CSSE 220 Day 13

Designing Classes

Check out *DesigningClasses* from SVN

Questions?

What is good object-oriented design?

- It starts with good classes…

Good Classes Typically

- Come from nouns in the problem description
- May…
  - Represent single concepts
    - `Circle`, `Investment`
  - Be abstractions of real-life entities
    - `BankAccount`, `TicTacToeBoard`
  - Be actors
    - `Scanner`, `CircleViewer`
  - Be utilities
    - `Math`

Q1

What Stinks? Bad Class Smells

- Can’t tell what it does from its name
  - `PayCheckProgram`
- Turning a single action into a class
  - `ComputePaycheck`
- Name isn’t a noun
  - `Interpolate`, `Spend`

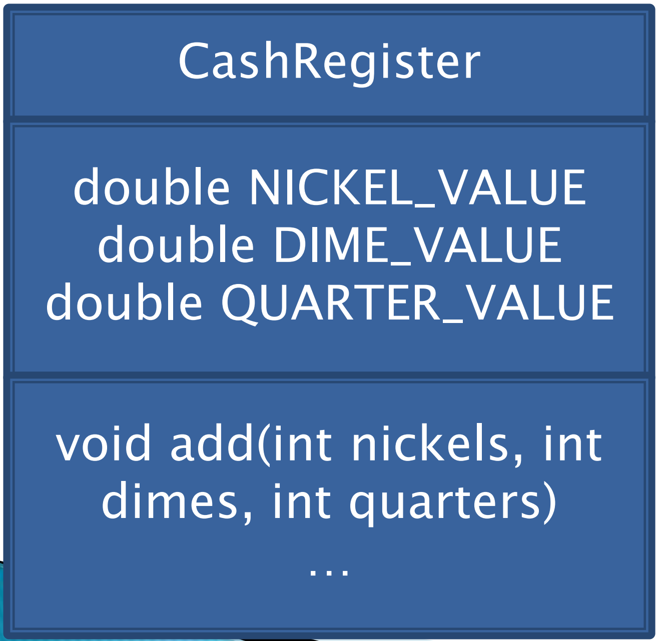
Q2

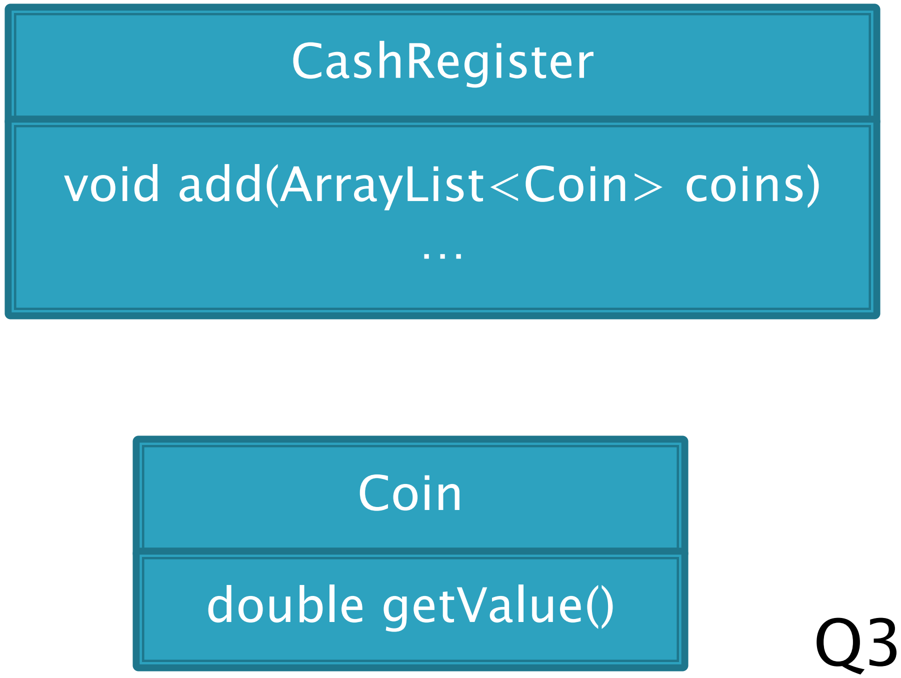
Analyzing Quality of Class Design

- Cohesion
- Coupling

Cohesion

- A class should represent a single concept
- Public methods and constants should be cohesive
- Which is more cohesive?

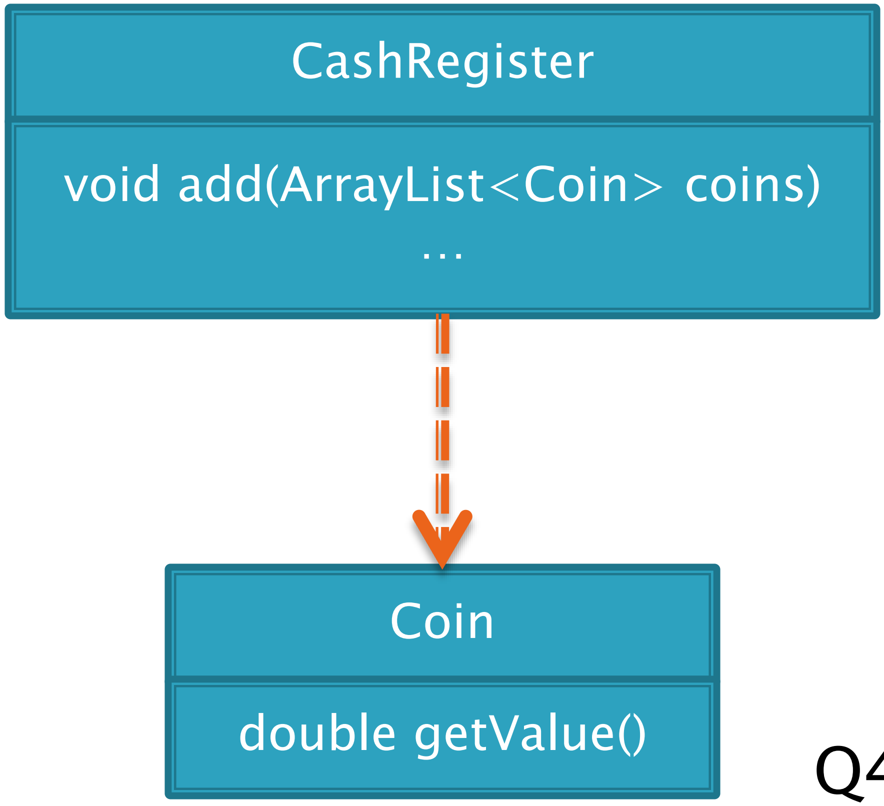




Q3

Dependency Relationship

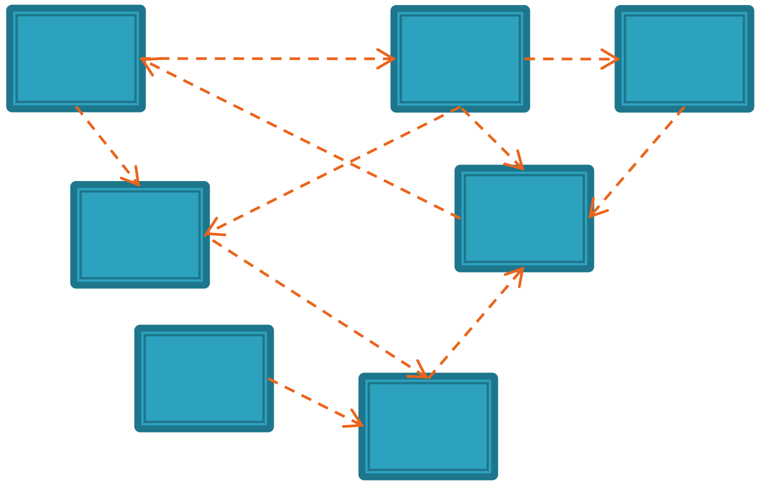
- When one classes requires another class to do its job, the first class depends on the second
- Shown on UML diagrams as:
  - dashed line
  - with open arrowhead

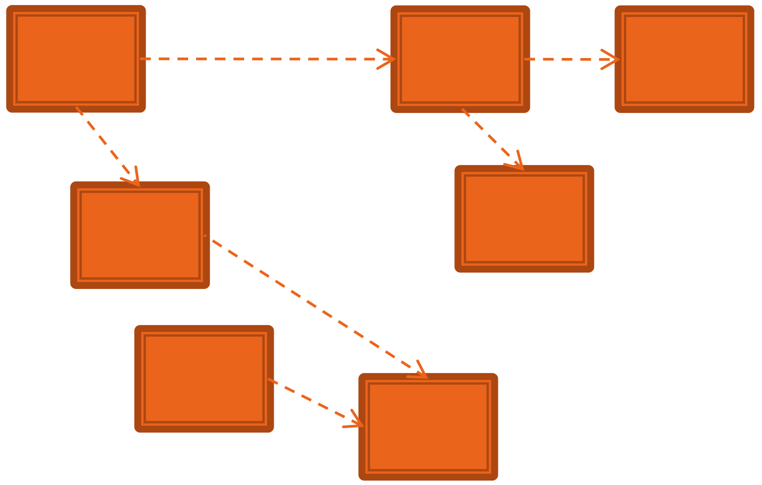


Q4

Coupling

- Lots of dependencies == high coupling
- Few dependencies == low coupling





- Which is better? Why?

Q5

Quality Class Designs

- High cohesion
- Low coupling

Accessors and Mutators Review

- **Accessor method**: accesses information *without changing any*
- **Mutator method**: *modifies* the object on which it is invoked

Q6

Immutable Classes

- Accessor methods are very predictable
  - Easy to reason about!
- Immutable classes:
  - Have only accessor methods
  - No mutators
- Examples: **`String`**, **`Double`**
- Is **`Rectangle`** immutable?

Immutable Class Benefits

- Easier to reason about, less to go wrong
- Can pass around instances “fearlessly”

Q7

Side Effects

- Side effect: any modification of data
- Method side effect: any modification of data *visible* outside the method
  - Mutator methods: side effect on implicit parameter
  - Can also have side effects on other parameters:
    -
      ```
      public void transfer(double amt, Account other)
      {
          this.balance -= amt;
          other.balance += amt;
      }
      ```

Avoid this if you can!

Q8

Class Design Exercise

- See HW12
  Can work in groups of three on initial steps