Video Name: Advanced Exceptions

Topics:

- checked and unchecked exceptions
- exception hierarchy
- custom exceptions

Java Class(es): RestrictedFileTotal

Checked exceptions. Can’t ignore, compiler requires that you handle or propagate (with throws clause). The goal is to create programs which are more reliable and robust. Example: FileNotFoundException

Unchecked exceptions. Compiler doesn’t require you to acknowledge. Often relate to programmer error, such as not allocating space (NullPointerException).

BOTH types of exceptions are thrown by the JVM at runtime. The difference is what happens at compile time - a checked exception that is not acknowledged, either via throws clause or try/catch, generates a compiler error.

Software engineering. Good error handling is a software engineering goal. Checked exceptions aid in this goal.

Custom Exceptions

Exception is a class. Create a custom exception by extending Exception (for a checked exception) or RuntimeException (for an unchecked exception).

Custom exception class will typically have a default constructor and a constructor that takes a string. This corresponds to a common way to throw exceptions: throw new Exception(“meaningful error message”);

    public NegativeNumberException() {}
    public NegativeNumberException(String message) {
        super(message);
    }

Software engineering. Should always have a reason for creating a custom exception. For example, logging exceptions, standardized exception handling for a larger system, etc.

Software engineering. Throw early, catch late. Throw an exception as soon as condition is discovered. Handle the error at the most appropriate time.
Software engineering. Don’t squelch exceptions! **Do not do:**

```java
try {
    FileReader reader = new FileReader(filename);
} catch (FileNotFoundException e) { }
```

The above code is very bad!!