CSSE 220 Day 10

Some Software Engineering Techniques
(Class Diagrams and Pair Programming)

Designing Classes

- Programs typically begin as abstract ideas
- These ideas form a set of abstract requirements
- We must take these abstract requirements, use piecewise elaboration and refinement until specifications emerge
  - Then models
  - … concrete implementation

Tools of the Trade

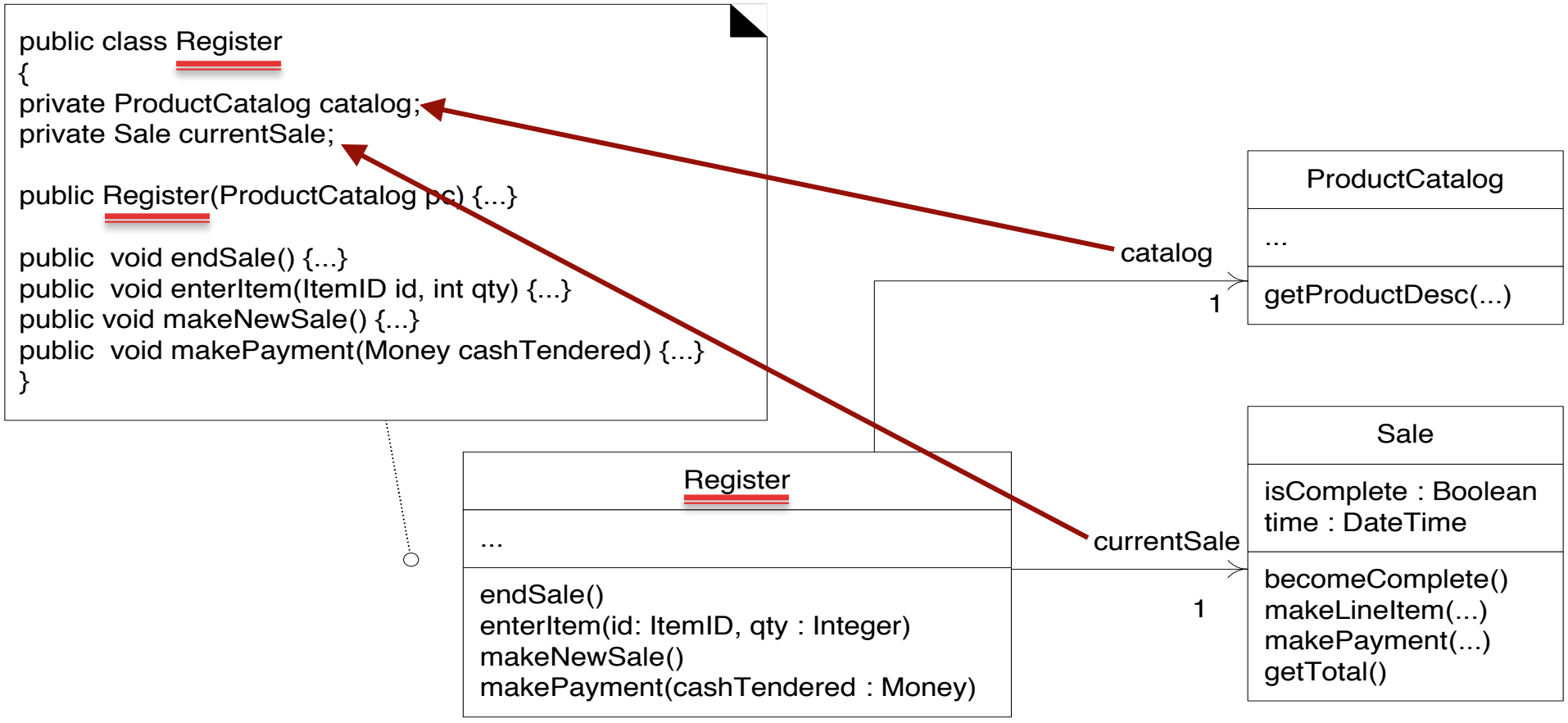
- Class Diagrams (UML)
- UML – Unified Modeling Language
  - Language unspecific
  - provides guidance as to the order of a team's activities
  - specifies what artifacts should be developed
  - directs the tasks of individual developers and the team as a whole
  - offers criteria for monitoring and measuring a project's products and activities

According to UML-Diagrams.org

- The **Unified Modeling Language™** (**UML®**) is a standard visual modeling language intended to be used for
  - modeling business and similar processes,
  - analysis, design, and implementation of software-based systems

UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.

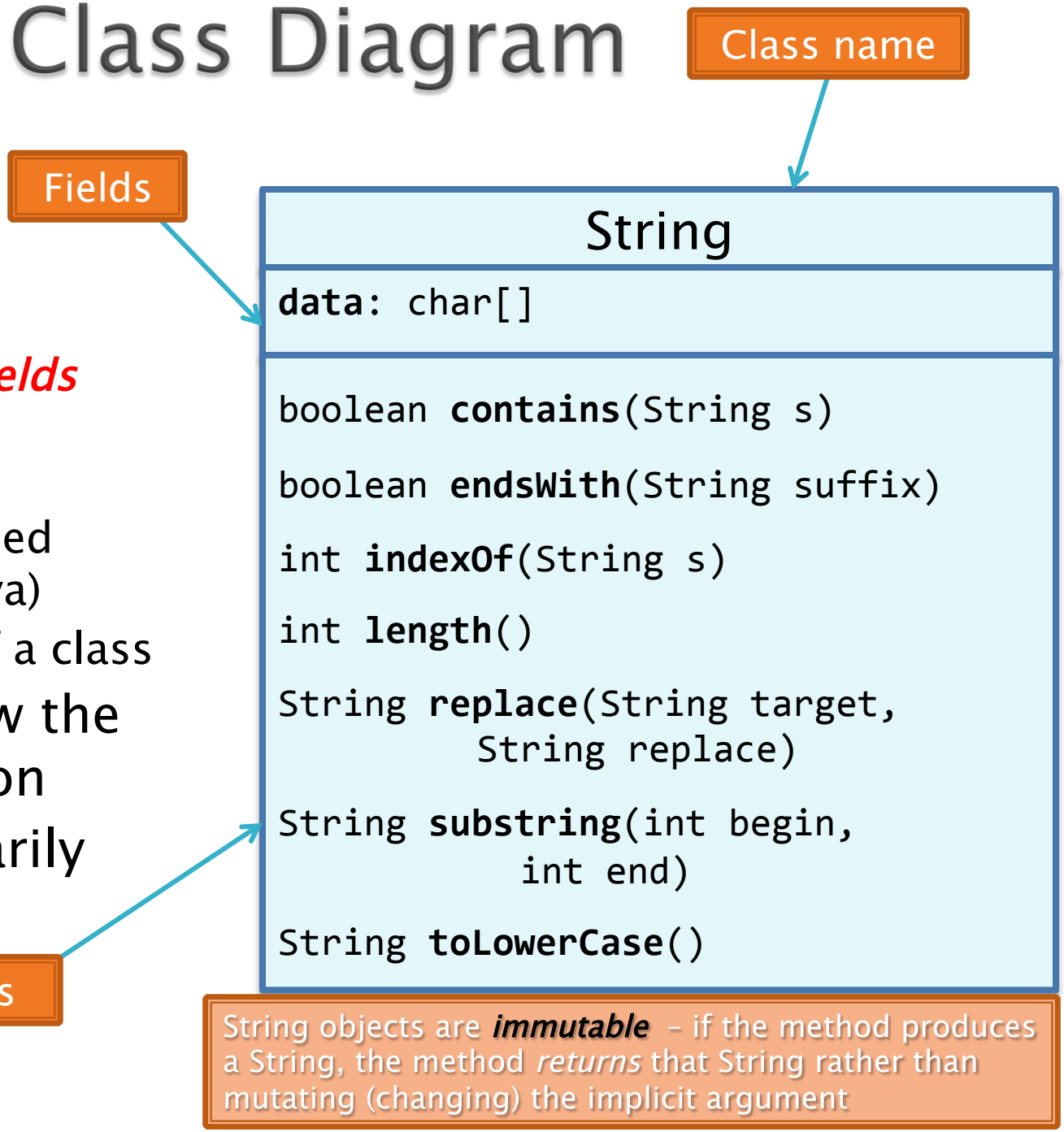
Diagramming Classes



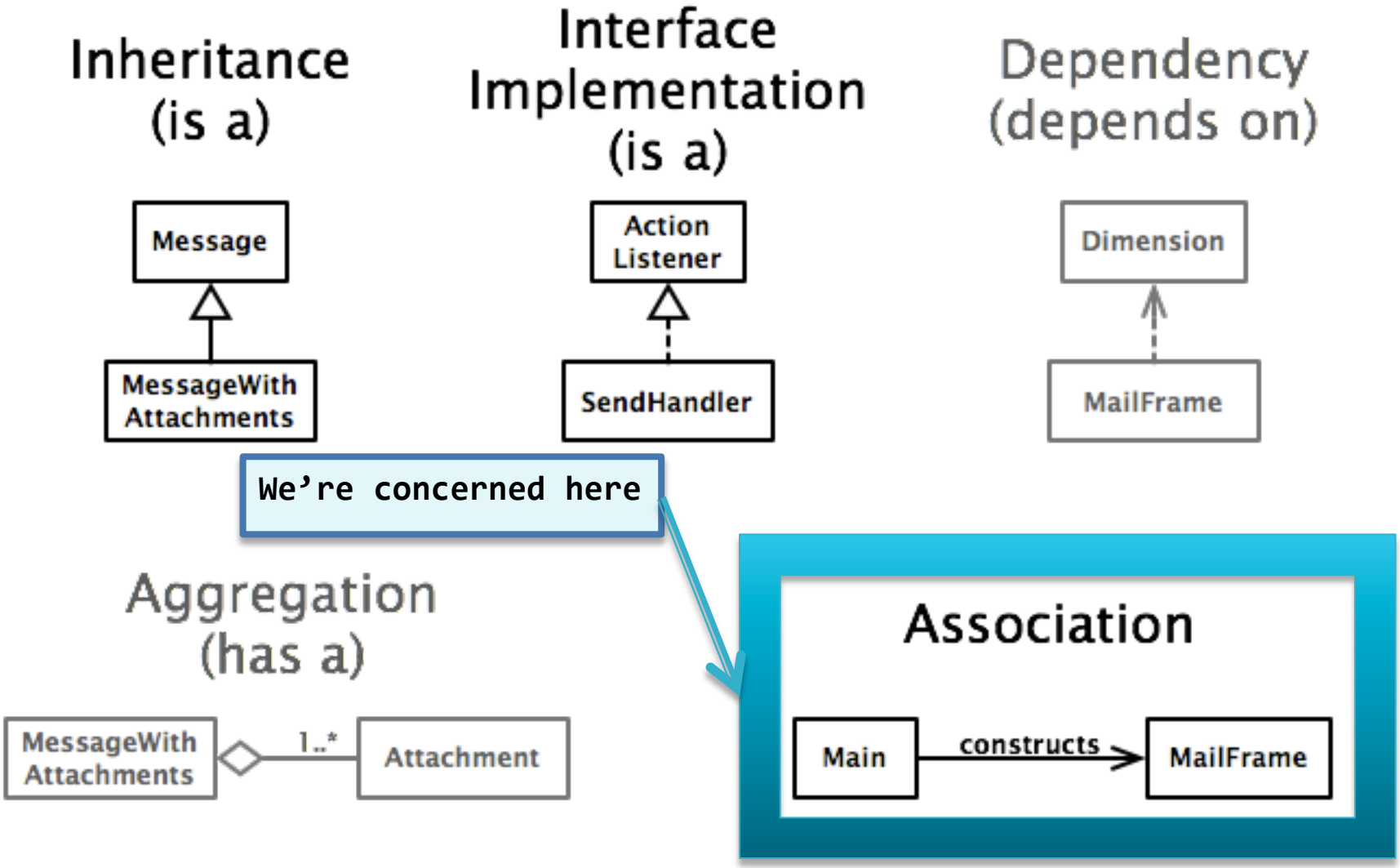
Example Class Diagram

- Shows the:
  - *Attributes* (data, called *fields* in Java) and
  - *Operations* (functions, called *methods* in Java)

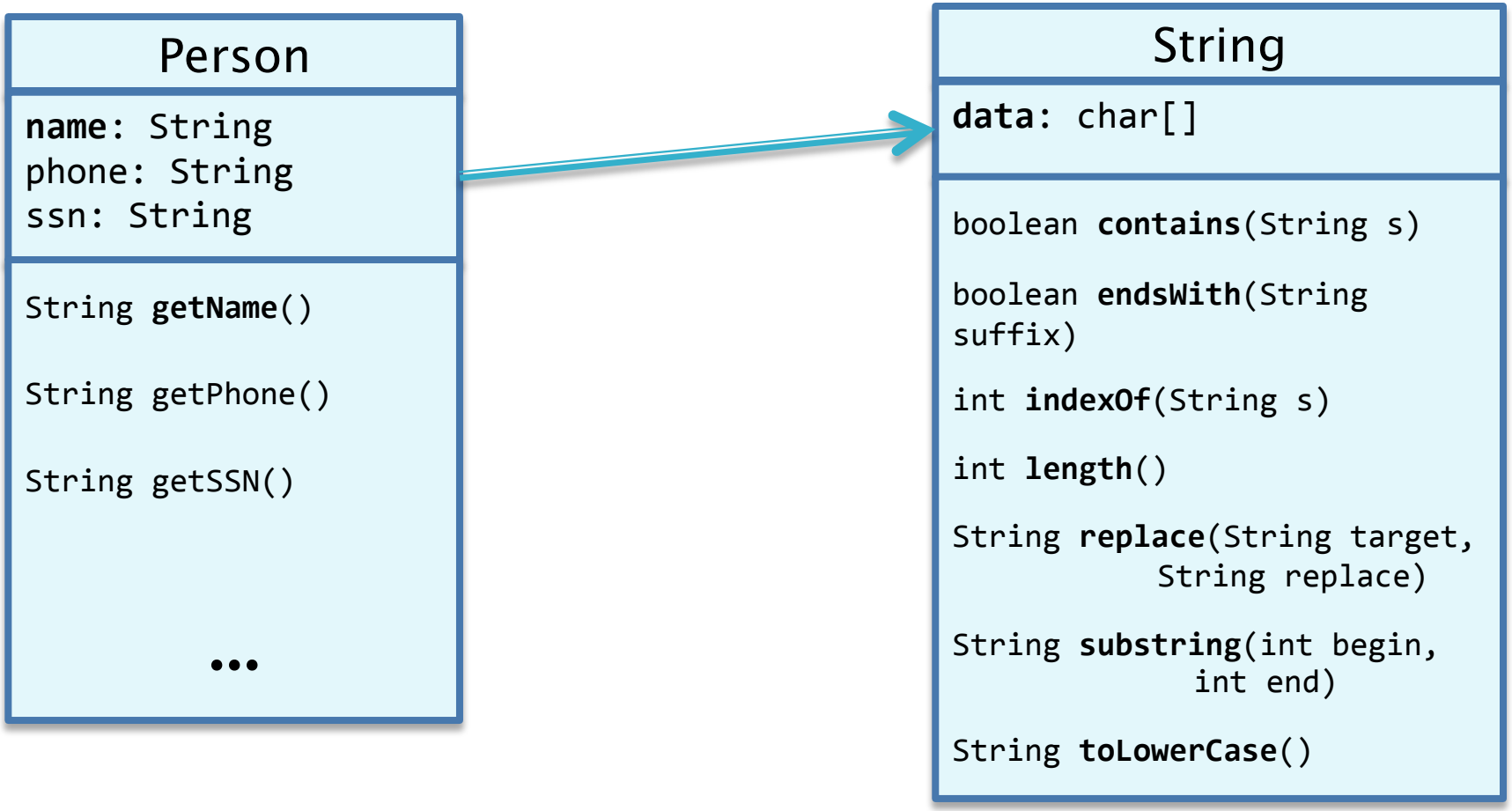
  of the objects of a class
- Does *not* show the implementation
- Is *not* necessarily complete



Summary of UML Class Diagram Arrows



Person creates String…



3 Things…

- The “things” of what you’re describing usually become the classes
  - The verbs usually become methods of the classes
- Avoid using plurals
  - We make an ArrayList of Face objects, not Faces.
- Make it work!
  - Go through it with some “use case” in mind and make sure that when this object is created, you can accomplish that case. Otherwise, redesign that design until it “works!!!”

Good Classes Typically

- Come from nouns in the problem description
- May…
  - Represent single concepts
    - `Circle`, `Investment`
  - Represent visual elements of the project
    - `FacesComponent`, `UpdateButton`
  - Be abstractions of real-life entities
    - `BankAccount`, `TicTacToeBoard`
  - Be actors
    - `Scanner`, `CircleViewer`
  - Be utility classes that mainly contain static methods
    - `Math, Arrays, Collections`

What Stinks? Bad Class Smells*

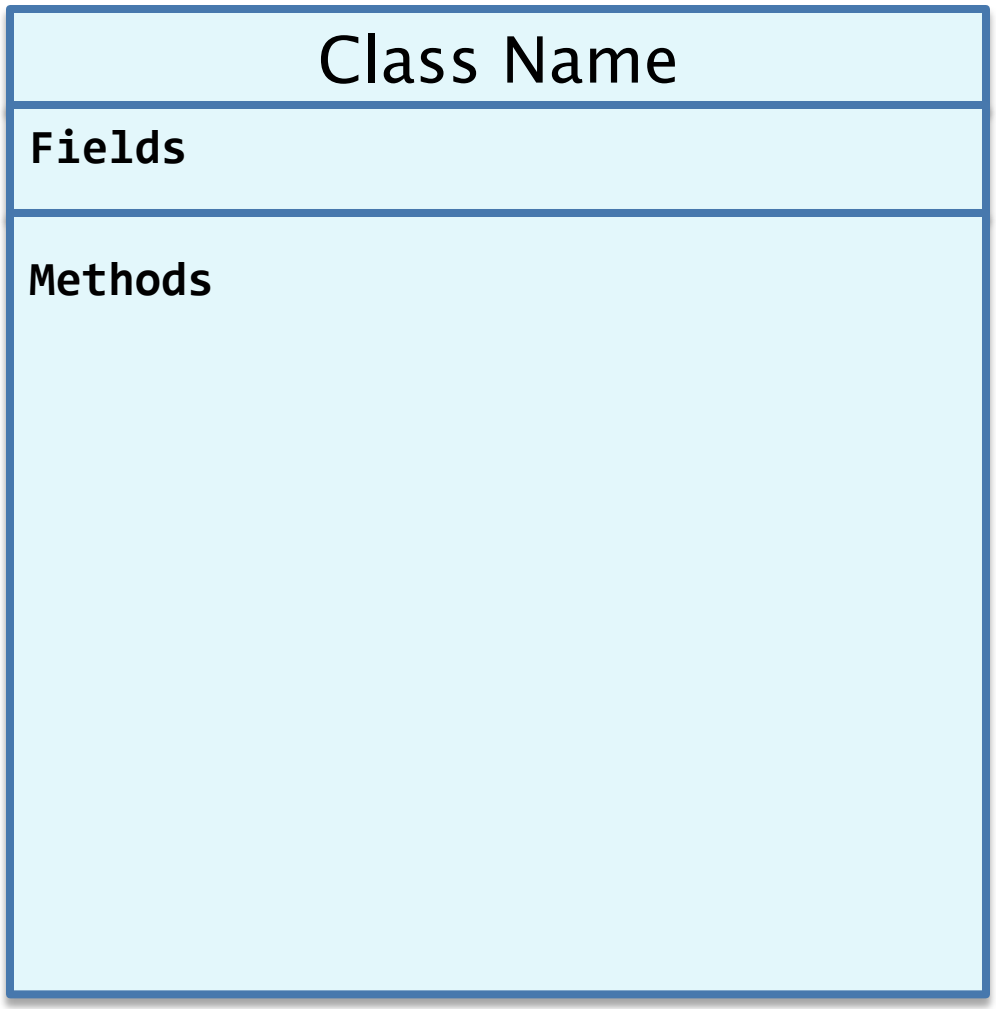
- Can't tell what it does from its name
  - `PayCheckProgram`
- Turning a single action into a class
  - `ComputePaycheck`
- Name isn't a noun
  - `Interpolate`, `Spend`

Function objects are an exception. Their whole purpose is to contain a single computation

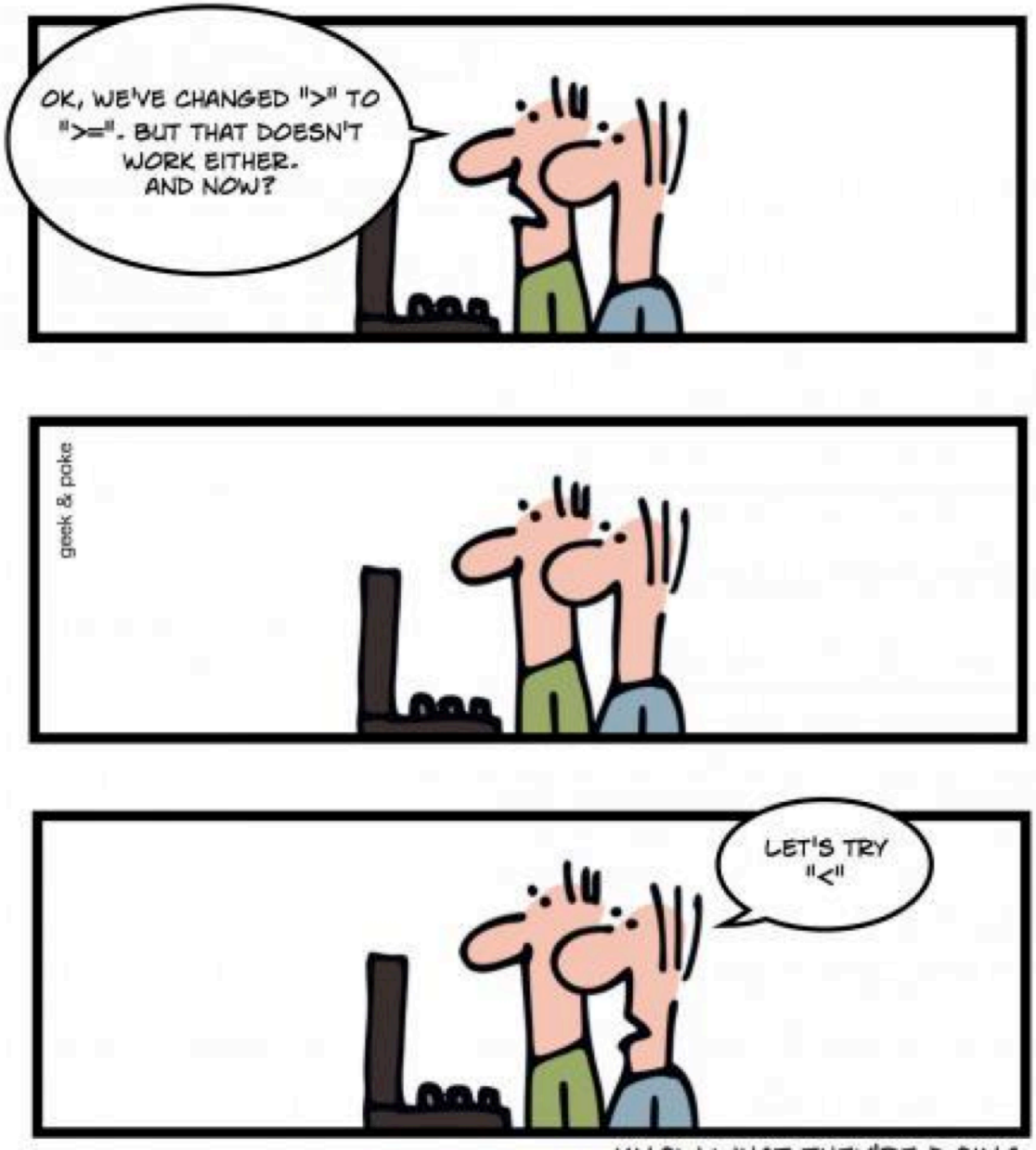
*See http://en.wikipedia.org/wiki/Code_smell
http://c2.com/xp/CodeSmell.html

Exercise: Class Diagrams

- Task: Make Class diagrams for the Invoice example

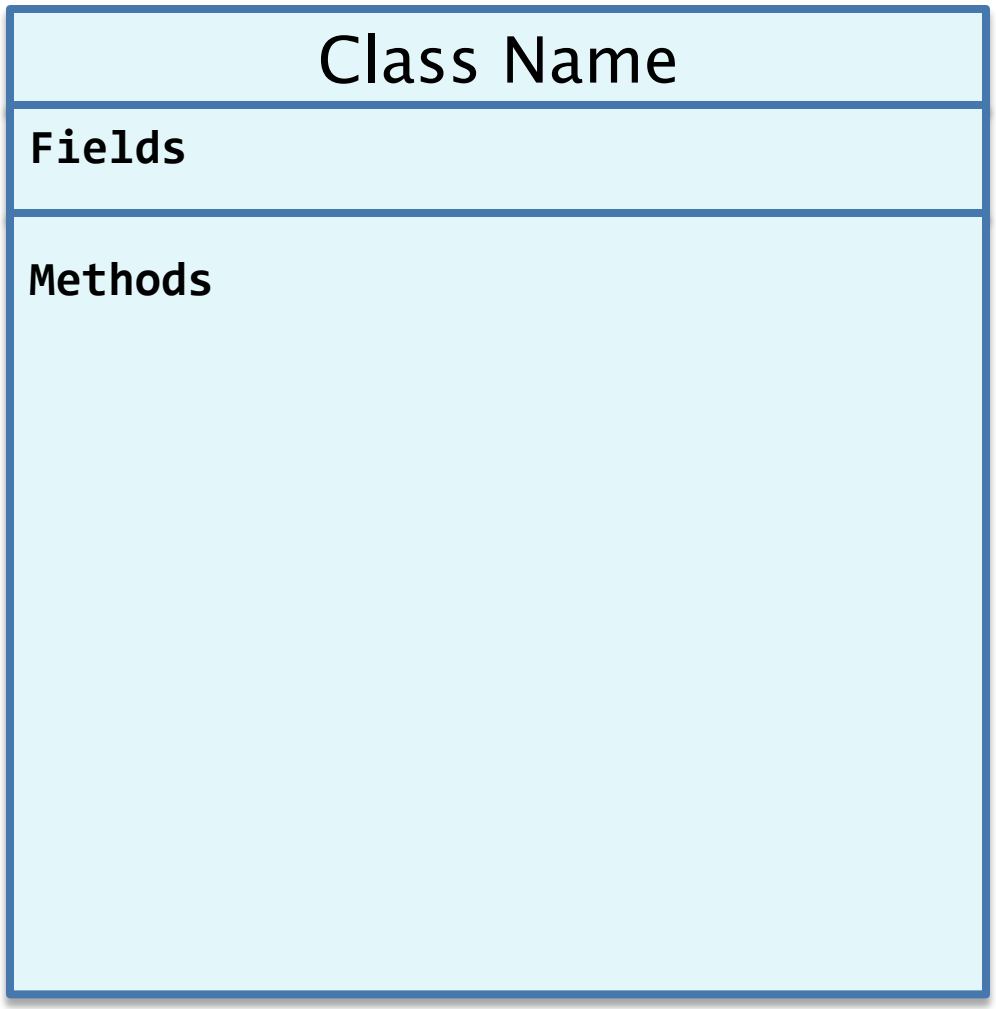


GOOD CODERS...


OK, WE'VE CHANGED ">" TO ">=". BUT THAT DOESN'T WORK EITHER. AND NOW?
geek & poke
LET'S TRY "<"
... KNOW WHAT THEY'RE DOING

Exercise: Class Diagrams

- Task: Make Class diagrams for the Simplified Blackjack example



What Is Pair Programming?

- Two programmers work side-by-side at a computer, continuously collaborating on the same design, algorithm, code, and/or test
- Enable the pair to produce higher quality code than that produced by the sum of their individual efforts
- Let's watch a video…

Pair Programming

- Working in pairs on a single computer
  - The *driver*, uses the keyboard, talks/thinks out-loud
  - The *navigator*, watches, thinks, comments, and takes notes
  - Person who really understands should start by navigating ☺
- For hard (or new) problems, this technique
  - Reduces number of errors
  - Saves time in the long run

How Does This Work? (1 of 2)

- Pair-Pressure
  - Keep each other on task and focused
  - Don't want to let partner down
- Pair-Think
  - Distributed cognition:
    - Shared goals and plans
    - Bring different prior experiences to the task
    - Must negotiate a common shared of action
- Pair-Relaying
  - Each, in turn, contributes to the best of their knowledge and ability
  - Then, sit back and think while their partner fights on

Abstracted from: Robert Kessler and Laurie Willia

How Does This Work? (2 of 2)

- Pair-Reviews
  - Continuous design and code reviews
  - Improved defect removal efficiency (more eyes to identify errors)
  - Removes programmers distaste for reviews (more fun)
- Debug by describing
  - Tell it to the "Rosie in the Room"
- Pair-Learning
  - Continuous reviews → learn from partners
  - Apprenticeship
  - Defect prevention always more efficient than defect removal

Abstracted from: Robert Kessler and Laurie Willia

Partnering the Pair

Expert paired with an Expert

Expert paired with a Novice

Novices paired together

Professional Driver Problem

Culture

Source: Robert Kessler and Laurie Williams