Object/Class Design
A (good) class can be

- A collection of data and routines that share a cohesive, well-defined responsibility
- A collection of routines that provide a cohesive set of services even if no common data is involved.
Abstract Data Types

- Define a class based around conceptual structures
  - Encapsulation
  - Make interfaces more informative (self-documenting)
  - Easier to reason about correctness

- Treat even simple items as ADTs
  - Good for extensibility

```java
public class NumberGenerator {
    private int nextNumber;

    public int getNext() {
        return nextNumber++;
    }
}
```
hasA vs. isA relationship

Which is a candidate for inheritance?

A) hasA relationship
B) isA relationship
C) both hasA and isA relationships
D) neither hasA and isA relationships
Inheritance can provides 2 things

- **Shared interface:**
  - Public methods
  - Ensure methods have consistent meaning in derived class
    - Liskov Substitution Principle
  
  Dry = *Don’t repeat yourself*

- **Shared implementation**
  - Code in shared super class, not replicated in each derived
  - Could be private data/methods
Inheritance vs. Interfaces

- Inheritance should be a isA relationship
- Interfaces are for capabilities ("mixin"s)
Designing Good Interfaces

- Sufficiently Expressive
- General
- Minimal
To Dos for Wednesday

- Read Ch. 7 (High Quality Routines)