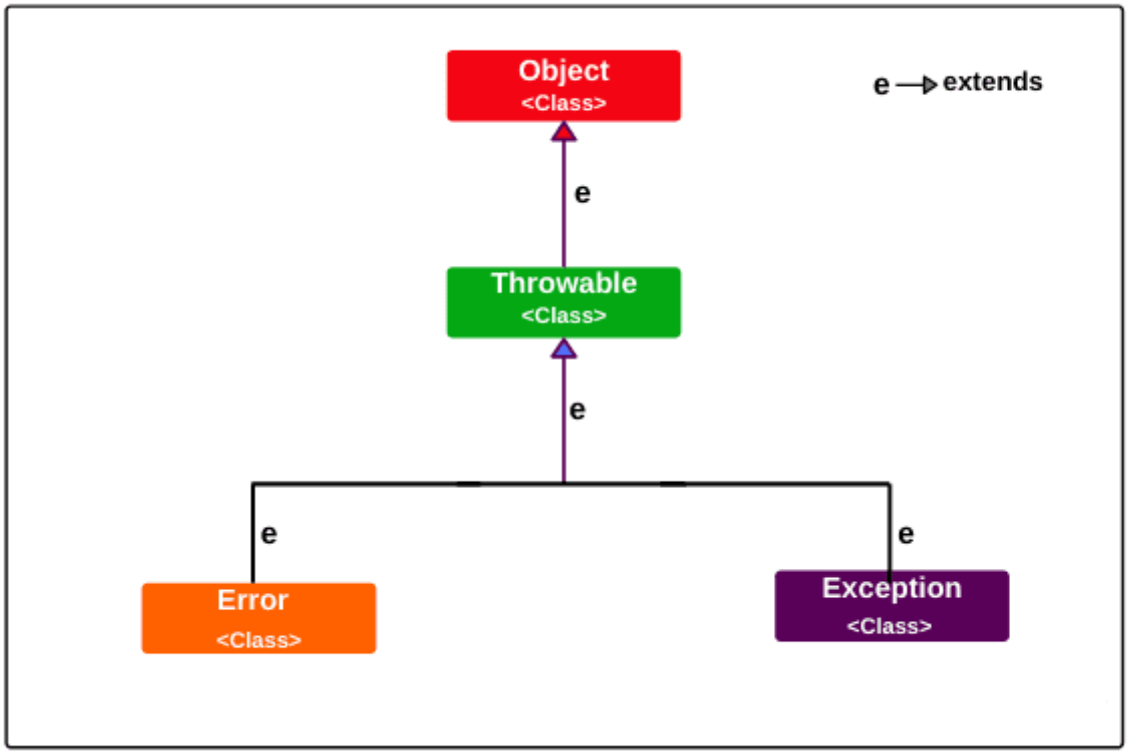
1. **Predefined Exceptions** (Built-in-Exceptions)
2. **Custom Exceptions**

Predefined Exceptions:

Predefined exceptions are those exceptions that are already defined by Java system. These exceptions are also called built-in-exceptions.

JavaAPI supports exception handling by providing the number of predefined exceptions. These predefined exceptions are represented by classes in java.

When a predefined exception occurs, JVM (Java runtime system) creates an object of predefined exception class. All exceptions are derived from java.lang.Throwable class but not all exception classes are defined in the same package.



1. Throwable class: Throwable class which is derived from Object class, is a top of exception hierarchy from which all exception classes are derived directly or indirectly. It is the root of all exception classes. It is present in java.lang package.

Throwable class is the superclass of all exceptions in java. This class has two subclasses: Error and Exception. Errors or exceptions occurring in java programs are objects of these classes. Using Throwable class, you can also create your own custom exceptions.

2. Error: Error class is the subclass of Throwable class and a superclass of all the runtime error classes. It terminates the program if there is problem-related to a system or resources (JVM).

An error generally represents an unusual problem or situation from which it is difficult to recover. It does not occur by programmer mistakes. It generally occurs if the system is not working properly or resource is not allocated properly.

VirtualMachineError, StackOverFlowError, AssertionError, LinkageError, OutOfMmeoryError,…..

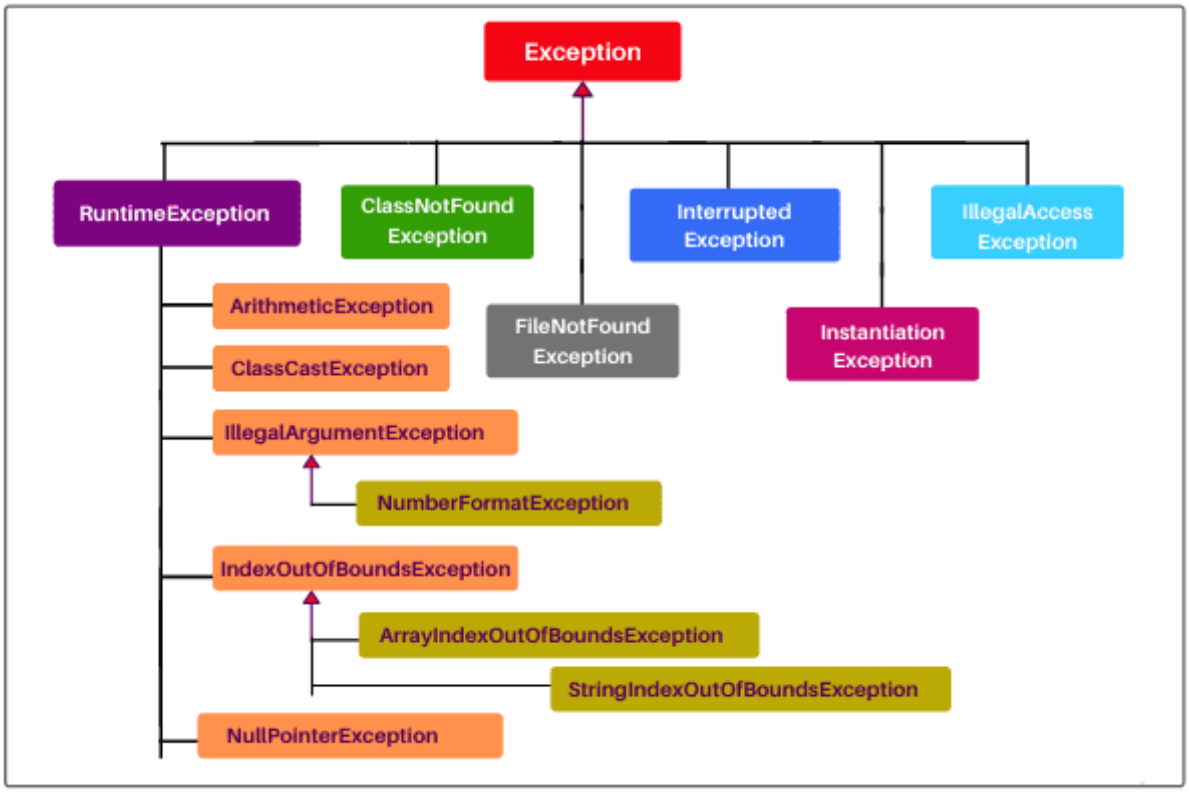
3. Exception: It is represented by an Exception class that represents errors caused by
the program and by external factors. Exception class is a subclass of Throwable class and a superclass of all the exception classes.

All the exception classes are derived directly or indirectly from the Exception class. They originate from within the application.

The exception class provides two constructors:

- public Exception() (Default constructor)
- public Exception(String message) (It takes a message string as argument)

Each of the exception classes provides two constructors: one with no argument and another with a String type argument. Exception class does not provide its own method. It inherits all methods provided by Throwable class.



RuntimeException class (Unchecked Exceptions)

RuntimeException class is a subclass of the Exception class. It is thrown by JVM or programmatically when an arithmetic operation performed in the program is incorrect or defect/bug occurs in the program's code.

1. ArithmeticException: This exception is thrown when arithmetic problems, such as a number is divided by zero, is occurred. That is, it is caused by maths error.

2. ClassCastException:

The ClassCastException is a runtime exception that is thrown by JVM when we

attempt to invalid typecasting in the program. That is, it is thrown when we cast an object to a subclass of which an object is not an instance.

---

3. IllegalArgumentException:
This runtime exception is thrown by programmatically when an illegal or appropriate argument is passed to call a method. This exception class has further two subclasses:

- NumberFormatException
- IllegalThreadStateException

NumericFormatException: NumberFormatException is thrown by programmatically when we try to convert a *string* into the numeric type and the process of illegal conversion fails. That is, it occurs due to the illegal conversion of a string to a numeric format.

IllegalThreadStateException: IllegalThreadStateException exception is a runtime exception that is
thrown by programmatically when we attempt to perform any operation on a *thread* but it is incompatible with the current thread state.

---

4. IndexOutOfBoundsException: This exception class is thrown by JVM when an *array* or string is going out of the specified index. It has two further subclasses:

- ArrayIndexOutOfBoundsException
- StringIndexOutOfBoundsException

ArrayIndexOutOfBoundsException: ArrayIndexOutOfBoundsException exception is thrown when an array element is accessed out of the index.

StringIndexOutOfBoundsException: StringIndexOutOfBoundsException exception is thrown when a String or *StringBuffer* element is accessed out of the index.

---

5. NullPointerException: NullPointerException is a runtime exception that is thrown by JVM when
we attempt to use null instead of an object. That is, it is thrown when the reference is null.

6. ArrayStoreException: This exception occurs when we attempt to store any value in an array
which is not of array type. For example, suppose, an array is of integer type but we are trying to store a value of an element of another type.

---

7. IllegalStateException: The IllegalStateException exception is thrown by programmatically when
the runtime environment is not in an appropriate state for calling any method.

8. IllegalMonitorStateException: This exception is thrown when a thread does not have the right to monitor an object and tries to access wait(), notify(), and notifyAll() methods of the object.

9. NegativeArraySizeException: The NegativeArraySizeException exception is thrown when an array is created with a negative size.

List of Checked Exceptions in Java

1. ClassNotFoundException: The ClassNotFoundException is a kind of checked exception that is thrown when we attempt to use a class that does not exist.

Checked exceptions are those exceptions that are checked by the Java compiler itself.

2. FileNotFoundException: The FileNotFoundException is a checked exception that is thrown when we attempt to access a non-existing file.

3. InterruptedException: InterruptedException is a checked exception that is thrown when a
thread is in sleeping or waiting state and another thread attempt to interrupt it.

4. InstantiationException: This exception is also a checked exception that is thrown when we try to
create an object of abstract class or interface. That is, InstantiationException exception occurs when an abstract class or interface is instantiated.

5. IllegalAccessException: The IllegalAccessException is a checked exception and it is thrown when
a method is called in another method or class but the calling method or class does not have permission to access that method.

6. CloneNotSupportedException: This checked exception is thrown when we try to clone an object without implementing the cloneable interface.

7. NoSuchFieldException: This is a checked exception that is thrown when an unknown variable is used in a program.

8. NoSuchMethodException: This checked exception is thrown when the undefined method is used in a program.

Custom Exceptions:

Custom exceptions are those exceptions that are created by users or programmers according to their own needs. The custom exceptions are also called user-defined exceptions that are created by extending the exception class.

Checked Exceptions in Java:

Checked exceptions are those exceptions that are checked by the java compiler itself at compilation time and are not under runtime exception class hierarchy.

List of Checked Exceptions in Java

- ClassNotFoundException
- InterruptedException
- InstantiationException
- IOException
- SQLException
- IllegalAccessException
- FileNotFoundException, etc

`InterruptedException`

```
package exceptionHandling;
public class CheckedExceptionEx1
{
public static void main(String[] args)
{
 System.out.println("Hello Java");
 Thread.sleep(1000); // Here, main thread paused for a specified amount of time. // Com
pilation error.
 }
}
```

`Handle Checked Exception`

```
public class CheckedExceptionEx1
{
public static void main(String[] args)
{
   System.out.println("Hello Java");
// Apply the try-catch block to handle checked exception.
   try
   {
     Thread.sleep(1000);
   }
   catch(InterruptedException e)
   {
     e.printStackTrace();
    }
 }
}
```

`FileNotFoundException`

```
package exceptionHandling;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
public class CheckedExceptionEx
{
public static void main(String[] args) throws FileNotFoundException
{
// Creating a file object.
   File file = new File("not_existing_file.txt");

// Create an object of FileInputStream and pass reference variable file to it.
   FileInputStream stream = new FileInputStream(file);
 }
}
```

Unchecked Exceptions (Runtime Exceptions) in Java

Unchecked exceptions in Java are those exceptions that are checked by JVM, not by java compiler. They occur during the runtime of a program.

All exceptions under the runtime exception class are called unchecked exceptions or runtime exceptions in Java.

List of Unchecked Exceptions in Java

Some important examples of runtime exceptions are given below:

- ArithmeticException

- ClassCastException
- NullPointerException
- ArrayIndexOutOfBoundsException
- NegativeArraySizeException
- ArrayStoreException
- IllegalThreadStateException
- SecurityException, etc.

```
public class UncheckedExceptionEx1 {
public static void main(String[ ] args)
{
 int x = 10;
 int y = 0;
 int z = x/y; // runtime exception occurs.
 System.out.println(z);
 }
}
```

```
public class HandleRuntimeException {
public static void main(String[] args)
{
  int x = 10;
  int y = 0;
// Apply try-catch block to handle runtime exception.
  try {
    int z = x/y; // runtime exception.
    System.out.println(z);
  } catch(ArithmeticException ae)
    {
       System.out.println("A number cannot be divided by zero");
    }
  }
}
```

ArrayIndexOutOfBoundsException

```
public class UncheckedExceptionEx2 {
public static void main(String[] args)
{
   int x[ ] = {1, 2, 3, 4};
/* Here, an array contains only 4 elements, but we will try to * display the value of 6
th element.
   It should throw ArrayIndexOutOfBoundsException */
   System.out.println(x[6]);
```

```
 }
}
```

```
public class UncheckedExceptionEx2 {
public static void main(String[] args)
{
 try
 {
   int x[ ] = {1, 2, 3, 4};
   System.out.println(x[6]);
 }
 catch(ArrayIndexOutOfBoundsException e)
 {
   System.out.println("Specified index does not exist. " + "Please correct the erro
r.");
  }
 }
}
```