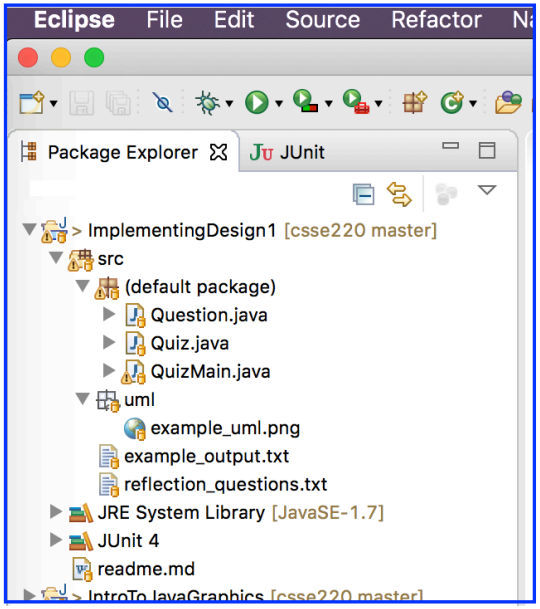
CSSE 220

Software Engineering Techniques
Design Principles
Encapsulation

Today's Agenda

- ImplementingDesign1
  1. Team | Pull to update repo
  2. Import from repo
  3. Upload submission to Moodle assignment



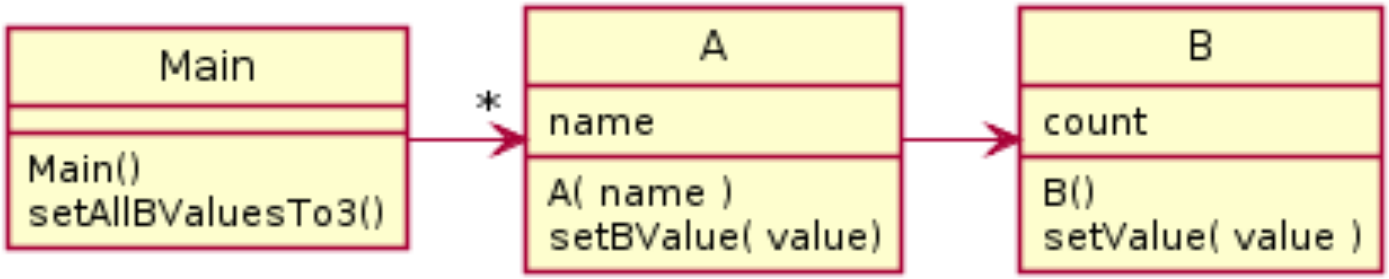
- Today: Focus on more OO Design principles:
  Spread functionality throughout the system
  Encapsulation

Turning in UML for ImplementingDesign1

- We have shown you a solution, but we want you to learn how to use tools to generate them
- http://www.plantuml.com/plantuml
  - Free web generator
  - Integration with Google Docs (search "get plantuml for google docs")
  - You will use this again in CSSE 374
- http://www.umlet.com/
  - Install program (works offline)

PlantUML

```
@startuml
skinparam style strictuml
class Main {
Main()
setAllBValuesTo3()
}
class A{
name
A( name )
setBValue( value)
}
class B{
count
B()
setValue( value )
}
Main -> "*" A
A->  B
@enduml
```



Moving on….

- More Object-Oriented Principles for Design
- Learn about next the few Design 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*.
  - Why do we care?
  - What does that mean?

Major Goals of ALL Program Design

Say someone has written a program that works and it has no bugs, but it is *poorly designed*.

- Why do we care?
- What does that mean?

Because:

- Software is written once
- But is read many times
  - for debugging
  - for adding/enhancing functionality

Major Goals of ALL Program Design

Because:

- Software is written once
- But is read many times
  - for debugging
  - for adding/enhancing functionality

This leads us to the goals

- There are two major goals:

1. Make software ~~easy~~ easier to understand

2. Make software so that it is ~~easy~~ easier to make changes

Q1

Principles of Design (for CSSE220)

- Make sure your design **allows proper functionality**
  - Must be able to **store required information** (one/many to one/many relationships)
  - Must be able to **access the required information** to accomplish tasks
  - Data should **not be duplicated** (id/identifiers are OK!)
- Structure design **around the data** to be stored
  - **Nouns should become classes**
  - **Classes should have intelligent behaviors** (methods) **that may operate on their data**
- Functionality should be **distributed efficiently**
  - **No class/part should get too large**
  - **Each class should have a single responsibility** it accomplishes
- **Minimize dependencies** between objects when it does not disrupt usability or extendability
  - Tell don't ask
  - Don't have message chains
- **Don't duplicate** code
  - Similar "chunks" of code should be **unified into functions**
  - Classes with similar features should be given **common interfaces**
  - Classes with similar internals should be simplified using **inheritance**

What are the principles?

3. Functionality should be **distributed efficiently**
   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?

Single Responsibility - Opeations

Called *procedural abstraction*

- In a client program
  - Don't duplicate a sequence of statements multiple times
  - Create a helper operation/method out of that sequence of statements
  - Where duplicated code would have been located, insert a call to the helper operation/method multiple times from client program
  - Helper has single responsibility, that sequence of statements

- Benefits:
  - Reduces size of client program
  - Improves readability of client program
  - Allows making a modification (e.g., a fix) in one place, i.e., in the helper operation

Single Responsibility for Classes

We want to do the same thing at the Class-level size

- The class is responsible for one thing:
  - A collection of related data fields
  - And the operations (methods) that work on those internal data fields

Counter Example:

- A class is responsible for multiple different tasks
  - The String class that also has methods for Math functions, e.g., cosine

A client program that wants only Math functions would also get the unwanted String class methods

Q2

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.

Skip

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)  );
```

Skip

Class sizes

- Why not have one monster *import* that brings in all of the Java packages in one Class?
  - We could just get anything we need done with one library!

One problem: Difficult to find what you're looking for, i.e., we'd need google search just for that one monster class

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 $12

Design a UML diagram to model this.

Guidance – Identify nouns and verbs

- Noun – often gives rise to a class, or fields within a class
- Verb – often gives rise to an operation that works on a object declared from the class

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 $12.

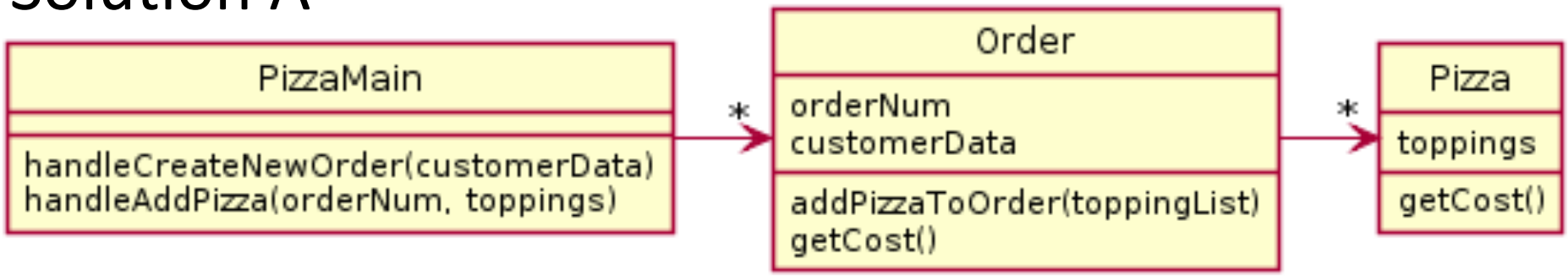
Lots of this prompt is spent on calculating the cost, noticing this key, for realizing where the complexity lies

UML

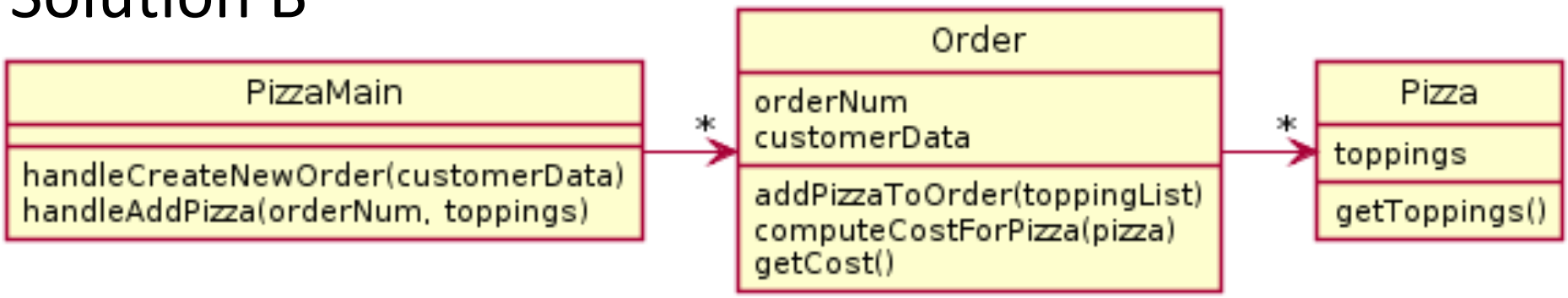
1. What classes did you have?
2. Where did you put “costOfPizza()”?

Evaluate Quality

Solution A



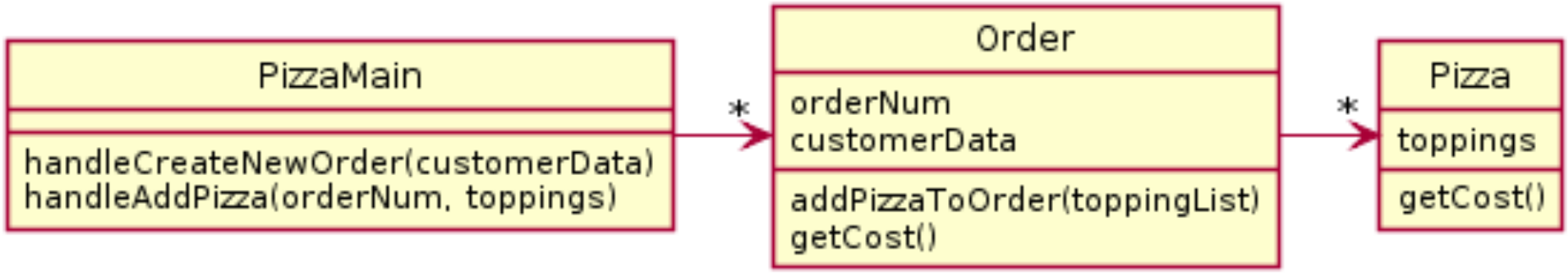
Solution B



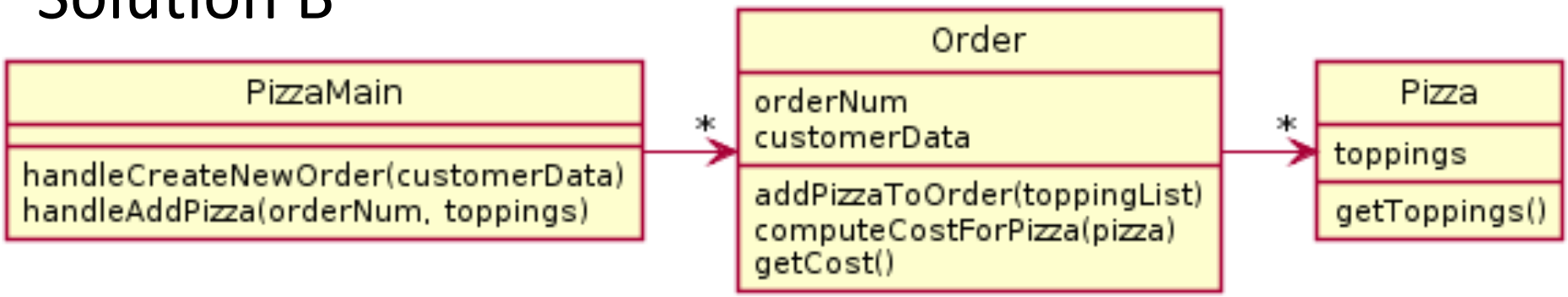
Which is better?

1 minute to evaluate - then we vote

Solution A



Solution B



Calculating costs could belong in either order or pizza.

- In Solution B, Order is doing a lot of stuff – Pizza is just a dumb data holder.
- So by spreading the functionality into the Pizza (in Solution A), we improve the design – Since an order might have multiple pizzas, let each pizza determine its own cost

Modify Software's Functionality

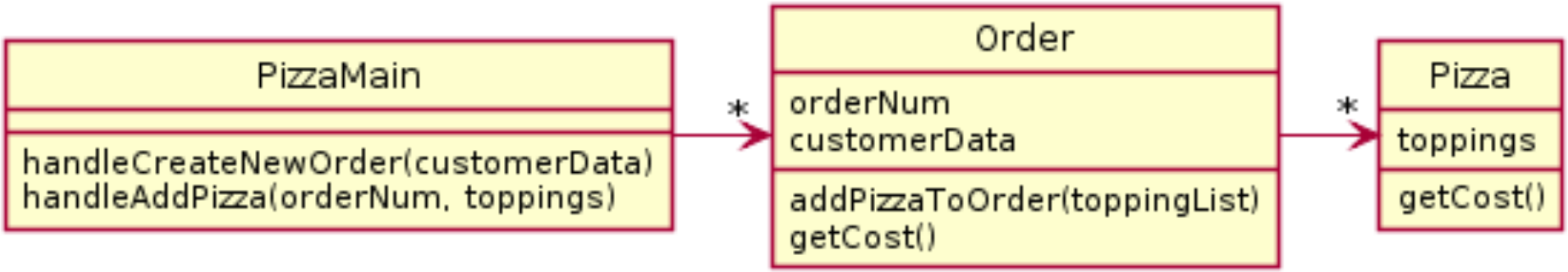
As mentioned earlier today:

- Software is written once and read many times
- So now we want to modify the software by adding additional functionality
- This will cause us to reread our design and software – if the design is "good" this task will be easier

Design Alternate Pizza Restaurant

(1) 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 a percentage off of the entire order.

(2) In addition, calculate how long it takes to create a pizza based on its size and toppings.

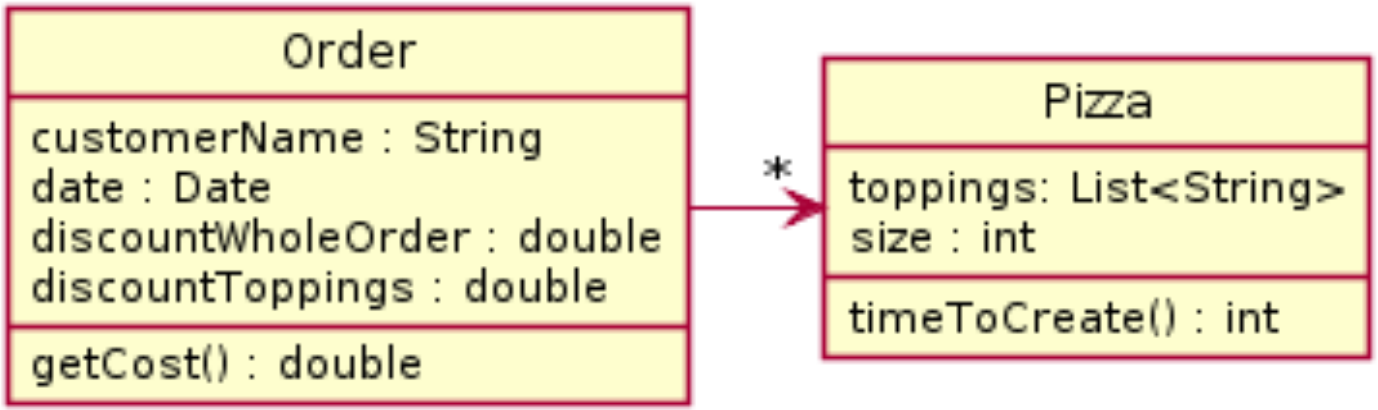


UML

1. What classes did you have?
2. Where did you put “getCost()”?
3. What about computing the time to create?

Do we need Coupon or Topping Classes?

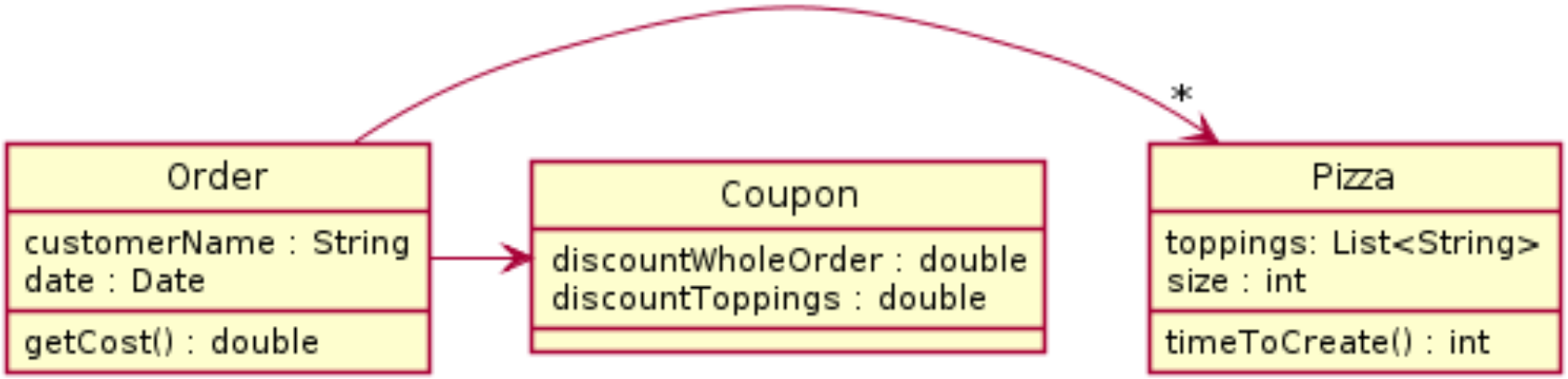
- It depends, do the classes do anything ***with*** their data, or are they just ***data classes*** that simply allow you to get and set values?



- Here: Order growing larger

One Solution

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



Split off the responsibility of storage of Coupon

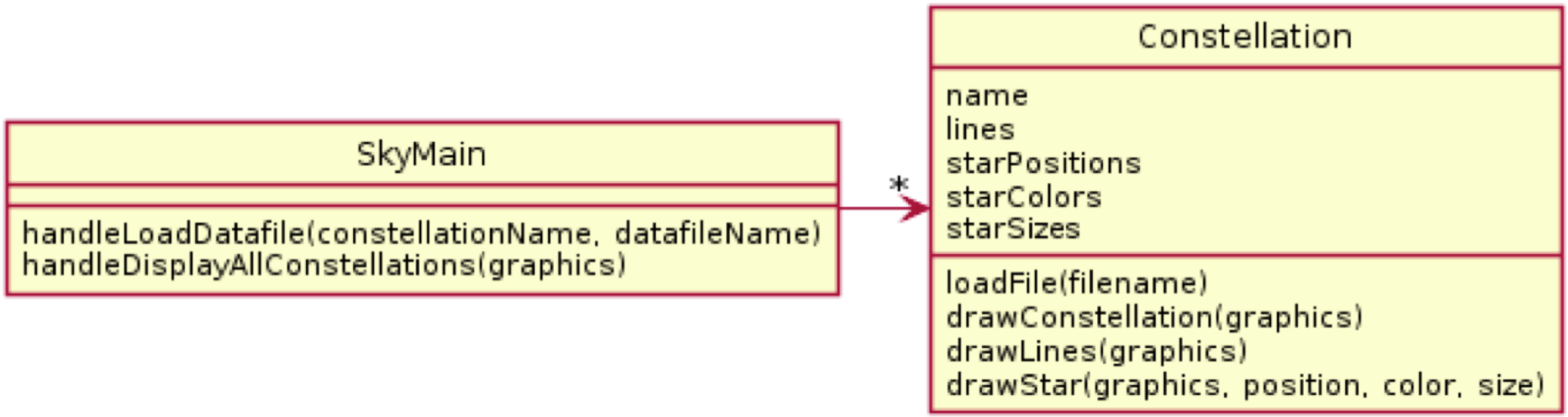
Rule of Thumb - Avoid Data Classes!

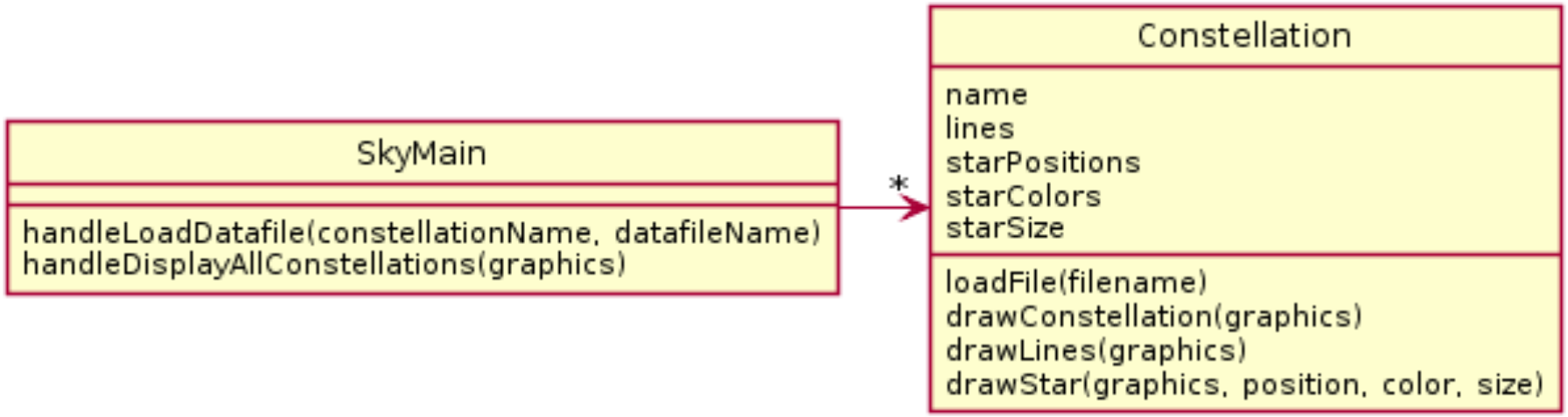
- A data class's methods comprise only *getters* and *setters*, no other methods exported
- Often, we think of Data Classes as violating a principle of OOD called ***encapsulation*** because they aren't in control of their own data – they are just dumb repositories for other classes to use
- Often you can improve a data class by finding functionality to add to them, and then add methods that capture that functionality
- But, if a grouping of pieces of data into a data class helps the system's overall design, then create a data class.

A particular program is designed to load constellations from datafiles and draw them on the screen. The datafiles 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. correct size, correct color).

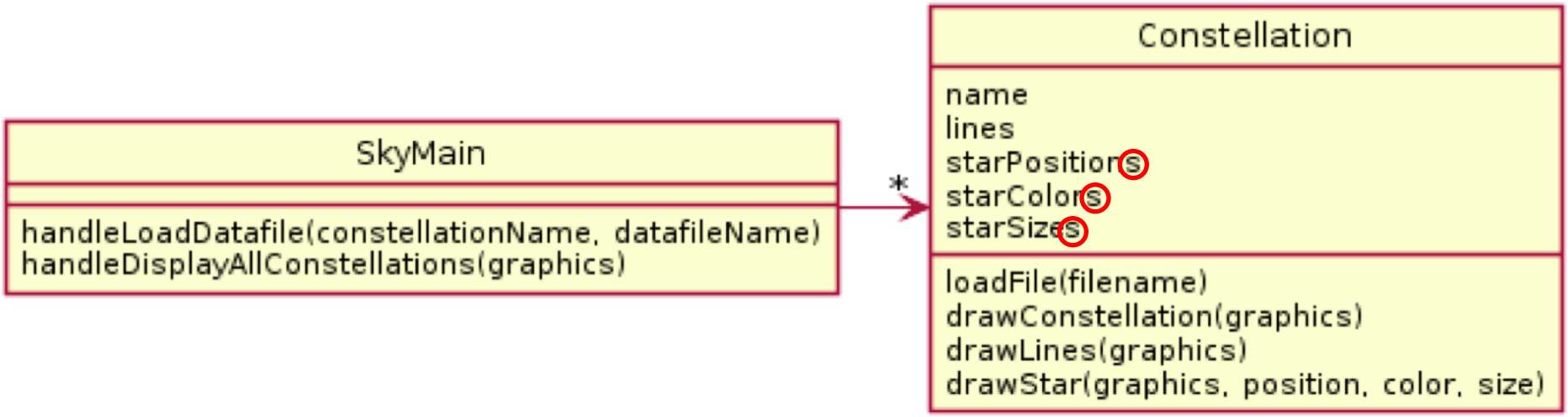
Explain how the "handle" operations might be implemented based on the UML diagram (below) and then propose a UML solution of your own.

5 minutes to do this





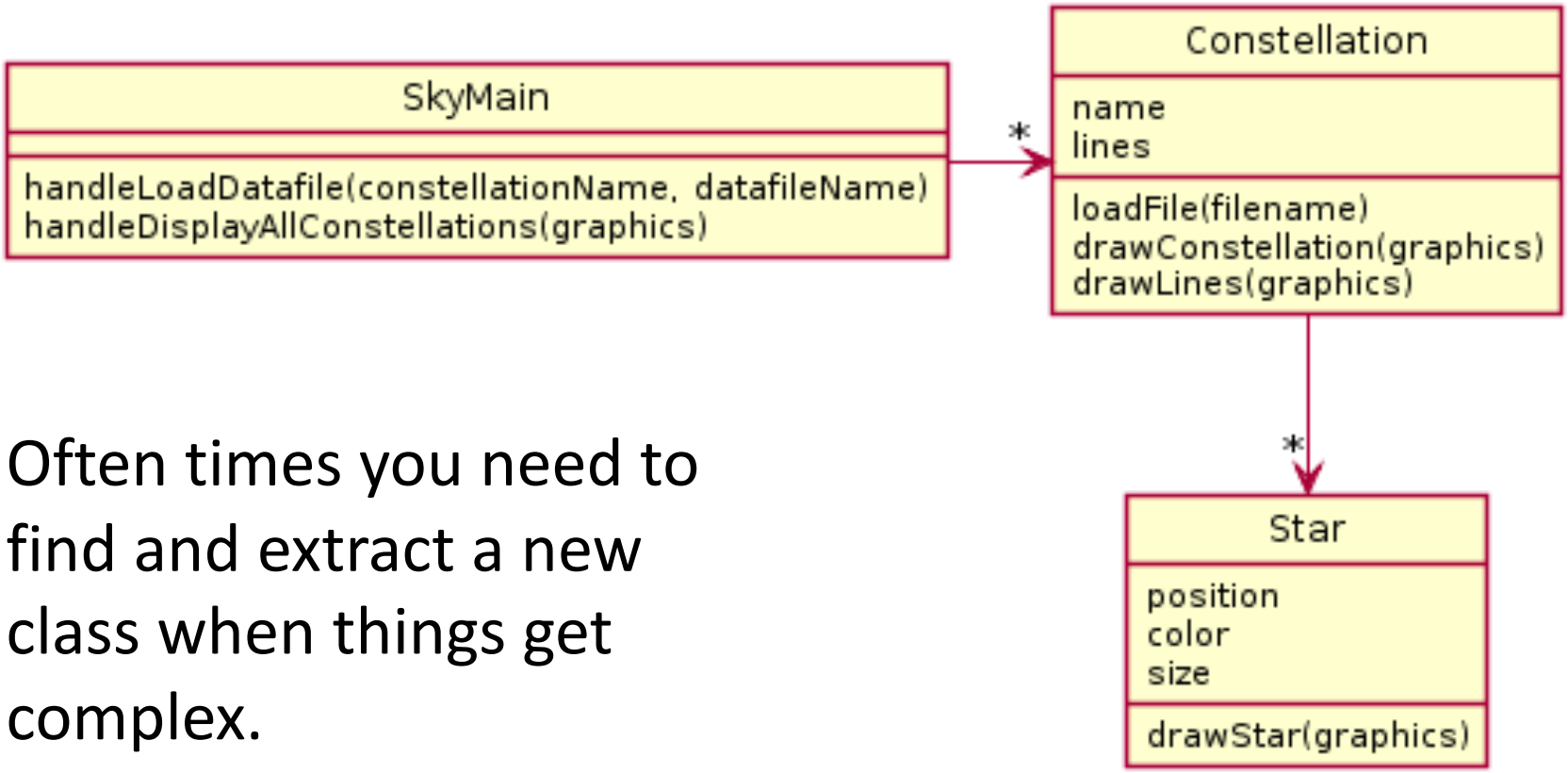
3a. Constellation does everything (except maybe the parsing done by main).



Notice the 3 fields with plural names

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

My solution



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

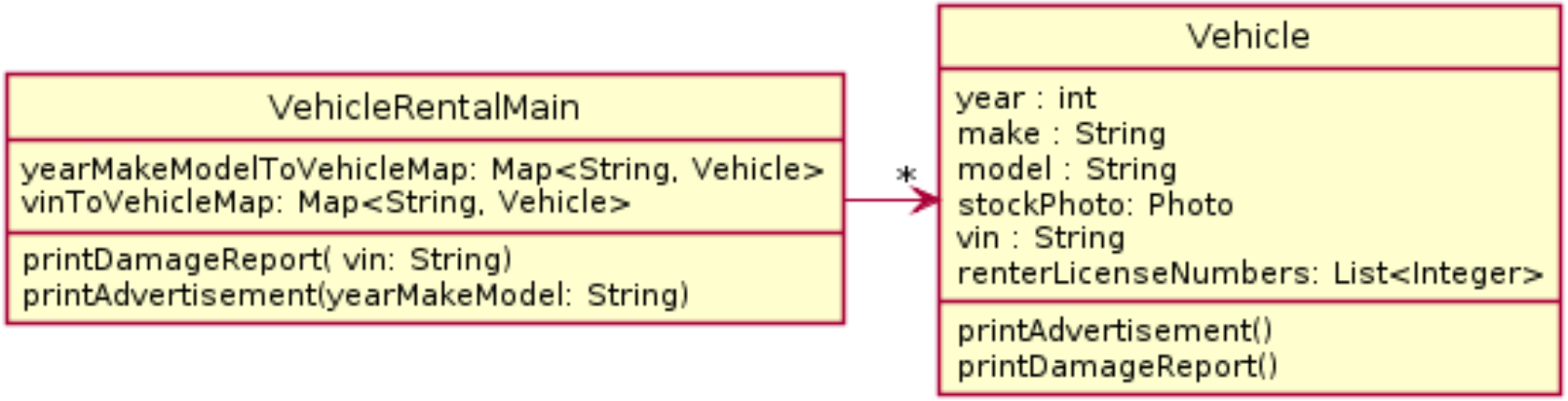
Your turn!

- Try to design UML for the following scenario

Rental Company

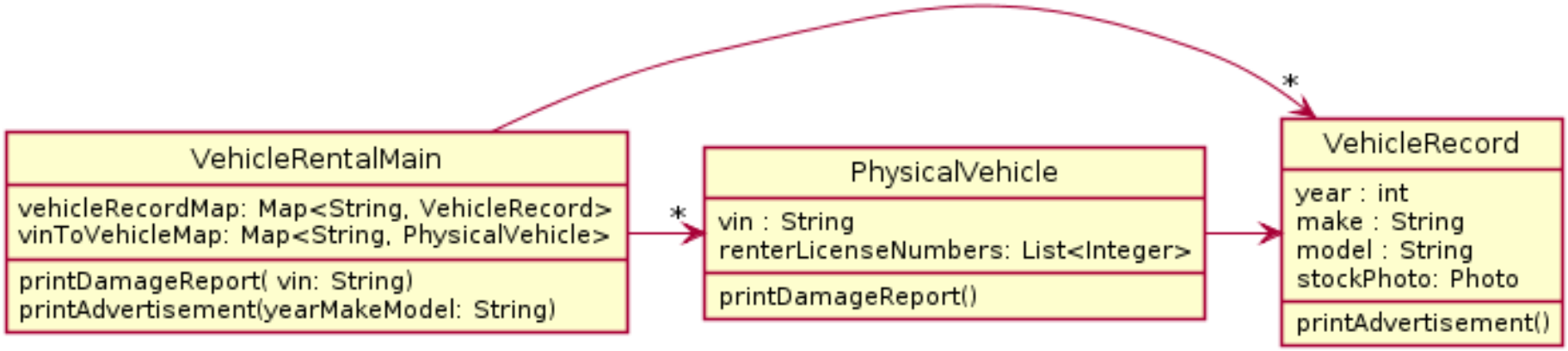
- A rental company has many vehicles that it rents. A vehicle has a year, make, and model and a stock photo advertising the vehicle on file. There are multiple vehicles that are the same year, make, model.
- However, additional information on the specific physical vehicles is also required. For instance, 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.
- 5 minutes – btw: none of these are being collected

Operable but poor solution



- What is wrong?
- Notice types have been included

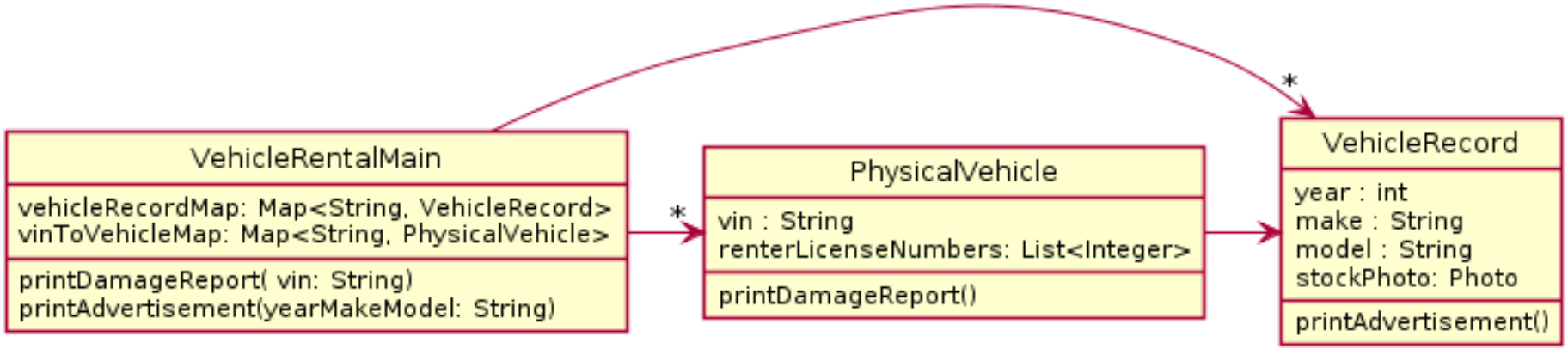
Better Solution



- 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 – i.e., do one thing, and one thing well



Encapsulation – Using a Class

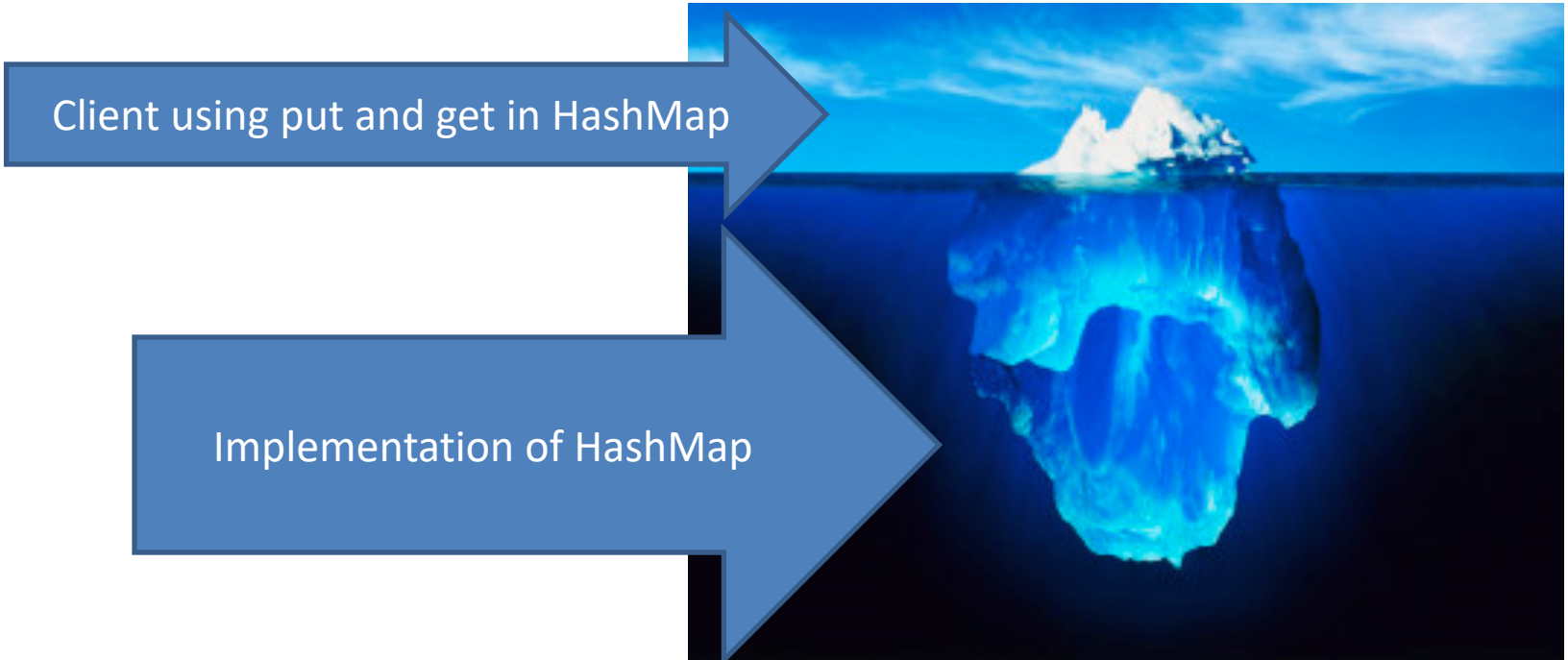
- Makes your program easier to understand by
  - Grouping related stuff together – data & operations

- Rather than passing around data, pass around objects created from a class:
  - The class provides a powerful set of operations that work on the internal data fields
  - The class protects the internal data from being used incorrectly – by using the *private* keyword on data (and on methods, i.e., helper methods)

Q2

Encapsulation

- Makes your program easier to understand by…
  - Allowing the client of your class to think about how *what* your class does, not *how* it does it



Encapsulation – Easier to Understand

1. Supports “separation of concerns”
   - Allows client to know: *What* it does
   - But client does not know: *How* it does it

2. Supports “information hiding”
   - Client does not know about implementation details, i.e., how the internal data is stored and how the methods make changes to the internal data is know know

Q3

Encapsulation – Easier to Maintain

Separation of Concerns and *Information Hiding*

- This allows you to more easily change internal implementation details with no ill effects on how client works
  - how data is stored
  - how methods modify data

Q4

Reminders

- Design Problems2 are due at end of this week – see schedule
- ImplementingDesign1
  - Make sure to upload the follow to Moodle by the due date:
    - **code**
    - **UML**
    - **Reflection**