Exceptions Hierarchy in Java

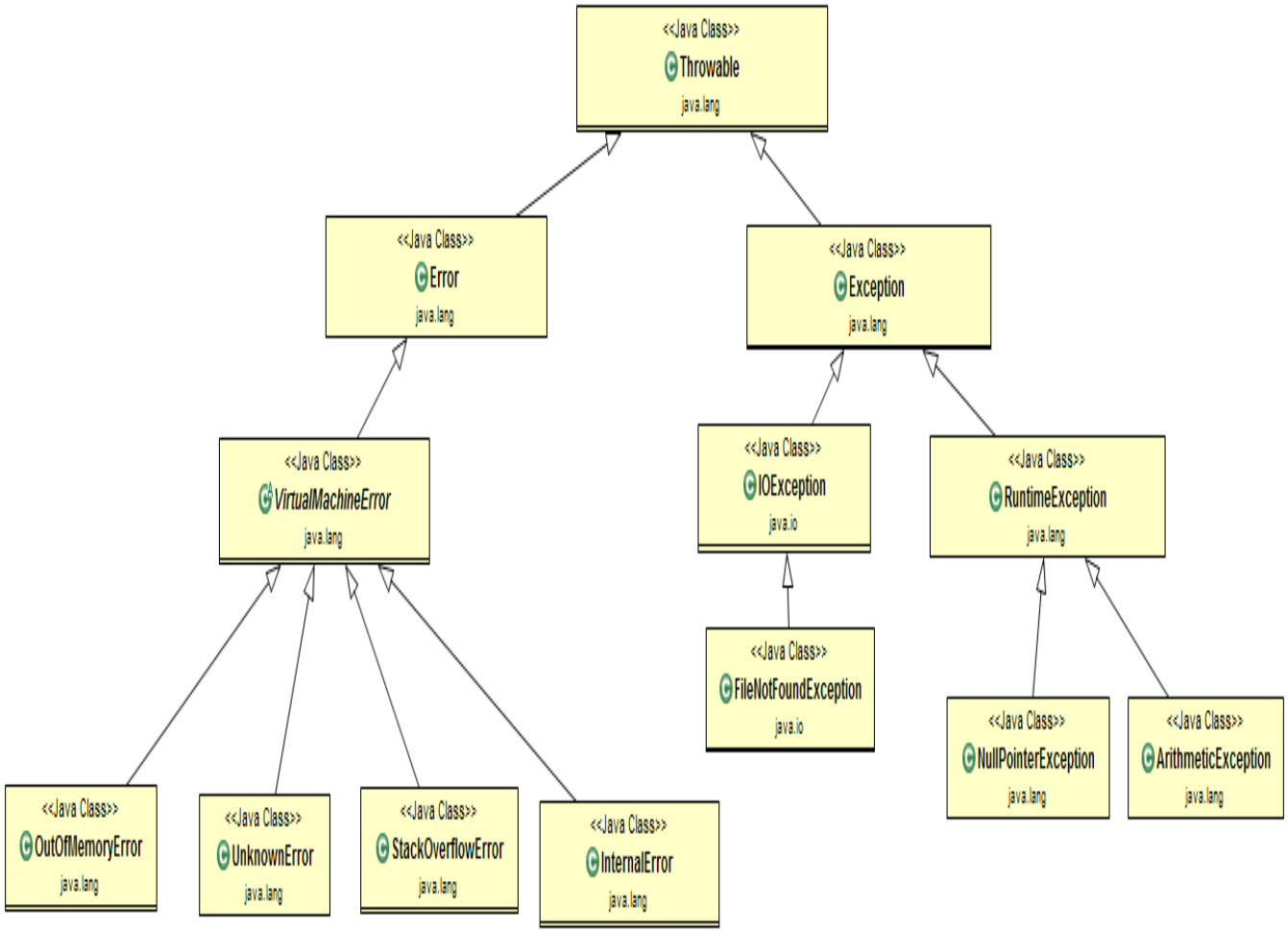
In this article, we will learn the exception class hierarchy in `java.lang` package.

At the top of the exceptions hierarchy is the **Throwable** class. Every class that can be used for exception handling in Java directly or indirectly inherits from this class. The Throwable class is divided into two major subclasses:

- Error
- Exception.

Exceptions Hierarchy in Java

The figure below illustrates the class hierarchy of the **Throwable** class and its most significant subclasses.

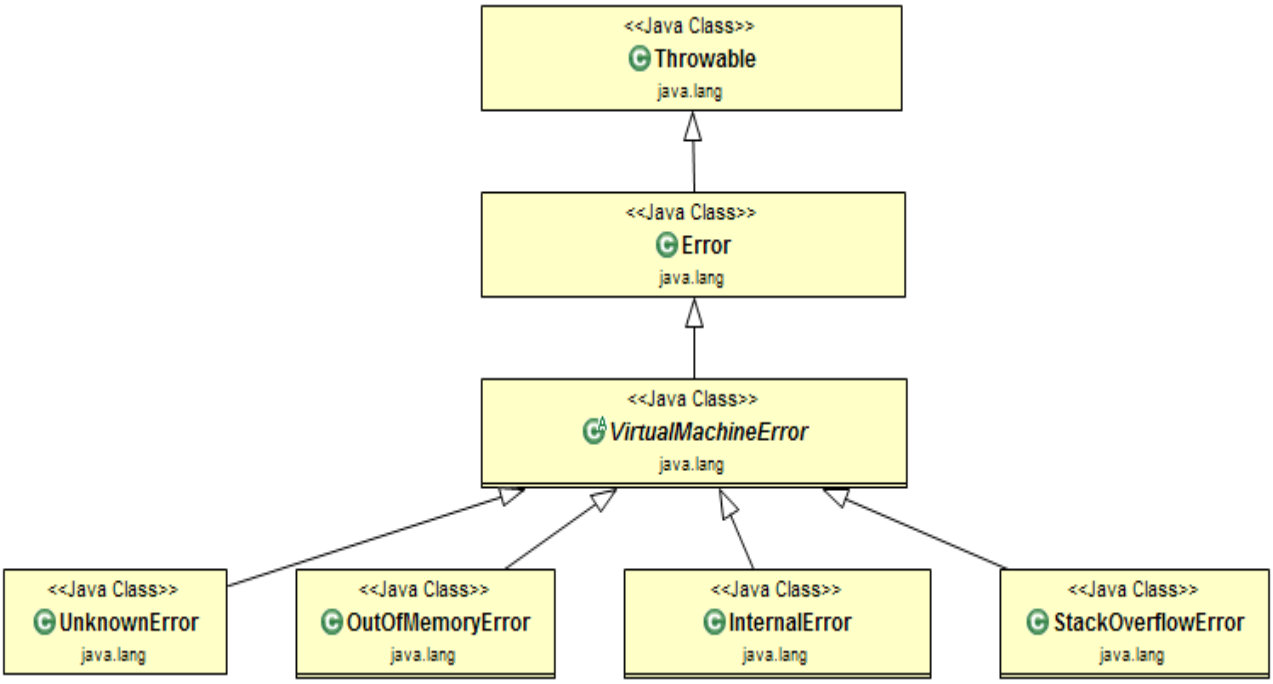


1. Error Class

The `Error` class and its subclasses represent abnormal conditions that should generally not be caught in your application. These indicate serious system problems, often JVM related, which an application might not be able to handle.

Hierarchy of Error Class

The figure below illustrates the class hierarchy of Error Class:



Learn more about below Java built-in errors:

- **OutOfMemoryError**
- **StackOverflowError**
- **NoClassDefFoundError**

2. Exception

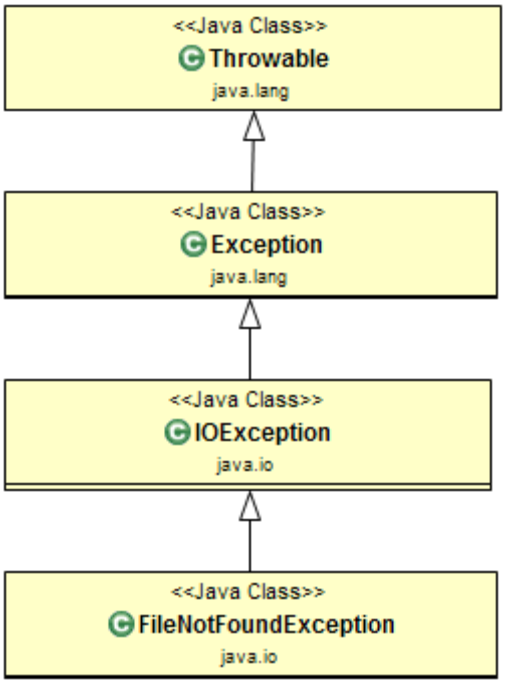
Checked Exceptions

These are exceptions that a method can throw but must either catch them or declare them using the `throws` keyword. They are direct subclasses of the `Exception` class, except `RuntimeException`.

Java built-in checked exceptions:

- **FileNotFoundException**
- **EOFException**
- **ClassNotFoundException**
- **SQLException**
- **NoSuchMethodException**
- **InterruptedException**

The figure below illustrates the class hierarchy of the Exception Class:



Exception Class Example: In this example, *`FileReader`* class try to read a file from an invalid location will throw *`FileNotFoundException`* exception.

```
public class FileNotFoundExceptionExample {
    public static void main(String[] args) {

        BufferedReader reader = null;
        try {
            reader = new BufferedReader(new FileReader(new
File("/invalid/file/location")));
        } catch (FileNotFoundException e) {
            System.err.println("FileNotFoundException caught!");
        }
    }
```

```
}
```

Output:

```
FileNotFoundException caught!
```

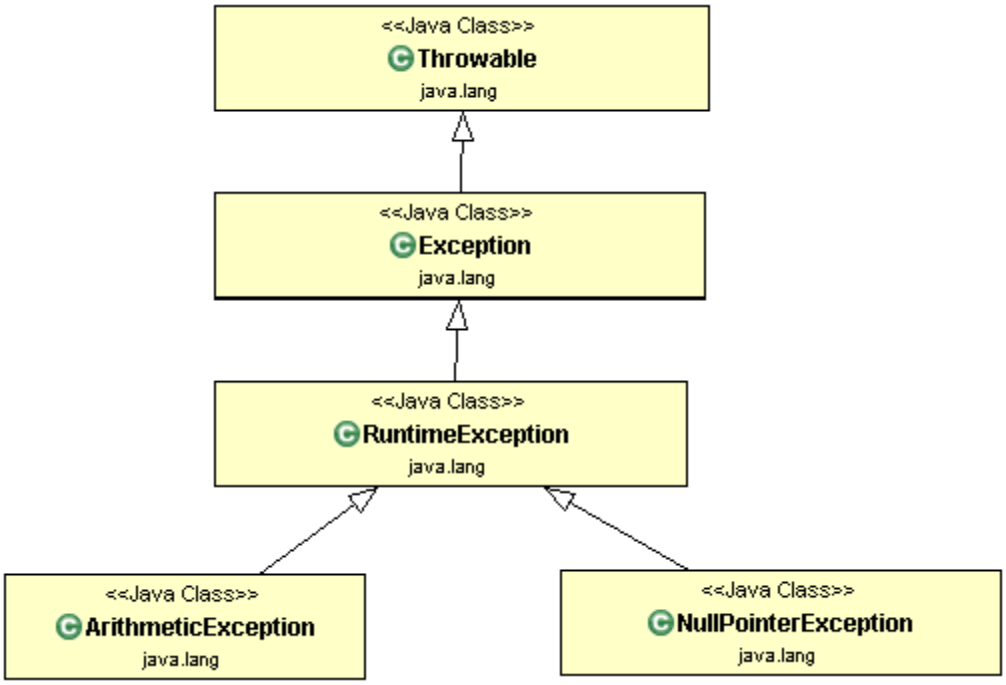
RuntimeException (Unchecked Exceptions)

These exceptions are not checked at compile-time. They're a direct subclass of the *RuntimeException* class.

Java built-in unchecked exceptions:

- **NullPointerException**
- **ArrayIndexOutOfBoundsException**
- **StringIndexOutOfBoundsException**
- **ArithmeticException**
- **IllegalArgumentException**
- **NumberFormatException**
- **IllegalStateException**
- **ClassCastException**

The figure below illustrates the class hierarchy of the *RuntimeException* Class:



RuntimeException Example: In the below example, the Person class object is not created using the new keyword but it is just declared with a null value. Now we are trying to access the personName field value on the null reference so JVM will throw a *NullPointerException* exception here.

```
public class NullPointerExceptionExample {

    public static void main(String[] args) {

        Person personObj = null;

        try {
            String name = personObj.personName; // Accessing the field of a null
object
            personObj.personName = "Jon Doe"; // Modifying the field of a null object
        } catch (NullPointerException e) {
            System.err.println("NullPointerException caught!");
        }

    }
}

class Person {

    public String personName;

    public String getPersonName() {
        return personName;
    }
```

```
    public void setPersonName(String personName) {
        this.personName = personName;
    }

}
```

Output:

```
NullPointerException caught!
```