Exception Handling

Instructor:
Nasrin Khodapanah

Exception Handling

An exception is a problem that arises during the execution of a program. An exception can occur for many different reasons, including the following:

- ✓ A user has entered invalid data.
- ✓ A file that needs to be opened cannot be found.
- ✓ A network connection has been lost in the middle
- ✓ of communications, or the JVM has run out of memory.

Exception Handling

Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

To understand how exception handling works in Java, you need to understand the three categories of exceptions:

Exception Handling

Checked exceptions (compile-time exception):

A checked exception is an exception that is typically a user error or a problem that cannot be foreseen by the programmer. For example, if a file is to be opened, but the file cannot be found, an exception occurs. These exceptions cannot simply be ignored at the time of compilation.

Developers must handle this type of error. If not, then compiler will give error.

Exception Handling

Unchecked exceptions (Runtime exceptions):

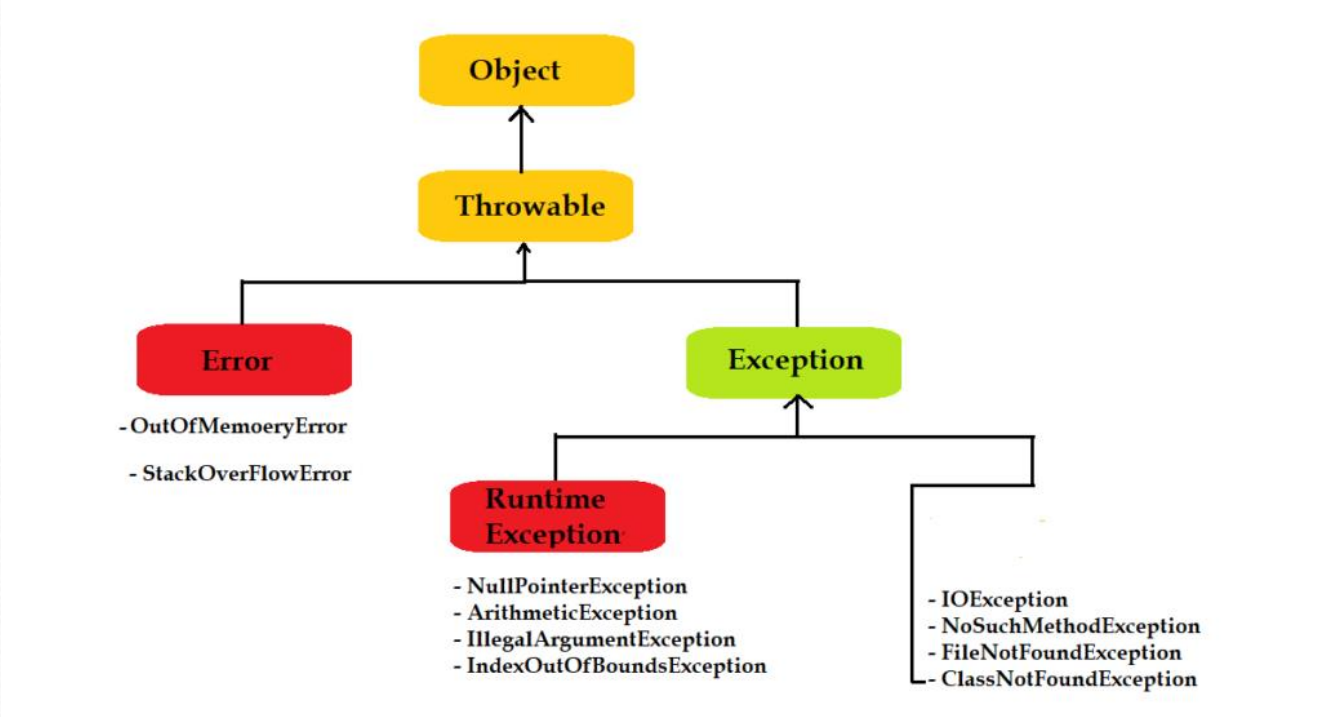
A runtime exception is an exception that occurs that probably could have been avoided by the programmer. As opposed to checked exceptions, runtime exceptions are ignored at the time of compilation.

Exception Handling

Errors:

These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in your code because you can rarely do anything about an error. For example, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.

Exception Handling



Exception Handling

Catching Exceptions

A method catches an exception using a combination of the **try** and **catch** keywords.

A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch looks like the following:

Exception Handling

Catching Exceptions

```
try {
        //Protected code
                }
    catch(ExceptionName e1) {
        //Catch block
}
```

Exception Handling

The finally Keyword

The finally keyword is used to create a block of code that follows a try block. A finally block of code always executes, whether or not an exception has occurred.
Using a finally block allows you to run any cleanup-type statements that you want to execute, no matter what happens in the protected code.

Exception Handling

The finally Keyword:

```
try {
        //Protected code
}
catch(ExceptionName e1) {
        //Catch block
}
finally {
        //The finally block always executes.
}
```

Exception Handling

Warning:

- ✓ A catch clause cannot exist without a try statement.
- ✓ It is not compulsory to have finally clauses when ever a try/catch block is present.
- ✓ The try block cannot be present without either catch clause or finally clause.
- ✓ Any code cannot be present in between the try, catch, finally blocks.
- ✓ A try block can be followed by one or more catch blocks.