java.io.*

- What is a package? How are packages organized in Java? How do you find how to use them?
  - Reading API, reading books, writing code
- In Java there are lots of ways of reading, mostly using *InputStream* and *Reader* abstract classes
  - How do you use an abstract class?
- These classes use the Decorator pattern
  - Reading files? FileReader to BufferedReader
  - How to read System.in line-by-line?
  - See also java.util.Scanner in Java 5

java.util.*

- Contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array)
- From programmer's perspective the *Collection* hierarchy provides data structures
  - Lists, Sets, Maps of elements (and others)
  - Your code must know how to play well with collections

Playing well with Collections

- Every object has an `equals(..)` method, contract?
  - What does this return? How do you implement it, what about apples and oranges?
  - Default behavior? When to over-ride?
- If you override/over-ride equals, see `hashCode()`
  - What does this return? Implementation issues?
  - Good, bad, …?

Loose coupling and collections

- How do you store data in nanoGoogle?
  - What do you store?
  - How do you access it?
  - What are performance issues and trade-offs?
- Difference in storing into a Map versus storing into a NanoCollector?
  - What heuristics are in play?
  - What do you do first?