Stream API

Added in Java 8

- Streams are wrappers around a data source
- They allow us to operate with that data source
    - making bulk processing convenient and fast
- A stream supports functional-style operations on stream elements (lambdas)
- A stream does not store data and is not a data structure
- A stream never modifies the underlying data source

Java Stream operations are divided into **intermediate** and **terminal** operations.

Intermediate Operations: return a new stream on which further processing can be done.

Terminal Operations: mark the stream as **consumed**, after which point it can no longer be used further.

Pipeline: consists of a stream source, followed by zero or more intermediate operations, and a terminal operation.

Lazy Evaluation: computation on the source data is only performed when the terminal operation is initiated, and source elements are consumed as needed.

- All intermediate operations are lazy
    - they are not executed until a result of a processing is actually needed.
- Processiong streams lazily allow avoiding examining all data when that's not necessary.
    - (aside) this will become even more important when the input stream is infinite, not just large

Optional: a wrapper class that contains an optional value

- meaning it can either contain an object or it can simply be empty (null)
- useful with functional programming so that we don't always get NullPointerExceptions

N.B.

- Java IO streams (FileInputStream etc.) are not related to the Streams API and have little to do with each other.