

Exception Handling

CS 272 Software Development

Java Exception Handling

- Program can operate even if exceptions occur
 - Warn and continue, or terminate gracefully
- Allows for grouping of types of exceptions
- Separates exception handling logic from normal functionality

<https://docs.oracle.com/javase/tutorial/essential/exceptions/advantages.html>



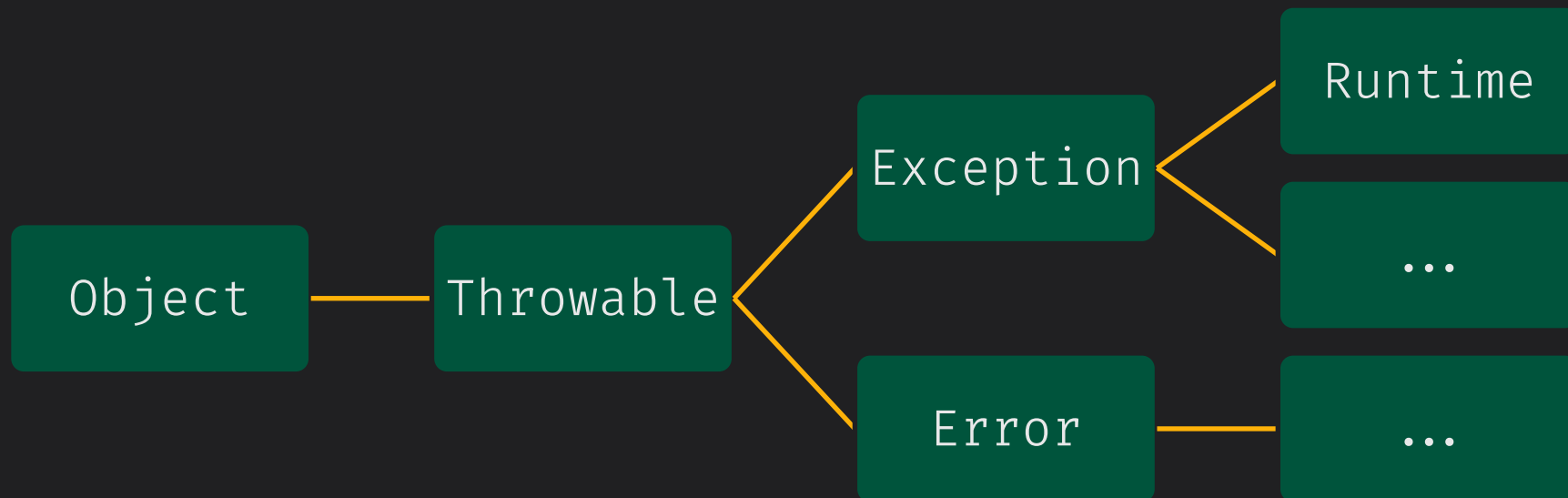
Types of Exceptions

Type	Description	Catch/Throw?
Error	Problem occurred external to application (in the JVM)	<i>Not Usually</i>
Runtime Exception	Programs cannot anticipate or recover from, usually a bug	Optional
Checked Exception	Programs should anticipate and recover from	Required

<https://docs.oracle.com/javase/tutorial/essential/exceptions/catchOrDeclare.html>



Exception Hierarchy



<https://docs.oracle.com/javase/tutorial/essential/exceptions/throwing.html>

Errors

- Indicate a serious problem external to program (in the JVM) occurred
- Usually not addressed by simple programs
- For example, an **IOException** from hard drive failure while reading an open file

<https://docs.oracle.com/javase/tutorial/essential/exceptions/throwing.html>



Runtime Exceptions

- Includes all **Exception** subclasses under **RuntimeException**
- Can be caught or thrown, but not required
- Often indicates code defects
 - e.g. accessing an array out of bounds

<https://docs.oracle.com/javase/tutorial/essential/exceptions/catchOrDeclare.html>



Common Runtime Exceptions

- `ArithmeticException`
- `IndexOutOfBoundsException`
 - `ArrayIndexOutOfBoundsException`
 - `StringIndexOutOfBoundsException`
- `NegativeArraySizeException`
- `NullPointerException`

<https://www.cs.usfca.edu/~cs272/javadoc/api/java.base/java/lang/RuntimeException.html>



Checked Exceptions

- Includes all **Exception** subclasses **except** those under **RuntimeException**
- Should be anticipated by programmer
- Must be handled by application
 - e.g. trying to open an non-existent file

<https://docs.oracle.com/javase/tutorial/essential/exceptions/catchOrDeclare.html>



Common IO Exceptions

- EOFException
- FileNotFoundException
- FileSystemException
- NoSuchFileException
- CharacterCodingException
- MalformedInputException

<https://www.cs.usfca.edu/~cs272/javadoc/api/java.base/java/io/IOException.html>



Handling Checked Exceptions

- Use **try**, **catch**, **finally** blocks to handle
- Use **try**-with-resources blocks
 - Added in Java 7 to auto-close resources
- Use **throws** keyword and **catch** elsewhere
 - Catch where makes sense to handle exception



Catching Exceptions

- Any exception type (even unchecked) can be caught
- Can specify multiple catch blocks for each try
 - Will execute first matching catch block
- Related to inheritance, discussed more later
- Can catch more than one exception per handler
- Can throw exceptions within handlers



API Example

nextInt

```
public int nextInt()
```

Scans the next token of the input as an int.

An invocation of this method of the form `nextInt()` behaves in exactly the same way as the invocation `nextInt(radix)`, where `radix` is the default radix of this scanner.

Returns:

the int scanned from the input

Throws:

`InputMismatchException` - if the next token does not match the *Integer* regular expression, or is out of range

`NoSuchElementException` - if input is exhausted

`IllegalStateException` - if this scanner is closed

[https://www.cs.usfca.edu/~cs272/javadoc/api/java.base/java/util/Scanner.html#nextInt\(\)](https://www.cs.usfca.edu/~cs272/javadoc/api/java.base/java/util/Scanner.html#nextInt())



Questions?

