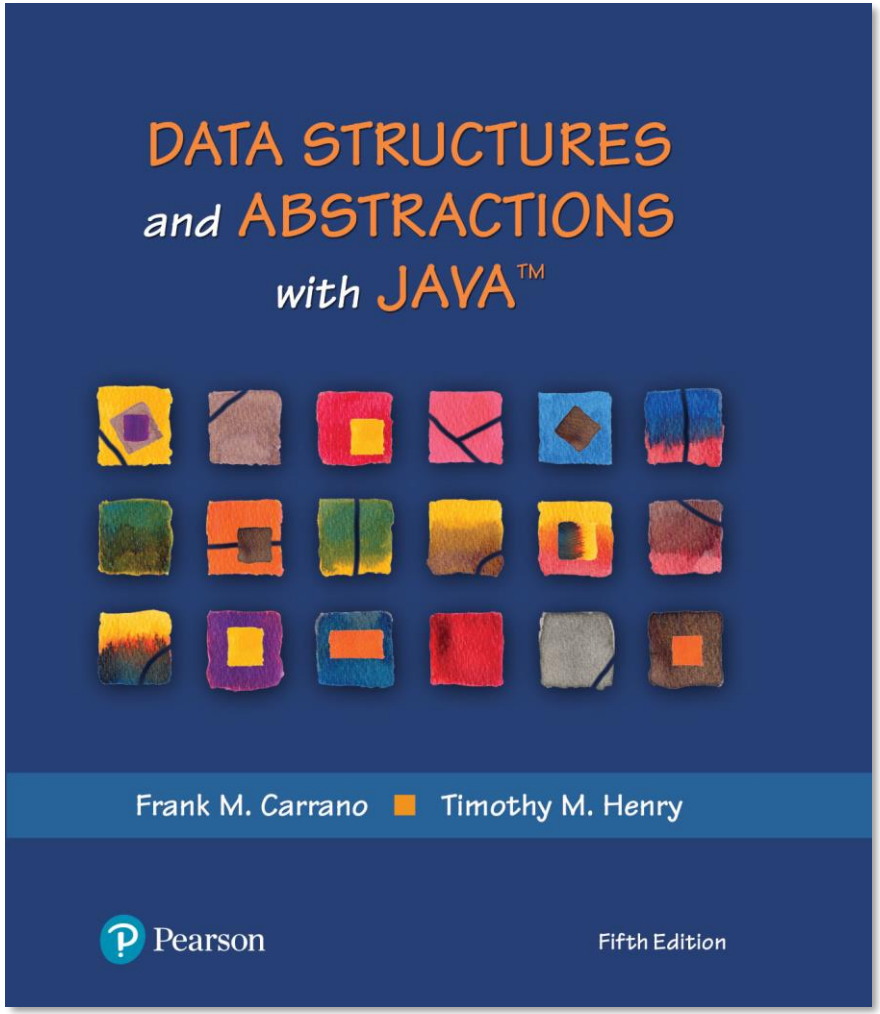
# Data Structures and Abstractions with Java™

5th Edition




## Java Interlude 2

# Exceptions

# The Basics

- Method creates an exception object
  - We say “throws an exception”
- Signal to program
  - Unexpected has happened
- Handle the exception
  - Detect and react

# The Basics

- Checked exceptions in the Java Class Library
  - **`ClassNotFoundException`**
  - **`FileNotFoundException`**
  - **`IOException`**
  - **`NoSuchMethodException`**
  - **`WriteAbortedException`**

# The Basics

- Runtime exceptions in the Java Class Library
  - **`ArithmeticException`**
  - **`ArrayIndexOutOfBoundsException`**
  - **`ClassCastException`**
  - **`IllegalArgumentException`**
  - **`IllegalStateException`**
  - **`IndexOutOfBoundsException`**
  - **`NoSuchElementException`**
  - **`NullPointerException`**
  - **`StringIndexOutOfBoundsException`**
  - **`UnsupportedOperationException`**

# Java Class Exception and Error Hierarchy

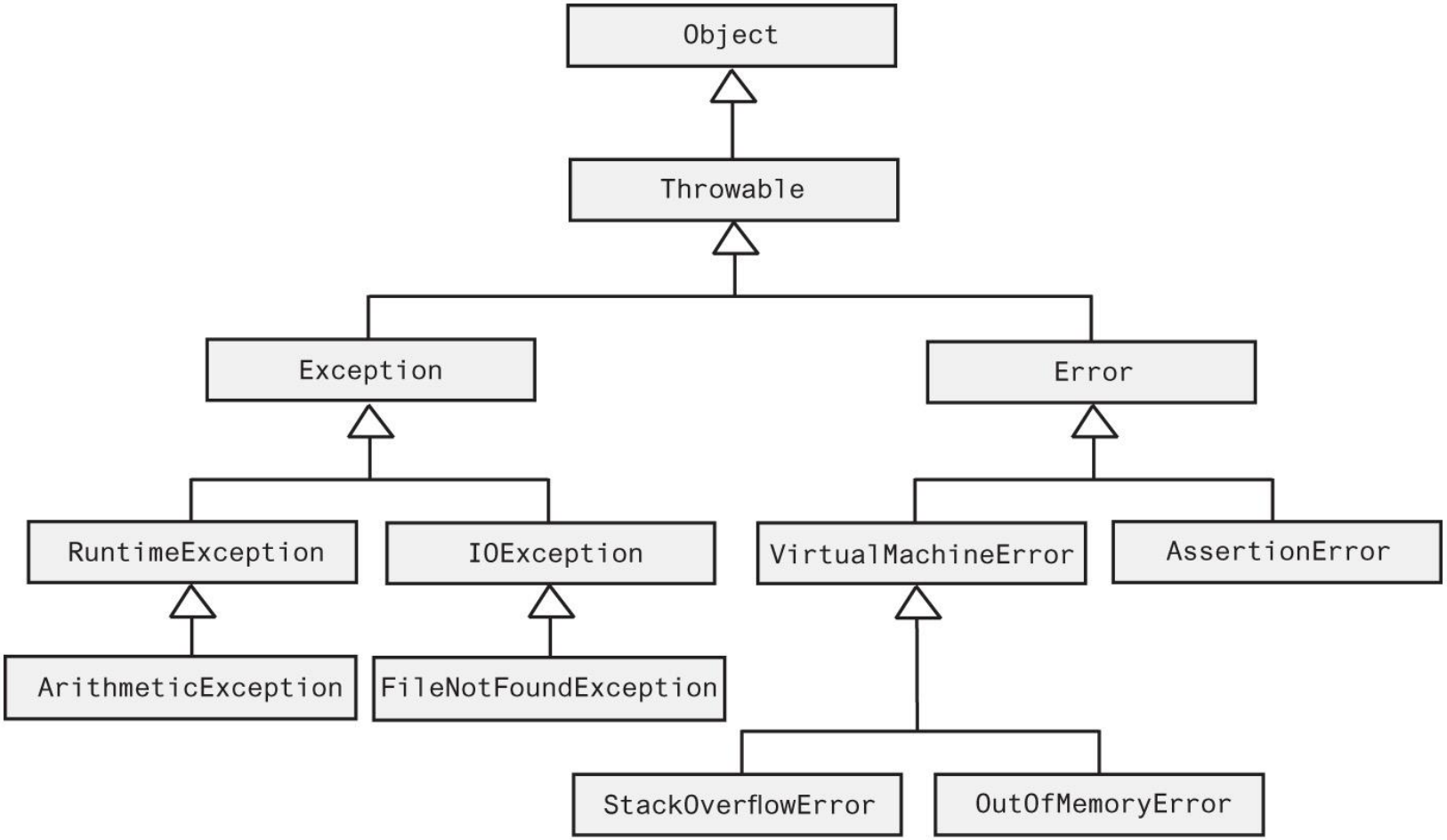



**FIGURE J2-1 The hierarchy of some standard exception and error classes**

# Handling an Exception

- Postpone handling: The throws clause
  - If programmer not sure what action is best for a client when an exception occurs
  - Leave the handling of the exception to the method's client
- Method that can cause but does not handle checked exception must declare in its header

```
public String readString(. . .) throws IOException
```

# Handle It Now: The try-catch Blocks

```
try
{
        < Possibly some code >
        anObject.readString(. . .); // Might throw an IOException
        < Possibly some more code >
}
catch (IOException e)
{
        < Code to react to the exception, probably including the following statement: >
        System.out.println(e.getMessage());
}
```

**Code to handle an `IOException` as a result of invoking the method `readString`**

# Multiple `catch` Blocks

```
catch (FileNotFoundException e)
{
      . . .
}
catch (IOException e) // Handle all other IOExceptions
{
      . . .
}
```

## Good order for `catch` blocks

# Throwing an Exception

- A method intentionally throws an exception by executing a throw statement.
- Programmers usually create the object within the throw statement

```
throw new IOException();
```

# Throwing an Exception

- If you can resolve unusual situation in a reasonable manner
  - likely can use a decision statement
- If several resolutions to abnormal occurrence possible, and you want client to choose
  - Throw a checked exception
- If a programmer makes a coding mistake by using your method incorrectly
  - Throw a runtime exception

# End

# Java Interlude 2