CSSE 220

Software Engineering Techniques
Design Principles
Encapsulation

Today's Agenda

- Collect Quizzes and go over solutions
- Software Engineering Techniques:
  - Pair programming
  - Version Control (briefly!)
- Focus on more OO Design principles:
  - **Spread** functionality throughout the system
  - Encapsulation

Software Engineering Techniques

- Pair programming
  - Upcoming assignment *CrazyEights* requires this!
- Version Control
  - How to avoid merge conflicts in SVN

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

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

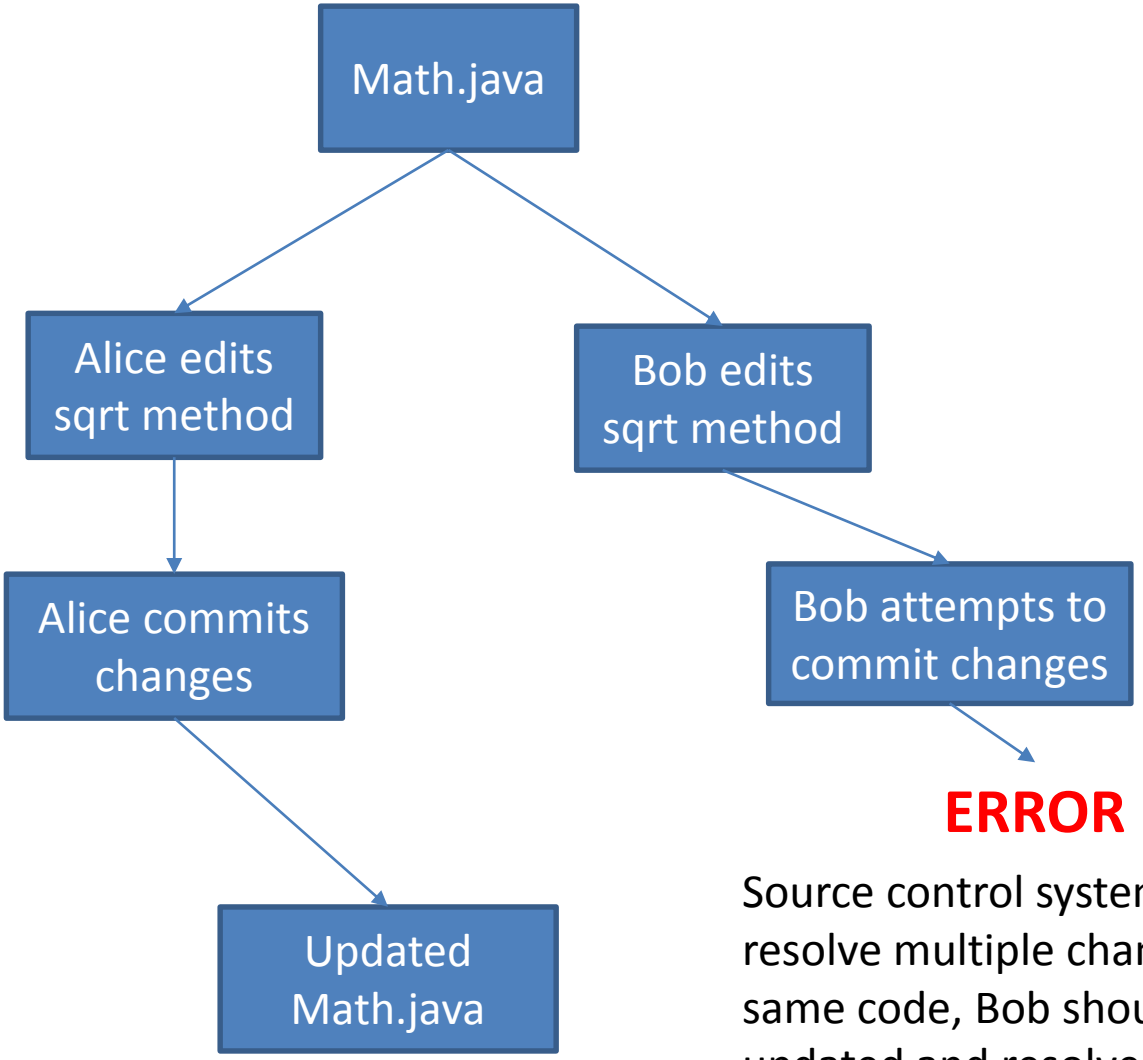
Q1

Pair programming video

- https://www.youtube.com/watch?v=rG_U12uqRhE

SOFTWARE VERSIONS

When Two+ People Edit the Same Code



ERROR

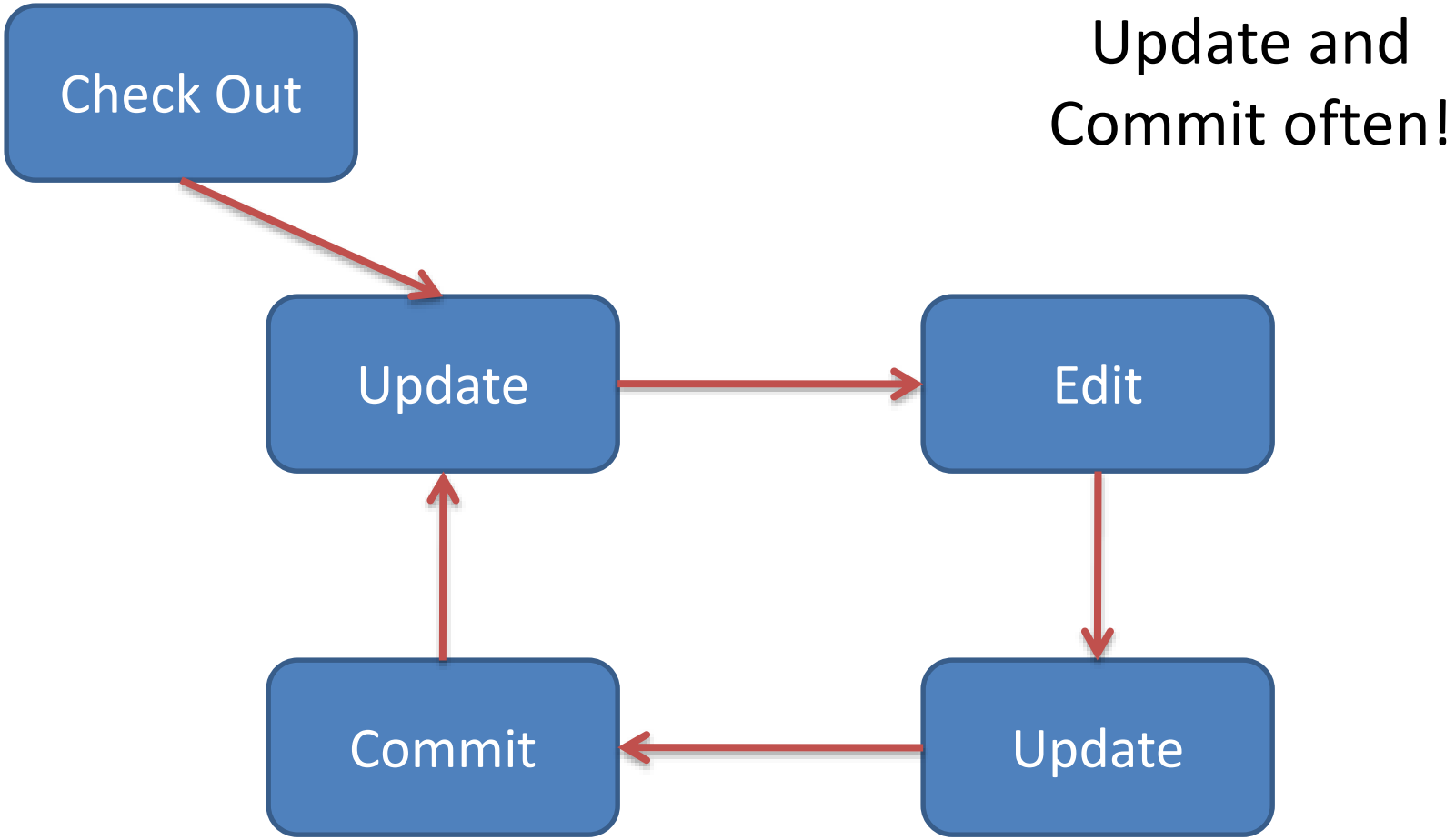
Source control system cannot resolve multiple changes on the same code, Bob should have updated and resolved conflicts before committing.

Team Version Control

- **Version control tracks multiple versions**
  - Enables old versions to be recovered
  - Allows multiple versions to exist simultaneously
- **Always**:
  - **Update before** working
  - **Update again** before committing
  - **Commit often** and with good messages
- **Communicate** with teammates so you don't edit the same code simultaneously
  - Pair programming ameliorates this issue ☺

Q2

Team Version Control



Update and
Commit often!

What if I get a conflict on update?

- If you did an update and now have File.java, File.java.mine, File.java.rN, and File.java.rM (where N and M are integers):
  - YOU HAVE A CONFLICT!
- Eclipse provides tools for resolving conflicts
- Follow the steps in this link to resolve a conflict:
  - http://www.rose-hulman.edu/class/csse/csse221/current/Resources/ResolvingSubversionConflicts.htm

Moving on….

- More Object-Oriented Principles for Design
- Learn about next set of principles
  - What are they?
  - Why are they useful?
  - When are the most important?
  - How can we apply them?

Major Goals of ALL Program Design

- Say someone has written a program that works and it has no bugs, but it is *poorly designed*.
  - What does that mean?
  - Why do we care?
- I think there are two things that would be nice

Principles of Design (for CSSE220)

1. Structure your program **around the data** that needs storing
   a) Nouns become your classes, operations become their methods
2. Your structure needs to **function correctly**
   a) Every class must have access (directly or indirectly) to the data it needs to complete its operations
   b) Usually this means the problem must be modeled correctly
   c) Data should also not be duplicated
3. Functionality should be **spread** throughout the system
   a) No single part of the system should get too large
   b) Each class should have a single responsibility it accomplishes
4. **Minimize dependencies** between objects when you can
   a) Ask don't tell
   b) Don't have message chains
5. **Don't duplicate** code
   a) Similar "chunks" of code should be unified into functions
   b) Classes with similar features should be given common interfaces
   c) Classes with similar internals should be simplified using inheritance

What are the principles?

3. Functionality should be **spread** throughout the system
   a) No single part of the system should get too large
   b) Each class should have a single responsibility it accomplishes

Why do we want to spread things out?

Why is it good to have a single responsibility?

Why do we even have classes?

What if there were no String class?

- Instead, what if we just passed around arrays of characters - char[]
- And every String function that exists now, would instead be a function that operated on arrays of characters
- E.g. char[] stringSubstring(char[] input, int start, int end)
- Would things be any different? Discuss this with the person next to you.

Concatenate…

```
String stringName1 = "jason";
String stringName2 = "yoder";
String stringConcat = stringName1.concat( stringName2 );
System.out.println(  stringConcat );
-------------------------------------------------------
char[] charName1 = {'j','a','s','o','n'};
char[] charName2 = {'y','o','d','e','r'};
char[] charConcat =
       new char[charName1.length + charName2.length];

for (int i=0; i< charName1.length; i++) {
       charConcat[i] = charName1[i];
}
for (int i=0; i< charName2.length; i++) {
       charConcat[charName1.length + i] = charName2[i];
}
System.out.println( Arrays.toString(charConcat)  );
```

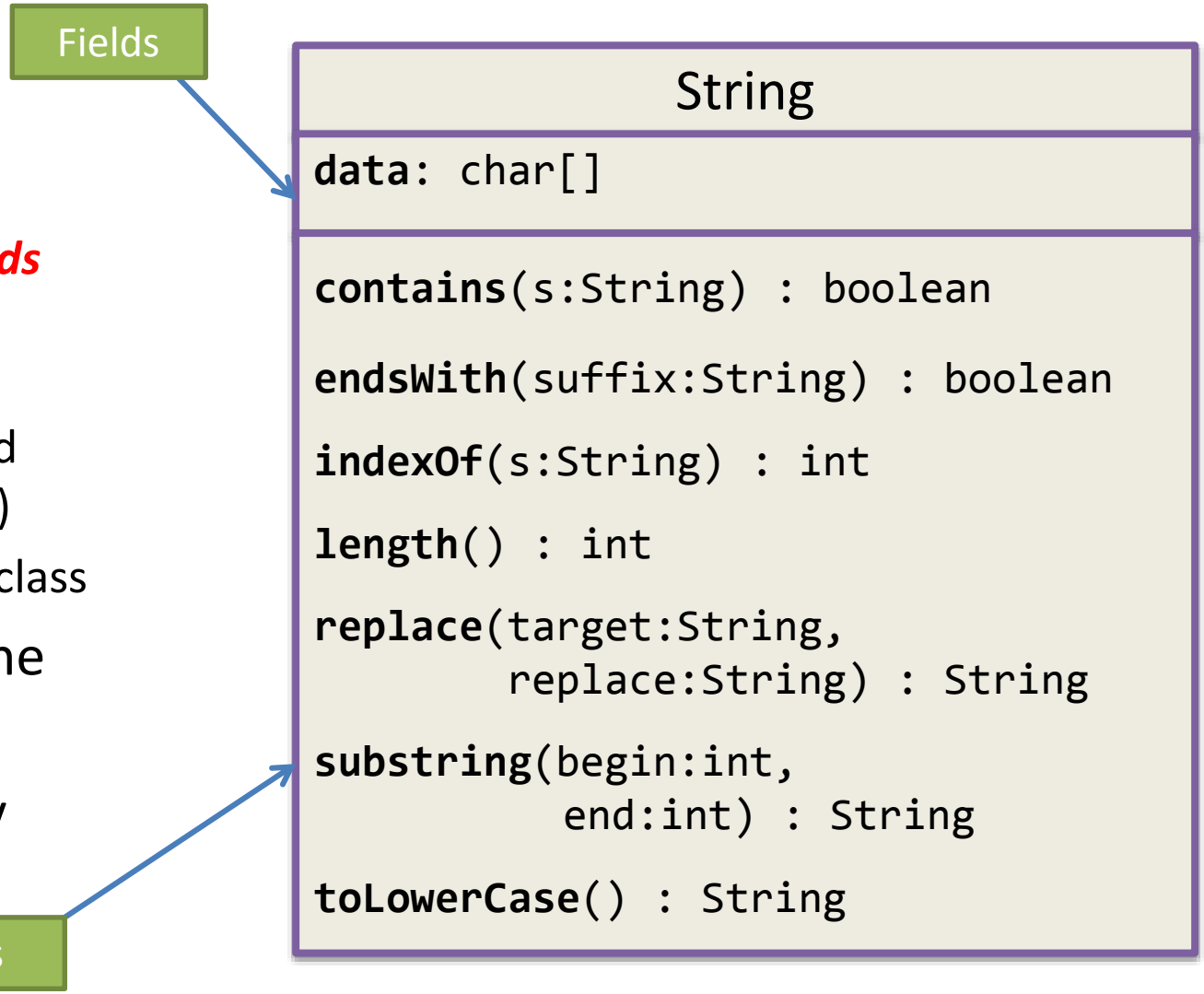
Class sizes

- Why not put all the Math utilities in the String class?
  - We could just get anything we need done with one library!
- Let's look at a slightly expanded UML of a portion of the String class for further consideration

Adding Types to The 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



Pizza Restaurant Scenario

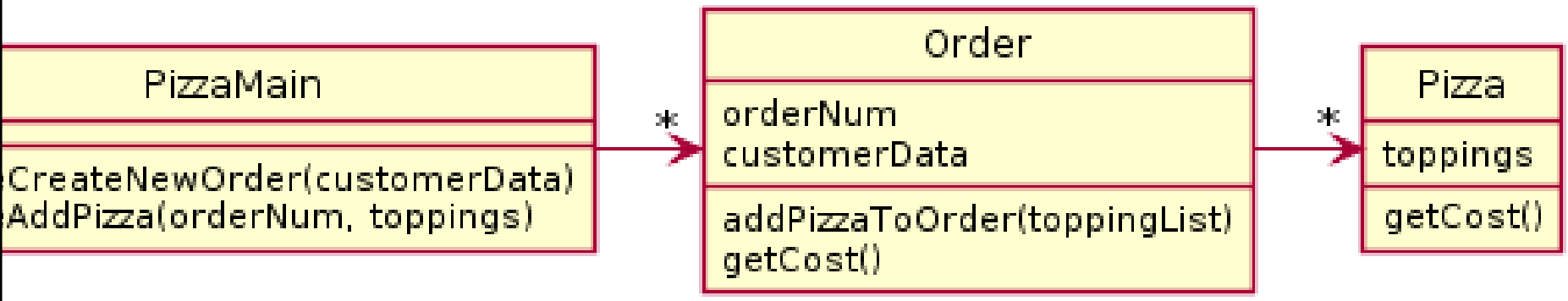
A pizza restaurant needs to calculate the costs of orders and record what pizzas need to be made. An order consists of a number of pizzas which might have toppings as well as a customer's name and an order date. Each pizza costs $8 with no toppings. The first 2 toppings cost $2 apiece. Additional toppings beyond that cost $1. If a pizza has just peppers, onions, and sausage - that's "The special" and it costs $13.

Design a UML diagram ***with types*** to model this.

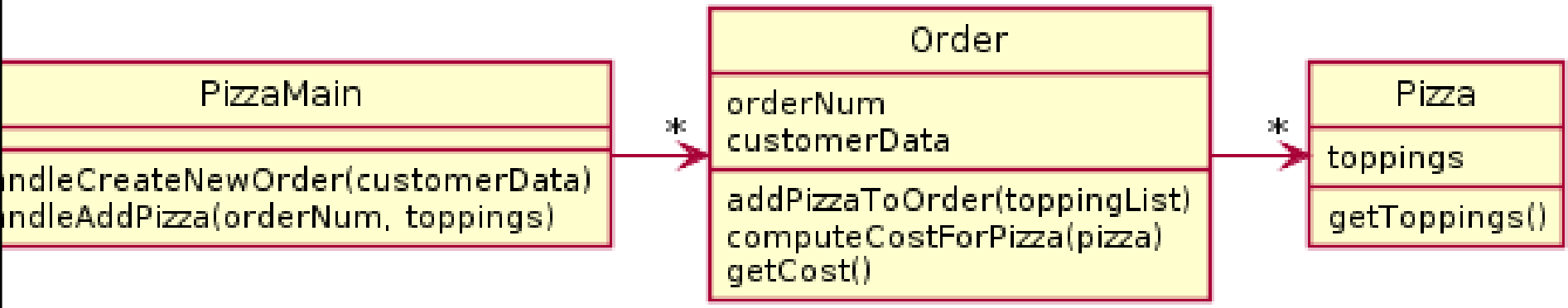
UML

1. What classes did you have?
2. Where did you put “costOfPizza()”?

Solution A

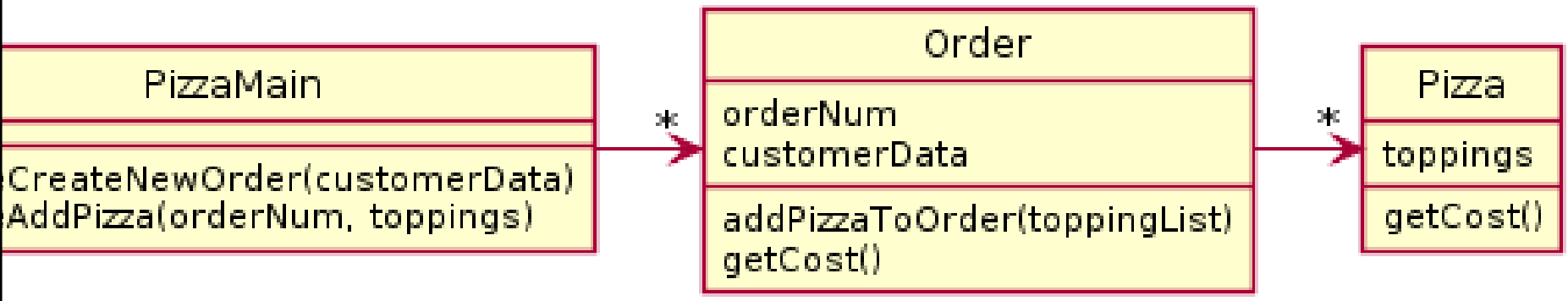


Solution B

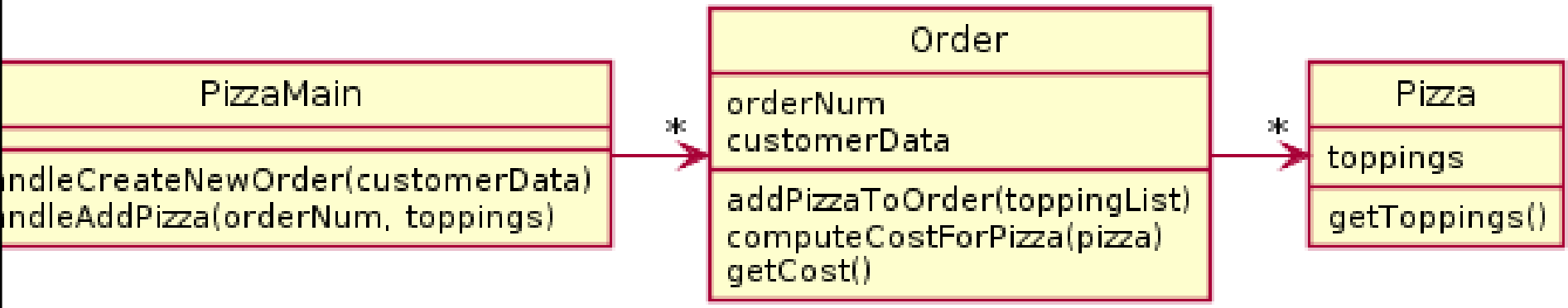


Which is better?

Solution A



Solution B



Conceptually, calculating costs could belong in either order or pizza. But order is doing a lot of stuff – Pizza is just a dumb data holder. So by spreading the functionality into the pizza, we improve the design.

Alternate Pizza Restaurant

Consider now the ability to add a discount to an order, such that a coupon can be added to an order and then it changes how the cost is calculated. A coupon may offer a discount percentage for toppings (50% off all toppings) and/or percentage off of entire orders. In addition, there should be a way to calculate how long it takes to create a pizza based on its size and toppings.

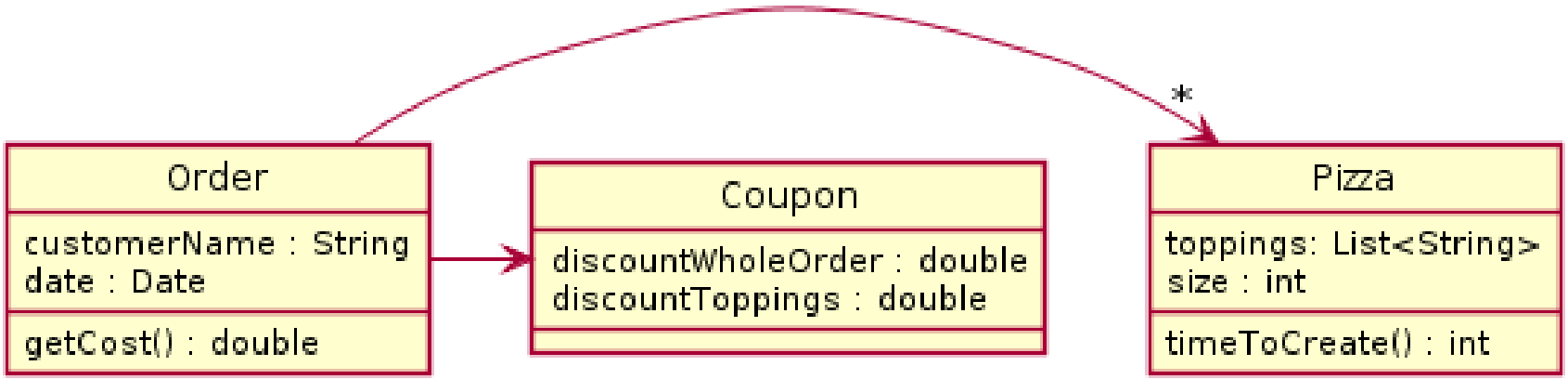
Design a UML diagram ***with types*** to model this.

UML

1. What classes did you have?
2. Where did you put “getCost()”?

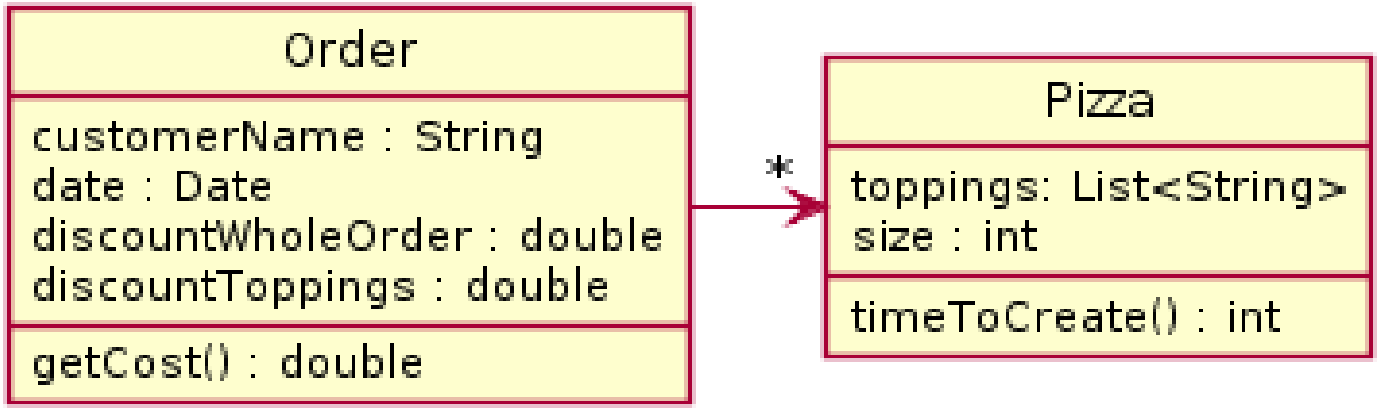
One Solution

3. Functionality should be **spread** throughout the system
   a) No single part of the system should get too large
   b) Each class should have a single responsibility it accomplishes



Do we need Coupon or Topping?

- It depends, do the classes do anything ***with*** their data, or are the just ***data classes*** that simply all you to get and set values?

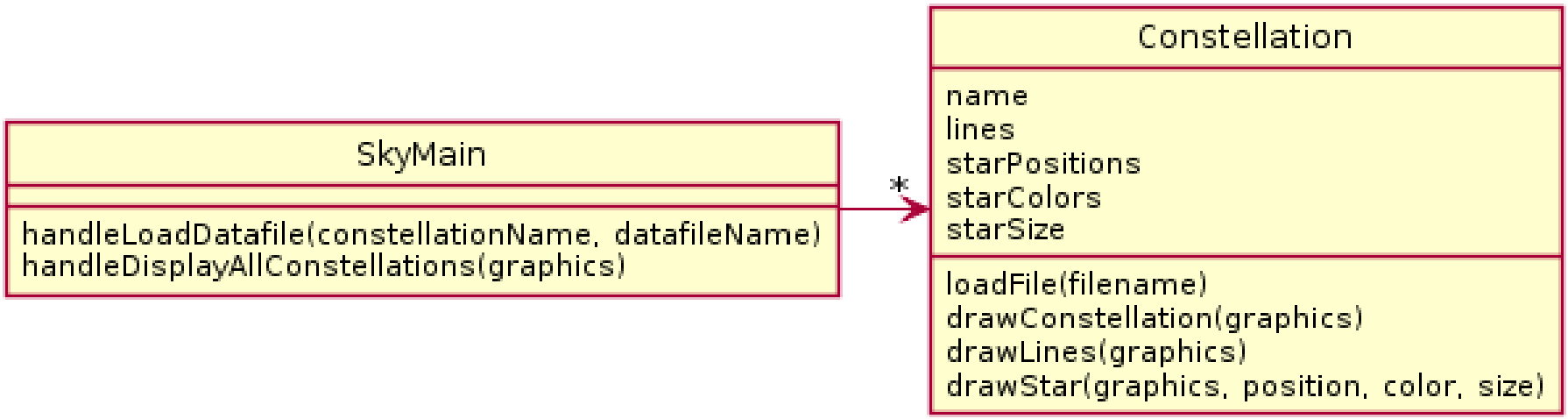


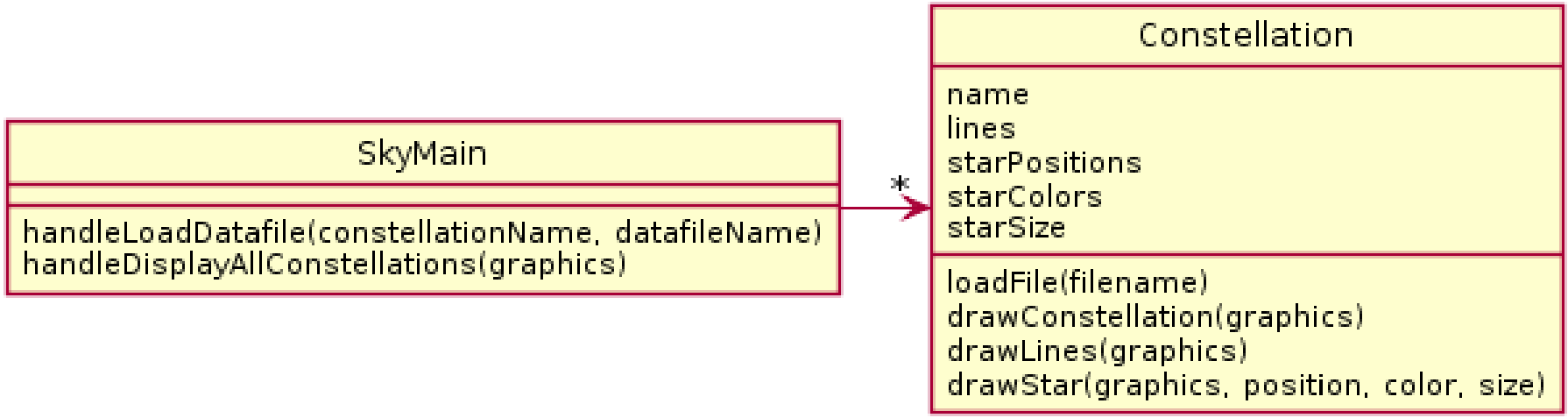
Rule of Thumb - Avoid Data Classes!

- A data class is a class that just contains getters and setters
- Often, we think of Data Classes as violating a principle of OOD called ***encapsulation*** because they aren't in control of their own data – they are just dumb repositories for other classes to use
- Usually you can improve a data class by finding functionality to add to them

A particular program is designed to load constellations from datafiles and draw them on the screen. The datafiles includes include details about star location size and color as well as which stars ought to be connected to draw the constellation. Depending on the star data, each star should be drawn differently (e.g. right size, right color).

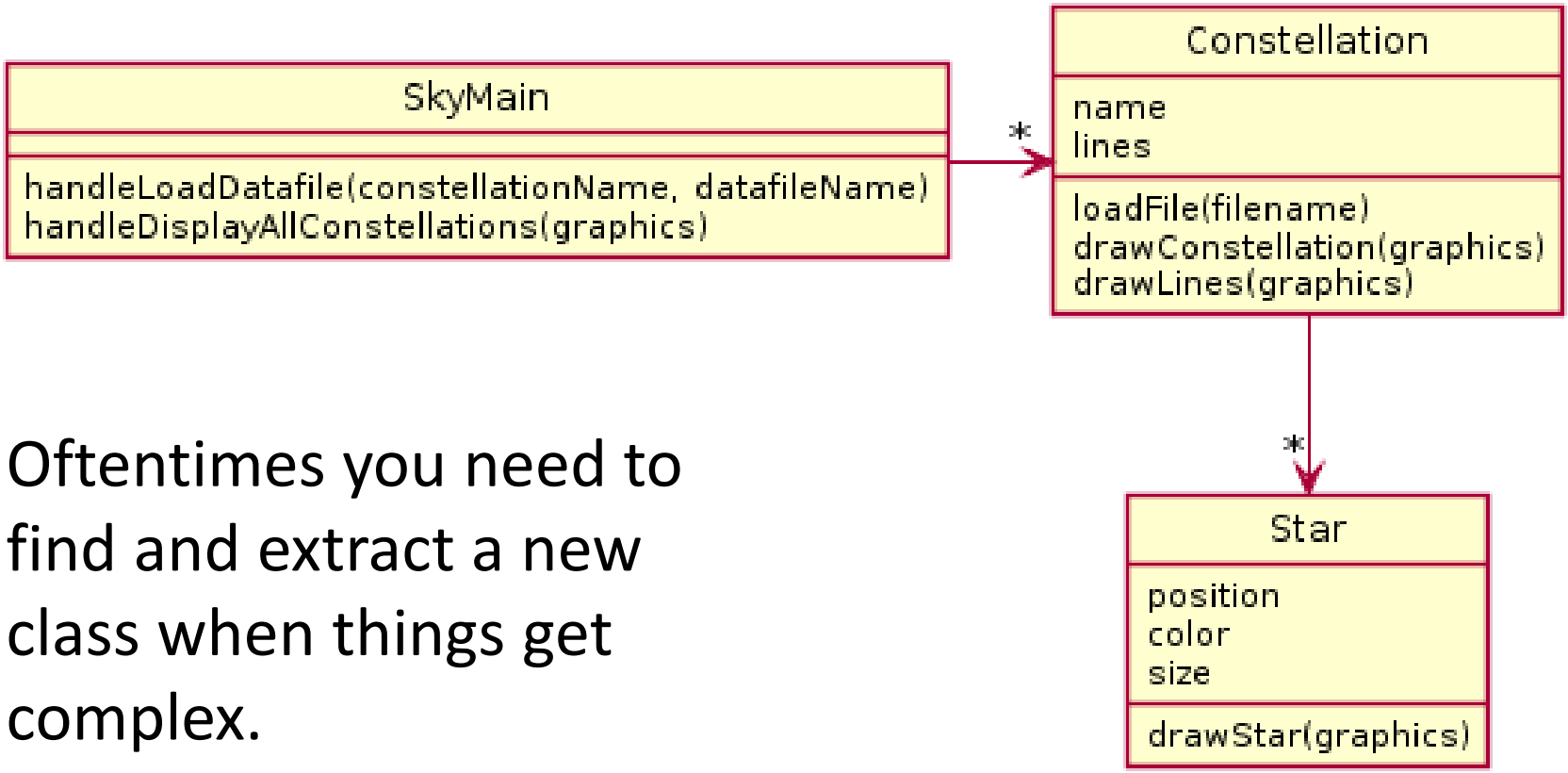
Explain the problem with the given solution and then propose a UML solution of your own.





3a. Constellation does everything (except maybe the parsing done by main).

My solution



Oftentimes you need to find and extract a new class when things get complex.

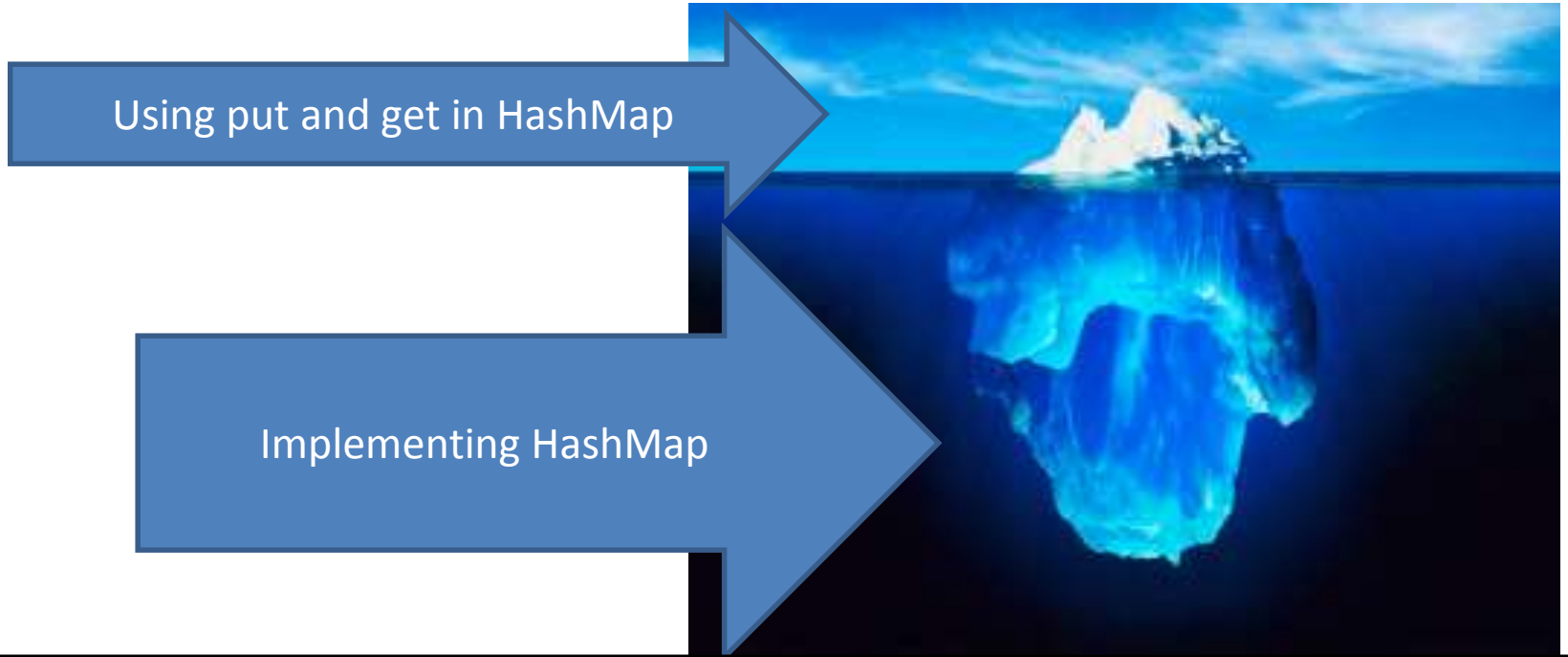
Encapsulation

- Makes your program easier to understand by
  - Grouping related stuff together
- Rather than passing around data, pass around objects that:
  - Provide a powerful set of operations on the data
  - Protect the data from being used incorrectly

Q3

Encapsulation

- Makes your program easier to understand by…
  - Saving you from having to think about how complicated things might be



Encapsulation

Makes your program easier to change by…

- Allowing you to change how your data is represented

Q4

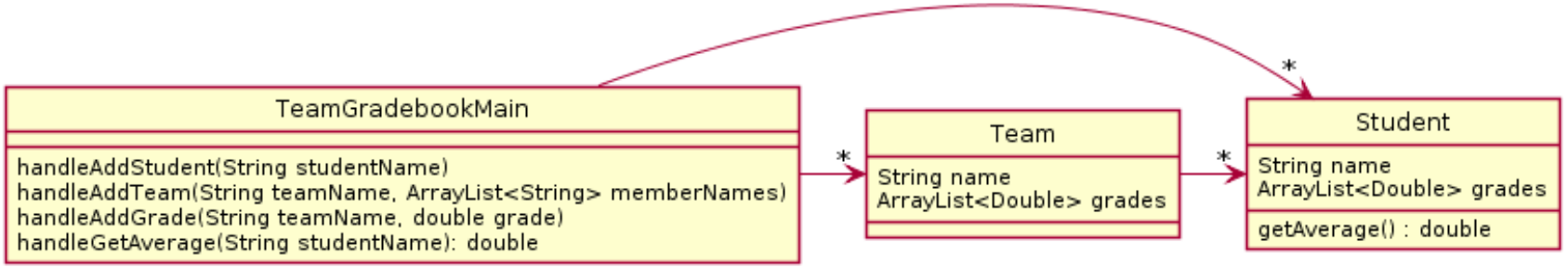
A simple example of encapsulation

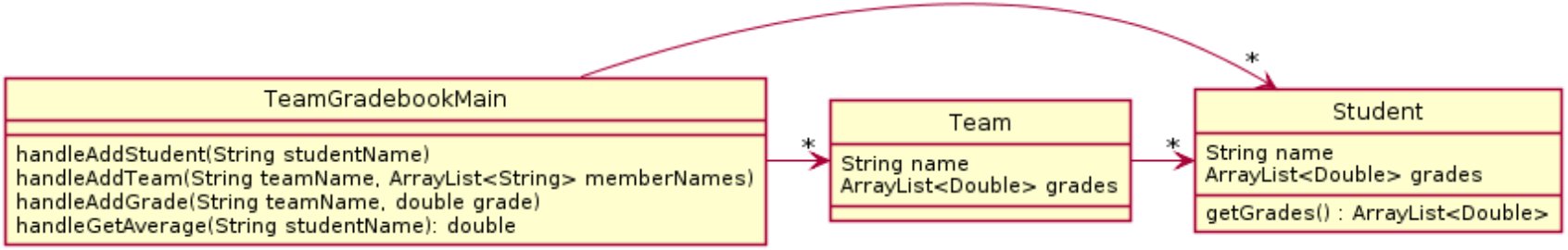
In your TeamGradebook classes, you need to calculate a student's average grade. This could be accomplished by:

1) Adding a getAverage() method to the Student class which calculates the average
2) Adding a getGrades() method to the student class, which the TeamGradebook class could call, and then use to compute the average

Which of these is most encapsulated?

Diagrams look similar!





Diagrams look similar!

How would the actual code compare when performing the stated task “calculate a student’s average grade”?

getGrades()

```
public class TeamGradebook {
…
private String handleGetAverage(String studentName) {
        Student student = getStudentByName(studentName);
        if (student == null) {
                return "student " + student + " not found";
        }
        double average = 0;
        for (double d: student.getGrades() ) {
                average += d;
        }
        return Long.toString(Math.round(average));
}
…
}
```

Calculation happening in TeamGradebook!

getAverage()

```
public class TeamGradebook {
…
private String handleGetAverage(String studentName) {
        Student student = getStudentByName(studentName);
        if (student == null) {
                return "student " + student + " not found";
        }
        return Long.toString(Math.round(student.getAverage()));
}
…
}
```

Calculation happening in Student!

Why does this improve the design?

- It makes the Student object more featureful, and puts the code in an expected place
- Reduces the code in TeamGradebook which is already quite long
- Allows you to change how the grades are represented in TeamGradebook, should you wish to (i.e. drop lowest score)

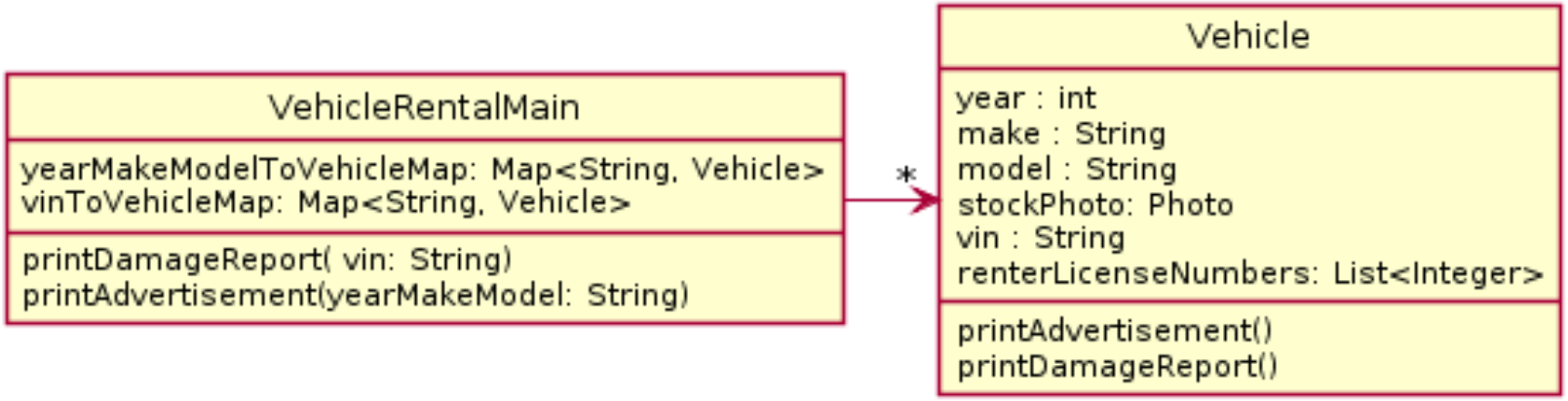
Your turn!

- Try to design UML for the following scenario

Rental Company

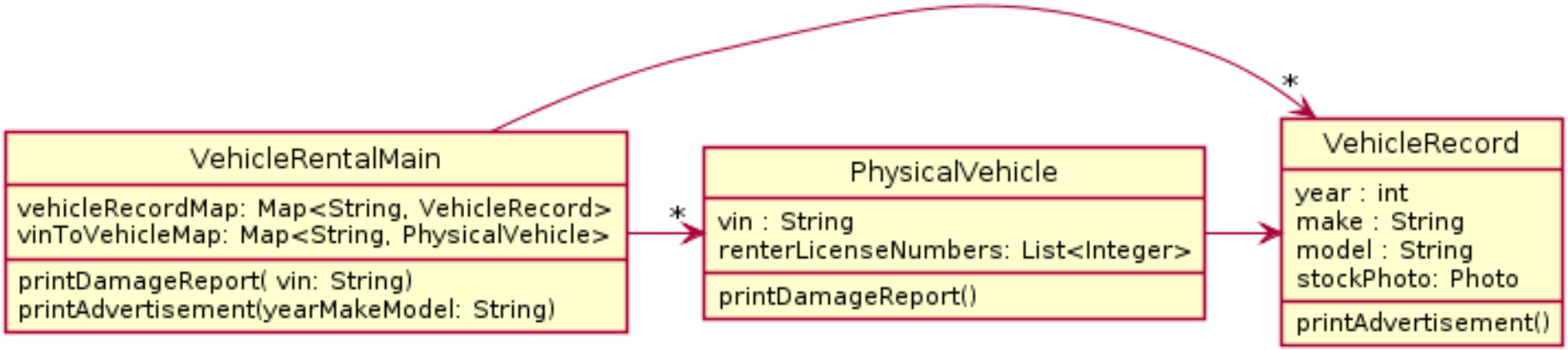
- A rental company has many vehicles that it rents. Vehicle have a year, make, and model and a stock photo advertising the vehicle on file. There are multiple vehicles that are the same year, make, model.
- However, additional information on the specific physical vehicles is also required. For instance, each physical vehicle has a vin (vehicle identification number unique to each), a description of any damage to it, and the driver's license numbers of everyone who has rented it.
- The company also needs to be able to print out the damage report of a vehicle given a VIN. The company also has to be able to print out an advertisement using the stock photo for a given year, make, and model.

Operable but poor solution



- What is wrong?

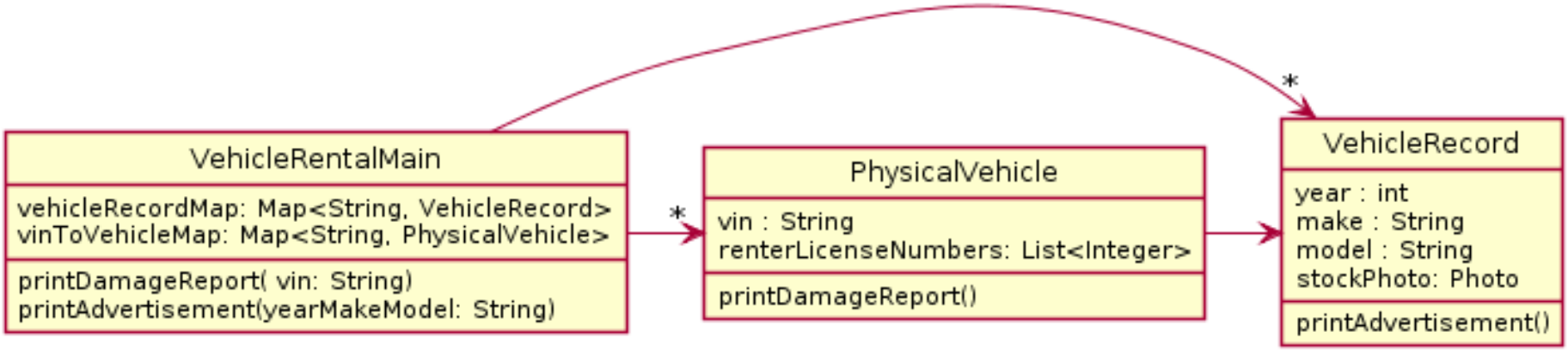
Better Solution



- There are two different things:
  - Actual physical vehicle
  - Records of specific vehicles
- Class has own behaviors (reports)
  - Used for specific purposes, specific data

Design Principles

3. Functionality should be **spread** throughout the system
   a) No single part of the system should get too large
   b) Each class should have a single responsibility it accomplishes



Crazy Eights

- Instructions are online
- This is to be done with a partner
  - These are assigned by the instructor
- If you have questions about the requirements, ask early!

Checkout CrazyEights Project

- Go to SVN repository view at bottom of workbench
  - Window➔ show view➔ Other➔ SVN➔ SVN Repositories
- Right click in SVN View, then choose New SVN Repository Location
  - http://svn.csse.rose-hulman.edu/repos/csse220-201810-crazy-eightsxx
  - Your team repository will be csse220-201810-crazy-eights-XX where XX is the team number
  - On Moodle, click on “Crazy Eights Team Assignments” to see to what team you have been assigned

UML for Crazy Eights Dealing

- Read the specification section for Crazy Eights called “Rules of the Game”
  - Don’t worry about the full requirements section right now
- With your partner, create a UML diagram that covers the initial dealing of player hands
  - Be sure you include main and enough information for each class to do its work
- When done, call me over to take a look
- Then we’ll discuss solutions

Work Time

- Work with your partner on the CrazyEights project
  - Get help as needed
  - ***Follow the practices of pair programming!***

- ***Don't do any of the work without your partner!***