CSSE 220 Day 10

A Software Engineering Technique:
(Class Diagrams)

Download FirstOODesignPractice from SVN

Two-dimensional arrays

```
public class TicTacToe {
    private final int rows;
    private final int columns;
    private String[][] board;

    /**
     * Constructs a 3x3 TicTacToe board with all squares blank.
     */
    public TicTacToe() {
        this.rows = 3;
        this.columns = 3;

        this.board = new String[this.rows][this.columns];

        for (int r = 0; r < this.rows; r++) {
            for (int c = 0; c < this.columns; c++) {
                this.board[r][c] = " ";
            }
        }
    }
```

What is the value of `this.board[1][2]` immediately after

Could have used: `this.board.length`

Could have used: `this.board[r].length`

Note the (very common) pattern: loop-through-rows, for each row loop-through columns

Maps

- Associate keys with values
- Real-world "maps"
  - Dictionary
  - Phone book
- Some uses:
  - Associating student ID with transcript
  - Associating name with high scores

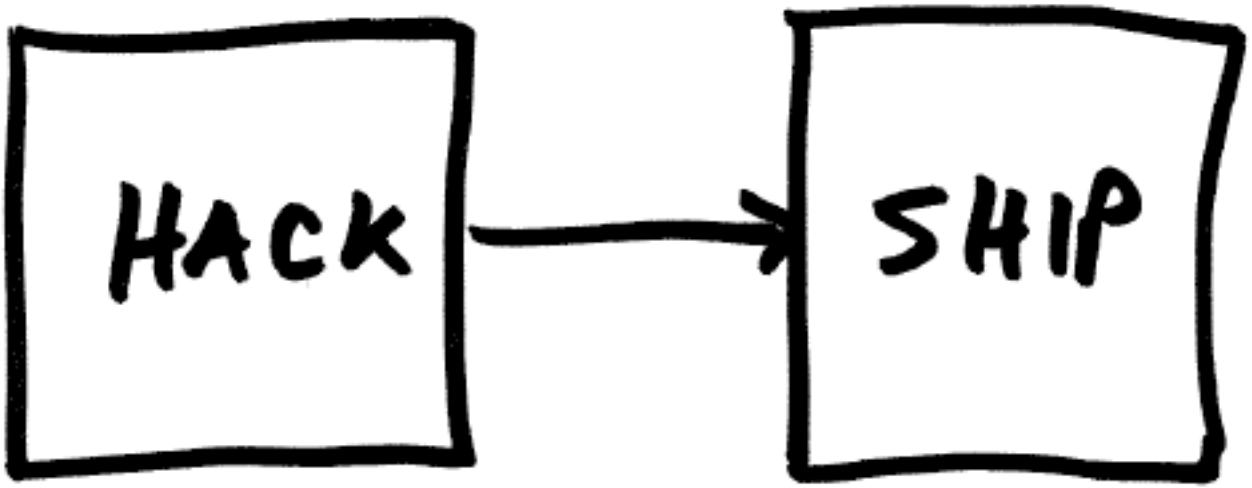
Work with your groups of 3/4

- Decide what classes ought to be in the system and what methods/fields those classes should have (your design should have at least 2 classes)
- Don't forget one class needs to have a main method
- Make sure your design works!
- Write down your answers on a piece of paper with all of your team's names on it
- Call me over when you think you're done – then you'll implement it

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

Software Process: The Early Days



So, what is Software Process?

Hint: software is the part of a computer system that is suppose to change!

- Take 15 seconds and think about it
- Turn to neighbor and discuss what you think for a minute
- Let's talk?

Iterative

Waterfall

Incremental

Spiral

Extreme Programming

Producing Software is an Elaboration and Refinement Process

- Starting with Abstract Requirements, successively *Elaborate* and *Refine* them into specifications, models, and more concrete implementation

- A Software Process organizes the life cycle activities related to the creation, delivery, and maintenance/evolution of software systems

Q1, 2

Software Engineering Techniques

- Class Diagramming (today)
- Pair programming
- Team version control
- Regression Testing

Tools of the Trade

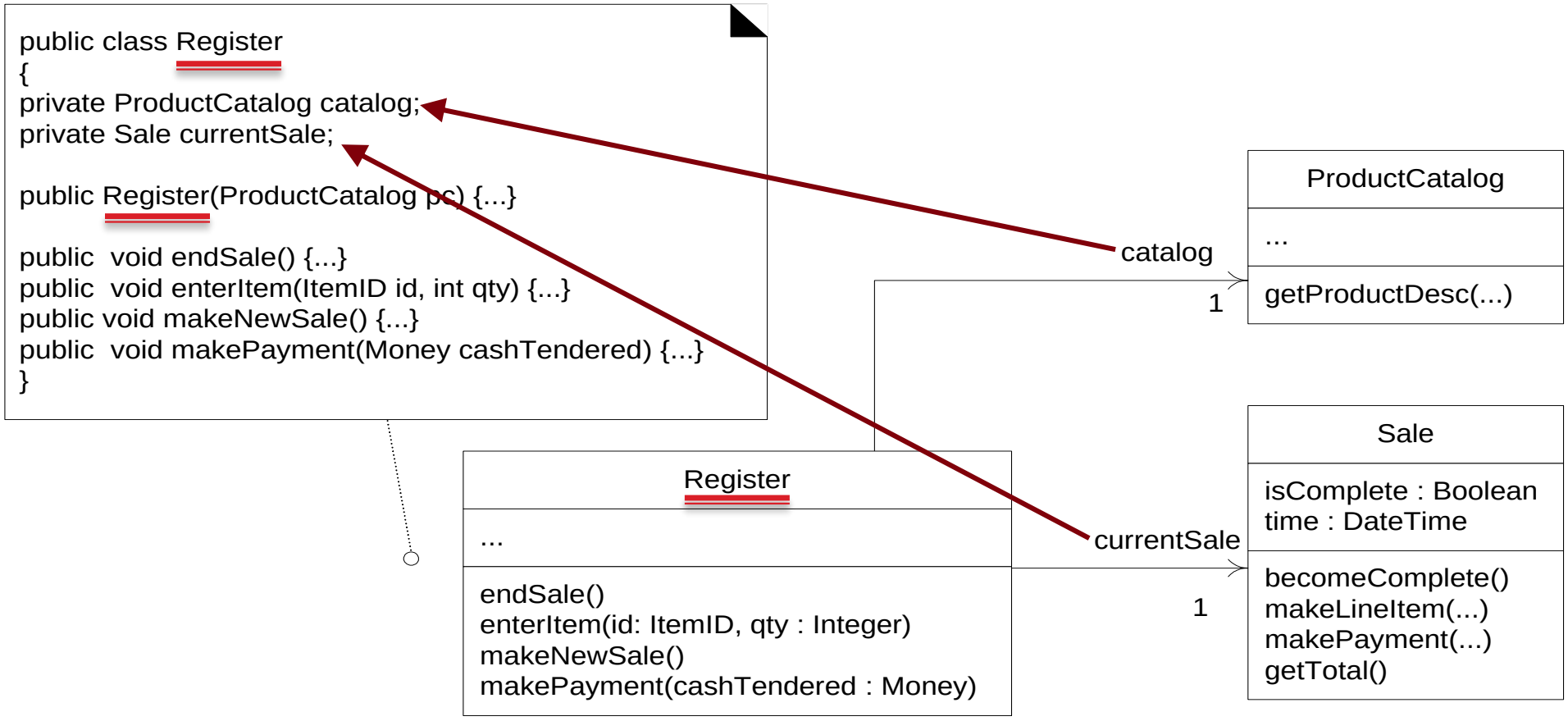
- Class Diagrams (UML)
- UML – Unified Modeling Language
  - Language unspecific
  - provides guidance as to the order of a team's activities
  - specifies what artifacts should be developed
  - directs the tasks of individual developers and the team as a whole
  - offers criteria for monitoring and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

According to UML-Diagrams.org

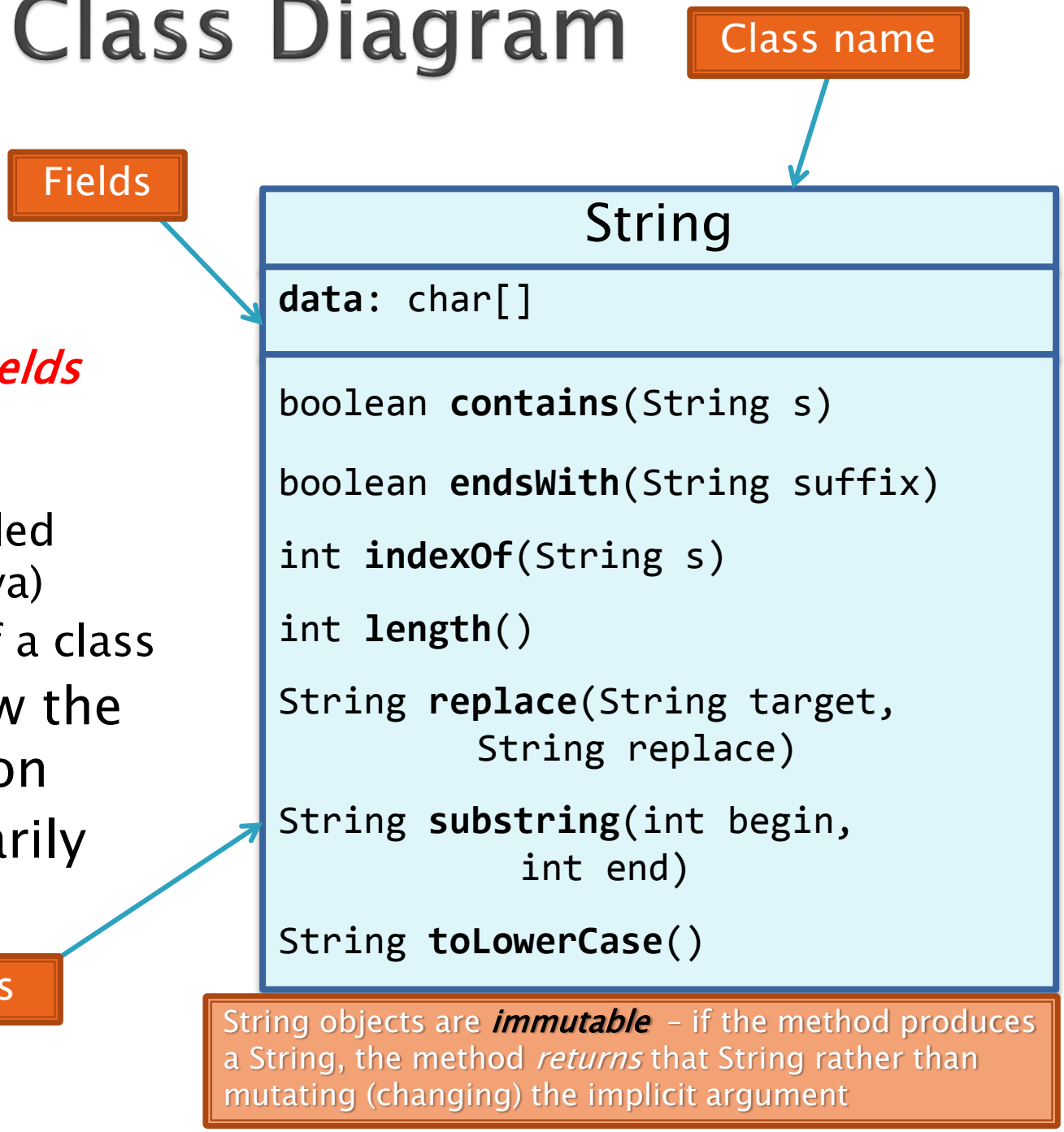
- The **Unified Modeling Language™** (**UML®**) is a standard visual modeling language intended to be used for
  - modeling business and similar processes,
  - analysis, design, and implementation of software-based systems

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



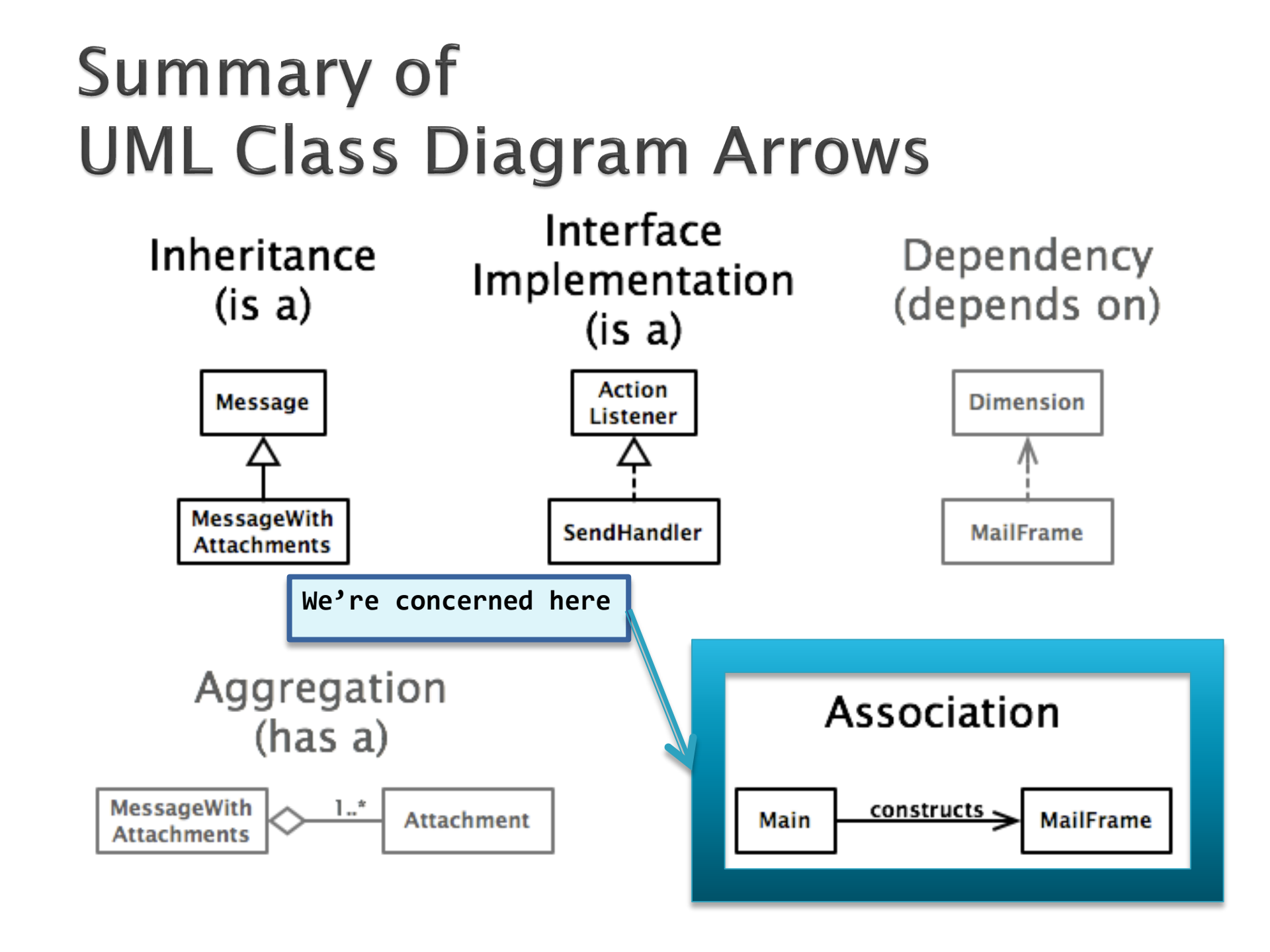
Example Class Diagram



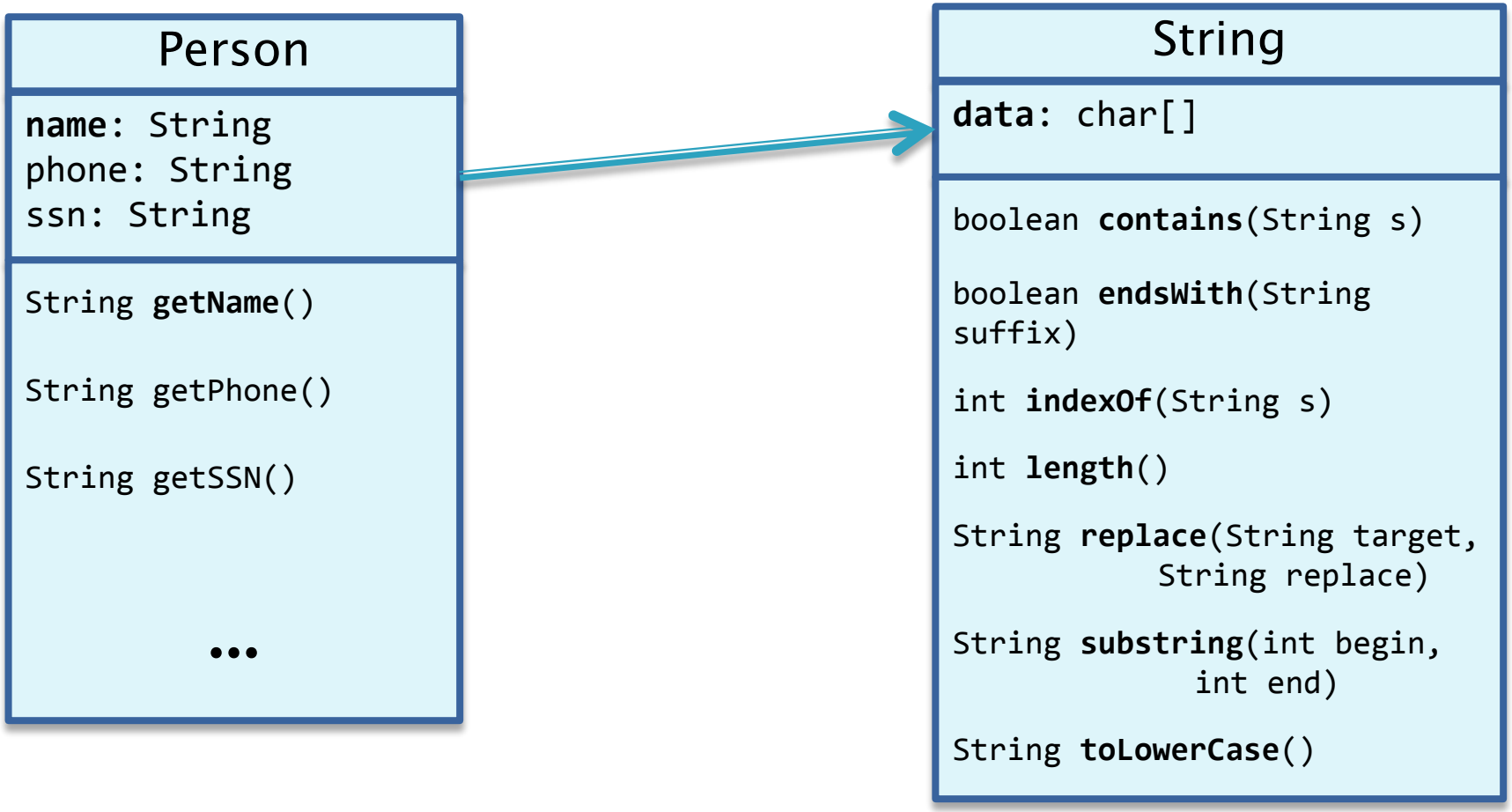
- Shows the:
  - *Attributes* (data, called *fields* in Java) and
  - *Operations* (functions, called *methods* in Java)

  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*

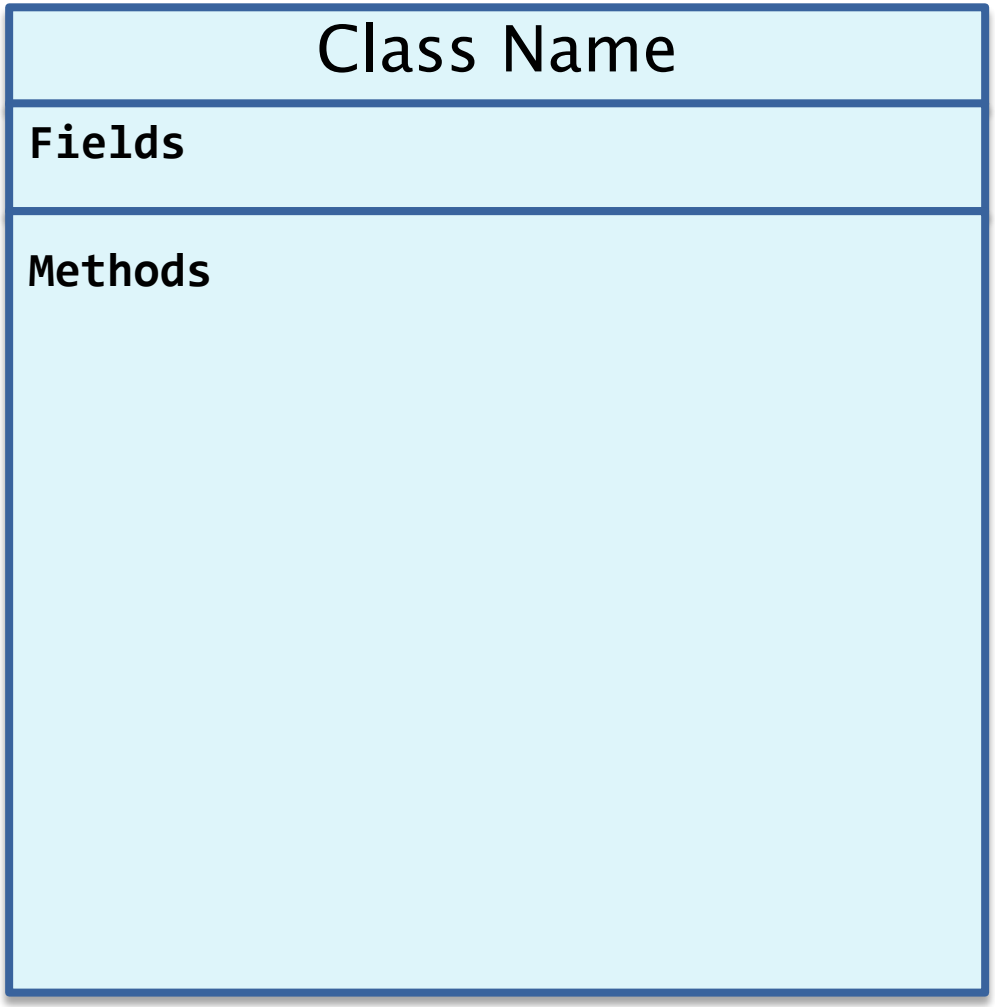
- Can't tell what it does from its name
  - `PayCheckProgram`
- Turning a single action into a class
  - `ComputePaycheck`
- Name isn't a noun
  - `Interpolate`, `Spend`

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

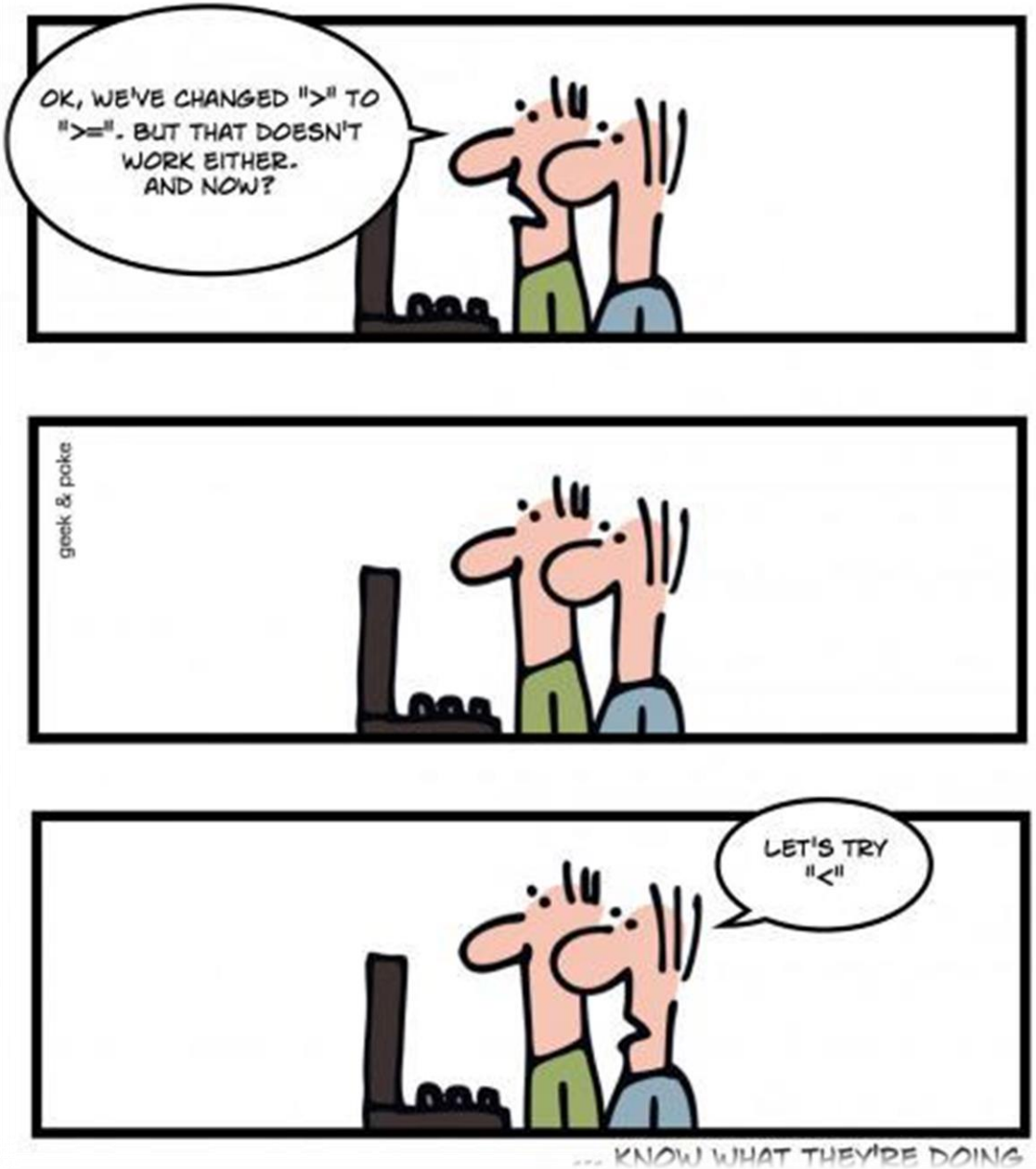
*See http://en.wikipedia.org/wiki/Code_smell
http://c2.com/xp/CodeSmell.html

Exercise: Class Diagrams

- Task: Make Class diagrams for the Invoice example from OrderTaker



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

Blackjack – Work with your groups of 4

- Decide what classes ought to be in the system and what methods/fields those classes should have (your design should have at least 3 classes)
- Don't forget one class needs to have a main method
- Make sure your design works!
- Write down your answers on a piece of paper with all of your team's names on it
- Call me over when you think you're done

Exercise: Class Diagrams

- Task: Make Class diagrams for the Simplified Blackjack example
- Make sure all names are on the page. Turn this in for your quiz grade today