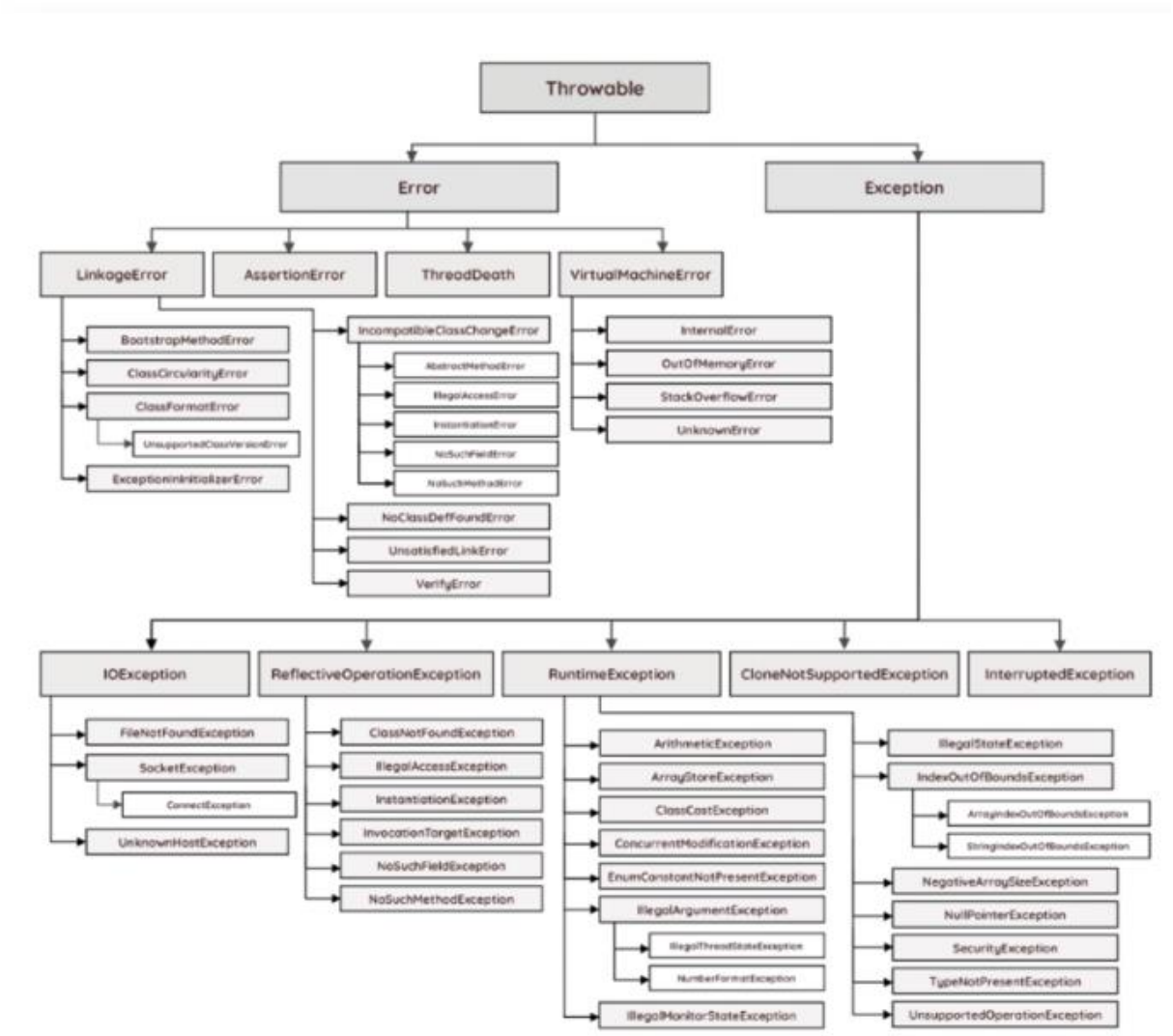
# Checked vs Unchecked Exceptions in Java

Java has two types of exceptions – checked and unchecked.

In Java, all errors and exceptions are of type with Throwable class.

When an error occurs within a method, the *method creates an object* and hands it off to the runtime system. This object is called the exception object.




## Checked Exceptions

All exceptions **other than Runtime Exceptions** are known as Checked exceptions as the compiler checks them during compilation.
These are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using the ***throws* keyword.**

```
private static void checkedExceptionWithThrows() throws FileNotFoundException {
    File file = new File("not_existing_file.txt");
    FileInputStream stream = new FileInputStream(file);
}
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;

public class CheckedException {
    public void readFile() throws FileNotFoundException {
        String fileName = "file does not exist";
        File file = new File(fileName);
        FileInputStream stream = new FileInputStream(file);
    }
}
```

```
import java.io.File;
import java.io.FileInputStream; import java.io.FileNotFoundException;

public class CheckedException {
    public void readFile() {
        String fileName = "file does not exist";
        File file = new File(fileName);
        try {
            FileInputStream stream = new FileInputStream(file);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

## Unchecked Exceptions

If a program throws an unchecked exception, it reflects some error inside the program logic.

```java
import java.util.ArrayList;
import java.util.List;

public class IndexOutOfBounds {
    public static void main(String[] args) {
        List<String> lst = new ArrayList<>();
        lst.add("item-1");
        lst.add("item-2");
        lst.add("item-3");
        var result = lst.get(lst.size());
    }
}
```

These exceptions are not checked at compiletime so compiler does not check whether the programmer has handled them or not but it's the responsibility of the programmer to handle these exceptions and provide a safe exit.

For example, if we divide a number by 0, Java will throw ***ArithmeticException***.

Some common unchecked exceptions in Java are NullPointerException,

ArrayIndexOutOfBoundException and IllegalArgumentException.

" Java does not verify unchecked exceptions at compile time. Furthermore, we don't have to declare unchecked exceptions in a method with the *throws* keyword. "

```java
private static void divideByZero() {
    int numerator = 1;
    int denominator = 0;
    int result = numerator / denominator;
}
```

" The *Exception* class is the superclass of checked exceptions, so we can create a custom checked exception by extending *Exception*: "

```java
public class IncorrectFileNameException extends Exception {
    public IncorrectFileNameException(String errorMessage) {
        super(errorMessage);
    }
}
```

**References :**

https://www.baeldung.com/java-checked-unchecked-exceptions

https://howtodoinjava.com/java/exception-handling/checked-vs-unchecked-exceptions-in-java/

https://bvrithyderabad.edu.in/wp-content/uploads/2020/03/Exception-handling.pdf

Şeyma Tezcan