Learning Consolidation

Implement

Exception Handling

Learning Objective

- Define errors in coding
- Introduction to Exception
- Describe and list the types of exceptions
- Define Unchecked Exception
- Use try, catch and finally blocks
- Explain try with multiple catch

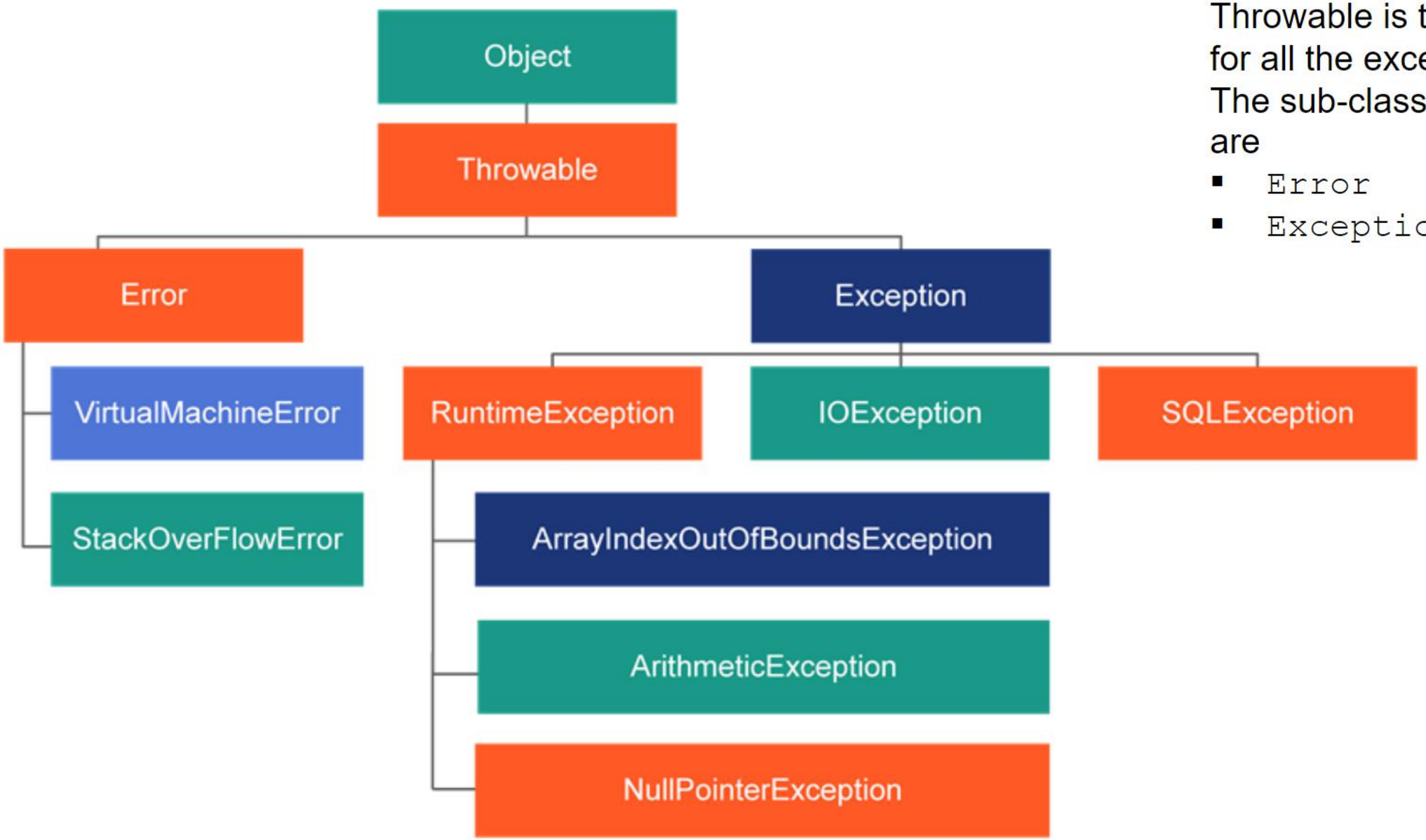
What Are Exceptions?

- An exception is an unwanted event that occurs during the execution of a program and disrupts the normal flow of the program.
- An Exception describes an exceptional condition that occurs in a piece of code.
- An exception can happen either during the compilation or execution phases.

Exception Handling in Java

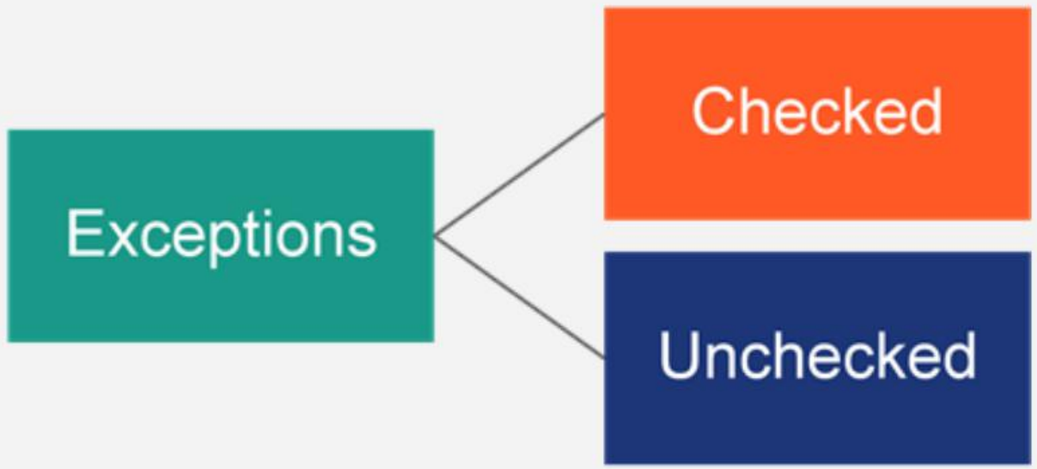
- Java provides a rich set of classes for exception handling.
- All Java programs that generate an exception will throw an object of the parent class of the exception or its sub-classes.
- The parent class is `Throwable`, and there are multiple subclasses like `Exception`, `Error`, etc. It resides in `java.lang` package.

The Exception Class Hierarchy



Throwable is the super class for all the exceptions.
The sub-classes of Throwable are

- `Error`
- `Exception`



Types of Exceptions

- Exceptions must be handled by the programmer.
- There are two types of exceptions in Java:
  - Checked or Compile-time exceptions like `IOException`, `SQLException`, etc.
  - Unchecked or Runtime exceptions like `NullPointerException`, `ArrayIndexOutOfBoundException`, etc.

Note: Checked exceptions will be discussed in the next Sprint.

Unchecked Exceptions

- Runtime exceptions or unchecked exceptions occur due to bad programming.
- For example, while trying to retrieve an element from an array, the length of the array must be checked first before trying to retrieve the element; otherwise, it might throw an `ArrayIndexOutOfBoundException` at runtime or execution time.
- Runtime exceptions can be avoided with better programming.

Exception Handlers in Java Using `try-catch-finally`

- Code in the `finally` block in Java is executed whether an exception occurs or not.
- A finally block follows a try or a catch block.

```
try{
    //code that can throw exception
}catch (Exception exception) {
    //Block of code to handle exception
}finally {
    //This line of code will always execute
}
```

Code Without Exception Handling

```
int number[] = new int[8];
//Logical error in for loop
for (int i= 0; i<=number.length; i++){
        System.out.print(number[i]+ " ");
}
//This line will never get executed
System.out.println("This line will not get executed");
```

- When using the for loop to iterate over the number array,a logical error is made.

`i <= number.length`

- This will result in a runtime `ArrayIndexOutOfBoundException`
- Once the exception occurs, the rest of the code will not get executed, including the last line that is present outside the `for` loop.

```
0 0 0 0 0 0 0 0 Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
```