Object Oriented Programming with Java
Class: Exception handling

Stelios Sotiriadis

Agenda

- Introduction to Exception Handling
  - Errors vs Exceptions
  - Run-time errors
  - Exception handling in Java
  - Types of Exceptions
  - Common examples of Exceptions in Java

Introduction to Errors and Exceptions

What is a run-time error?

- A runtime error occurs while a program is running or when you first attempt to start the application.
- What can cause a run-time error?
  - There's a bug in the software.
  - Memory or another system resource is in short supply.
  - Incorrect input, e.g. you enter a number in a String variable

Exceptions

- **Definition:**
  - An *exception* is an abnormal program behaviour or an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.

Quiz 1

- Can you think of an example of an exception?

* Divide a number by zero

* Access an array index that is negative, greater than, or equal to the length of the array

* Invalid user input

Errors versus Exceptions

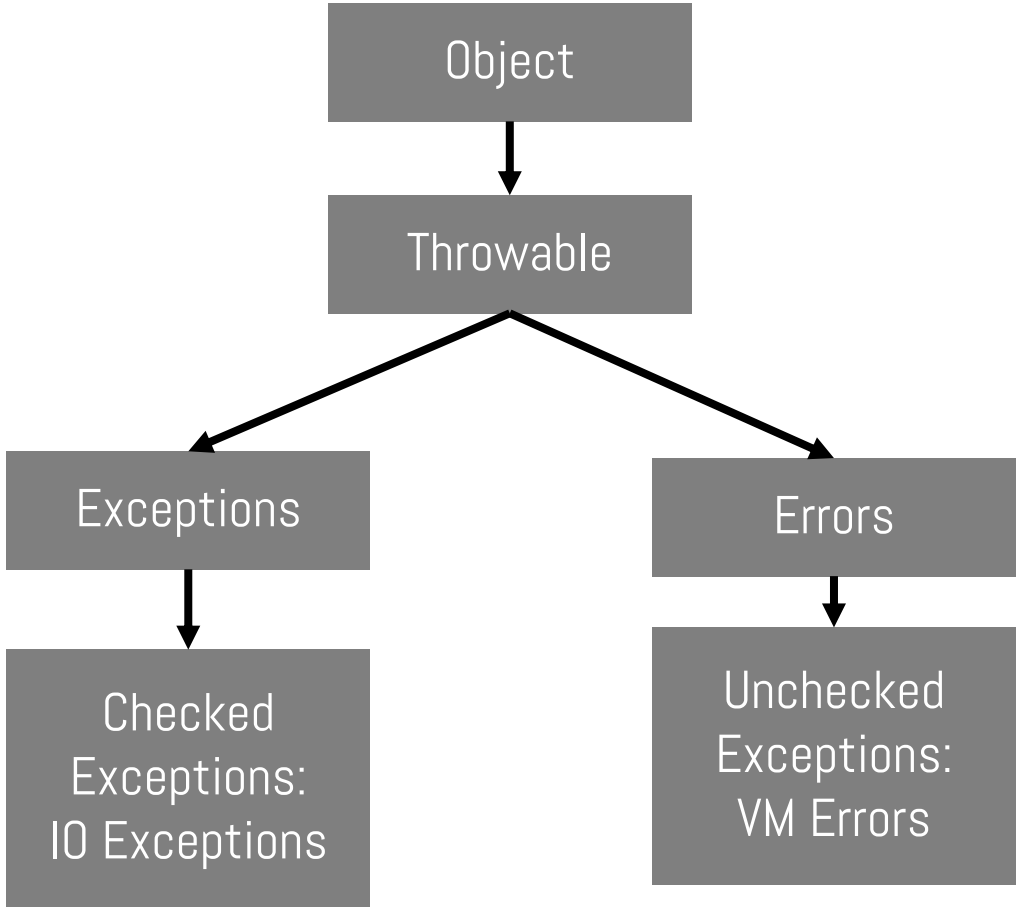
- **Errors** represent **irrecoverable** conditions
  - For example, if the Java virtual machine (JVM) running out of memory, and there are memory leaks
  - Errors are usually beyond the control of the programmer and we should not try to handle errors.
- **Exception:** Exception indicates conditions that a reasonable application might try to catch.

What is an Exception Handling?

- **Definition:**
  - Exception handling ensures that the flow of the program doesn't break when an exception occurs.
- Exception handling in Java:
  - A way to handle run-time errors so that the regular flow of the application can be preserved.
  - Java Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException and others.

Exception Handling in Java

- All exception and error types are subclasses of class **Throwable**, which is the base class of the hierarchy.
- Throwable is **the superclass of all errors and exceptions in the Java language.**



Built-in user Exceptions

- Built-in exceptions are the exceptions that are available in Java.
- These exceptions are suitable to explain certain error situations.
  - **Checked Exceptions:** Compile-time exceptions that are checked at compile-time by the compiler.
  - **Unchecked Exceptions:** The compiler will not check these exceptions at compile time.

Quiz 2

- Can you think of an example of:

a. A checked exception?

You try to open a file that does not exist (FileNotFoundException)

b. An unchecked exception?

You try to open a access an array index that does not exist (ArrayIndexOutOfBoundsException)

Introduction to Exception Handling with Java

What is the output (1/4)?

```
class ArithmeticException
{
    public static void main(String args[])
    {
        try {
            int a = 30, b = 0;
            int c = a/b;
            System.out.println ("Result = " + c);
        }
        catch(ArithmeticException e) {
            System.out.println ("Can't divide a number by 0");
        }
    }
}
```

Output:

Can't divide a number by 0

What is the output (2/4)?

```
class NullPointer
{
    public static void main(String args[])
    {
        try {
            String a = null; //null value
            System.out.println(a.charAt(0));
        }
        catch(NullPointerException e) {
            System.out.println("NullPointerException");
        }
    }
}
```

Output:

NullPointerException

What is the output (3/4)?

```
class StringIndexOutOfBound
{
    public static void main(String args[])
    {
        try {
            String a = "Hello World"; // length is 11
            char c = a.charAt(11); // accessing 11th element
            System.out.println(c);
        }
        catch(StringIndexOutOfBoundsException e) {
            System.out.println("StringIndexOutOfBoundsException");
        }
    }
}
```

last_index = length - 1

Output:

StringIndexOutOfBoundsException

What is the output (4/4)?

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
class File_notFound_Demo {
    public static void main(String args[]) {
        try {
            // Following file does not exist
            File file = new File("E://file.txt");
            FileReader fr = new FileReader(file);
        } catch (FileNotFoundException e) {
        System.out.println("File does not exist");
        }
    }
}
```

Output:

```
File does not exist
```

Quiz 3

- Can you fill the spaces?

```
class ArrayIndexOutOfBound
{
        public static void main(String args[]) {
                try {
                        int a[] = new int[5];
                        a[6] = 9; // Accessing 7th element of the array
                }
                 catch (ArrayIndexOutOfBoundsException e){
                        System.out.println (” Array Index is Out Of Bounds ");
                }
        }
}
```

Summary

- What can cause a run-time error?
  - There's a bug in the software.
- Exception indicates conditions that a reasonable application might try to catch.
- **Checked Exceptions:** Compile-time exceptions that are checked at compile-time by the compiler.
- **Unchecked Exceptions:** The compiler will not check these exceptions at compile time.

End of Class!

- Take home:
  - What is an Exception?
  - What is an error?
  - What are the Exceptions types in Java?
  - What is checked and unchecked Exception?
  - Can you develop a Java script for a build-in Exception such as `ArrayIndexOutOfBoundsException`?

Q&A

- Any questions?

Questions?

- Useful resources to check:
- Java documentation on Exception Handling:
  - https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html
- To explore at home:
  - Advantages of Exceptions:
    - https://docs.oracle.com/javase/tutorial/essential/exceptions/advantages.html