CSSE 220

Software Engineering Techniques
Encapsulation
Coupling and Cohesion
Scoping

Please check out EncapsulationExamples from your SVN

The plan

- Software Engineering Techniques:
  - Pair programming
  - Version Control
- Learn 3 essential object oriented design terms:
  - Encapsulation (today’s topic)
  - Coupling
  - Cohesion

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

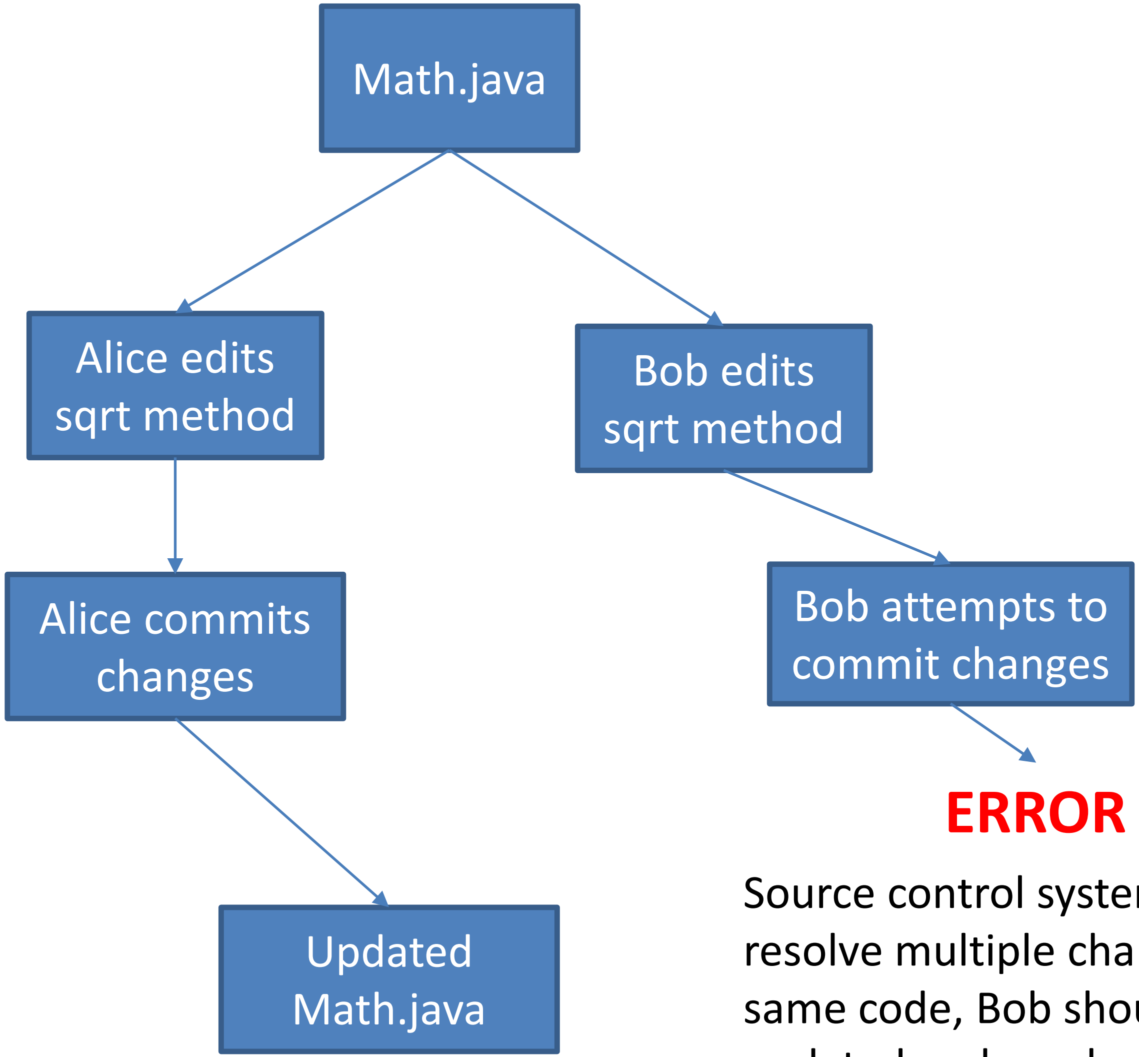
- Working in pairs on a single computer
  - The *driver*, uses the keyboard, talks/thinks out-loud
  - The *navigator*, watches, thinks, comments, and takes notes
  - Person who really understands should start by navigating ☺

- For hard (or new) problems, this technique
  - Reduces number of errors
  - Saves time in the long run

Q1

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

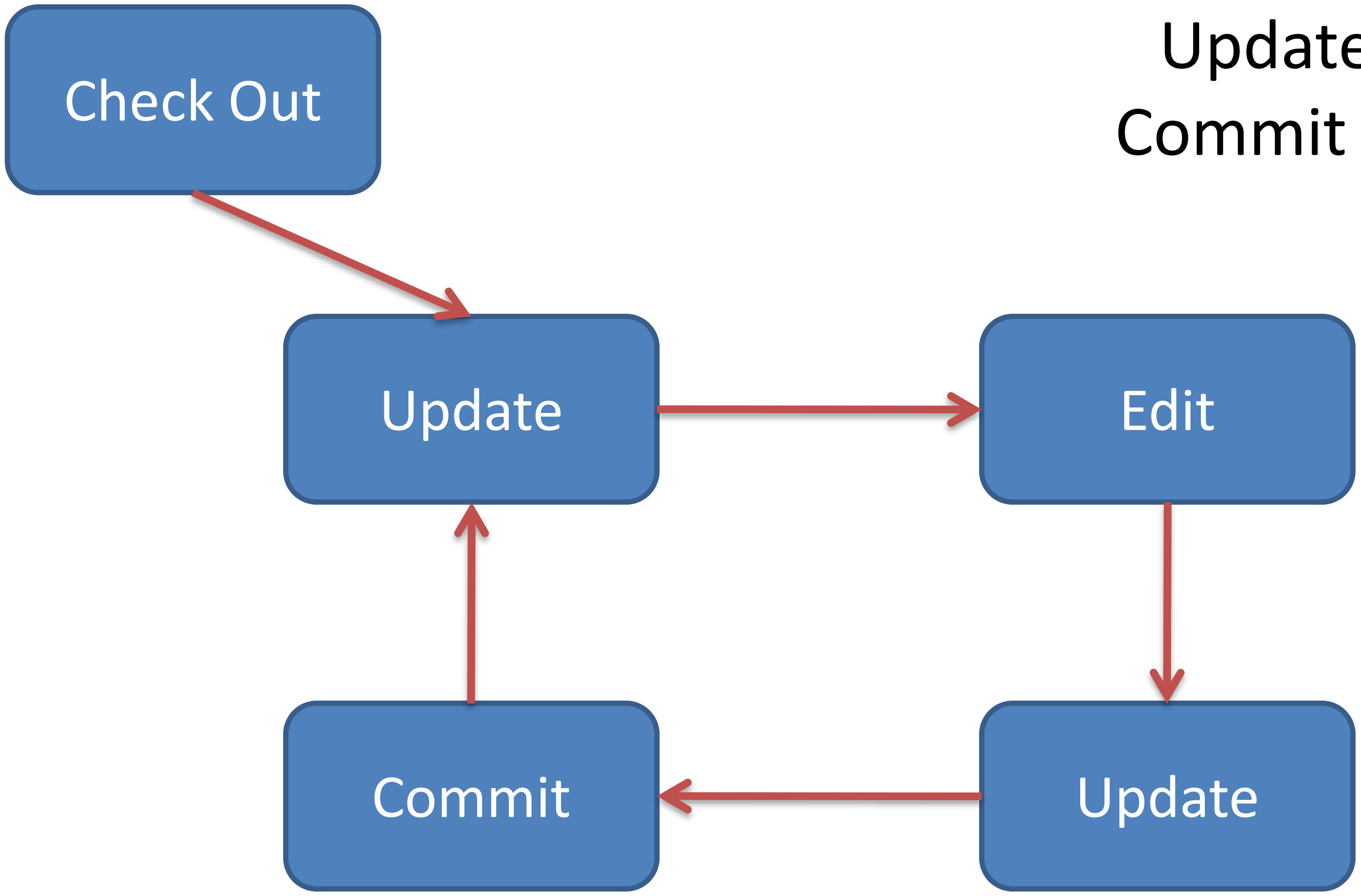
- **Version control tracks multiple versions**
  - Enables old versions to be recovered
  - Allows multiple versions to exist simultaneously

- **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….

- Learn 3 essential object oriented design terms:
  - **Encapsulation (today's topic)**
  - Coupling
  - Cohesion

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.

The Point 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

Encapsulation

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

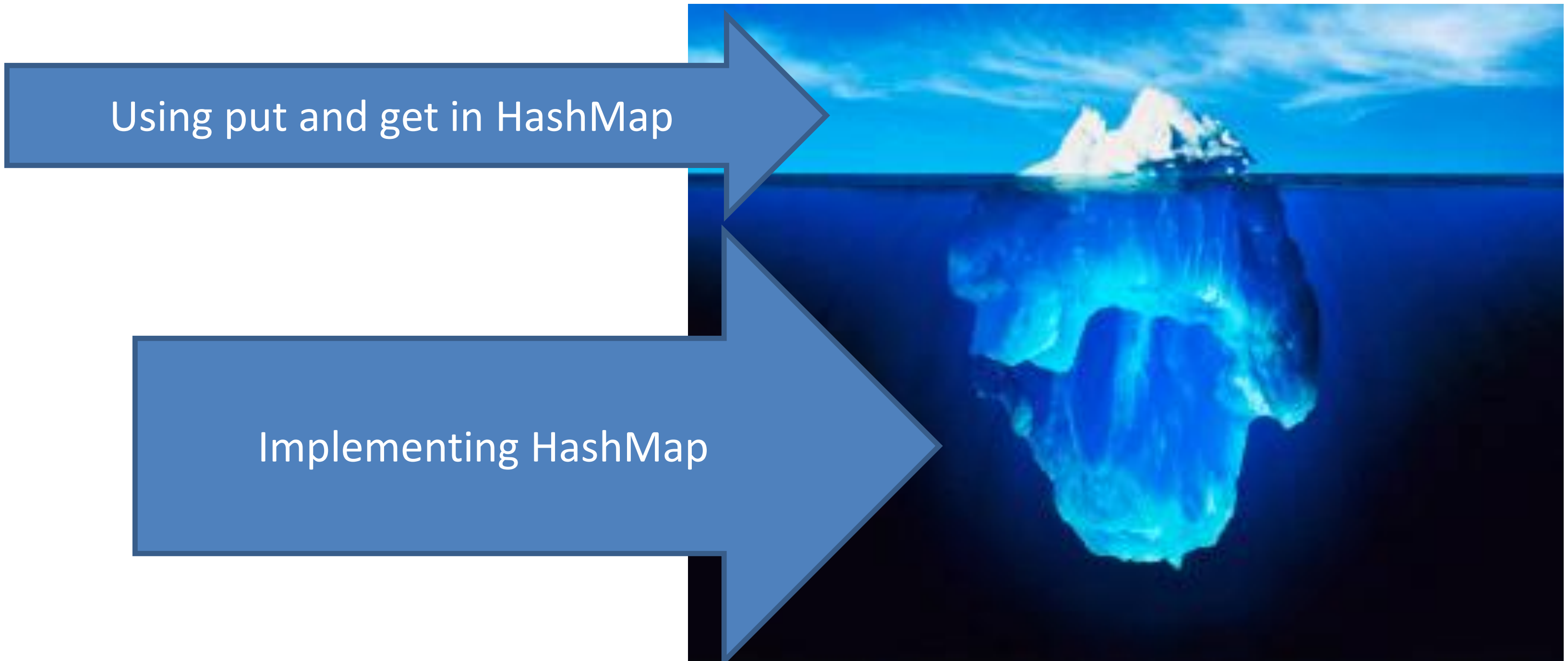
Encapsulation

- Makes your program easier to understand by
  - Grouping related stuff together

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?

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

City Temperature Activity

- I will split you into two groups
  - One group will solve the problem by creating a new class (see the Class Section example if you are unsure how to do that)
  - The other group will just write the code in main (see the Letters Example if you are unsure how to do that)
- If you finish early, try to solve it the other way too

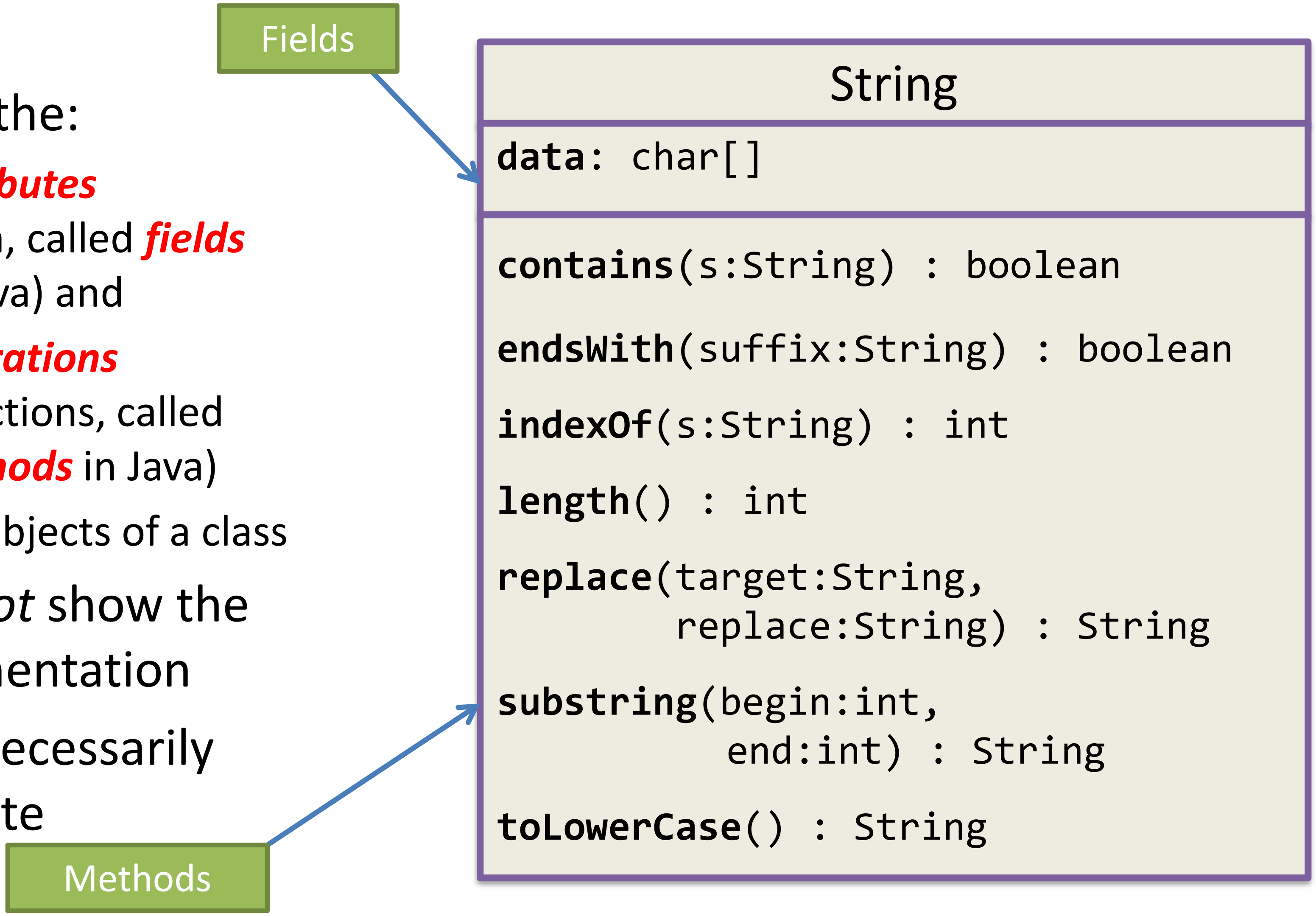
Encapsulation – a good thing?

- Note that we have the ability to change the representation of the CityTemperature class
  - but how important is that?
- Consider adding a bunch more statistics for each city (max, min, mode)
- Consider adding statistics overall (e.g. overall average)

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



```
data: char[]

contains(s:String) : boolean
endsWith(suffix:String) : boolean
indexOf(s:String) : int
length() : int
replace(target:String,
        replace:String) : String
substring(begin:int,
          end:int) : String
toLowerCase() : String
```

TwoVsTwo

- Look at the code to understand the problem
- Try to solve it using classes and encapsulation
  - Decide what classes/methods you would use
  (I used two new classes and TwoVsTwo main)
- Draw UML for the classes/methods

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 encapsulation because they aren’t in control of their own data – they are just dumb repositories for other classes to use

My TwoVsTwo Solution

- Let’s go through the code!

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/csse220-201730-crazy-eightsxx
  - Your team repository will be csse220-201730-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!***

Q5-6