CSSE 220

More interfaces

More recursion

More fun?

Check out *RecursiveHelperFunctions* and *DiscountInterfaces* from SVN

Exercise time

- Solve the sumArray function recursively
  - It's in the *RecursiveHelperFunctions* project
- You can work with friends, but each of you should get the code working on your own computer

Recursive Helper Functions – What, When, Why, How?

- What:
  - A recursive function that is called by another (non-recursive) function
  - The non-recursive function (the caller) doesn't do much
- When:
  - Additional parameters are needed
    - Often the initial function you're given is not in the ideal form for a recursive solution
  - Return values need to be updated

Recursive Helper Functions – What, When, Why, How?

- Why:
  - Makes function called by external code cleaner/easier to use
    - Does not rely on caller to understand how to initialize the information for the helper
  - Easier to understand by breaking problem down to smaller pieces
- How:
  - Methods named coolFunction & coolFunctionHelper
    - 90% of the code is in coolFunctionHelper

RecursiveHelperFunctions

- Solve the remaining problems
  - all the problems will require you to create a recursive helper function
- You can work with a friend but make sure both of you write the code
- If you finish early, work on the RecursionPractice homework assignment
  - none of these need recursive helper functions

Memoization

- Save every solution we find to sub-problems
- Before recursively computing a solution:
  - Look it up
  - If found, use it
  - Otherwise do the recursive computation
- Study the memoization code in the RecursiveHelperFunctions project

What if the recursive call isn't in the return?

- Let's start the quiz problem together, then you can finish it on your own.

DiscountInterfaces

- Get in groups of 2-3…no one working alone
- Understand the given code, the duplication, plus the additional features you will be adding
- Design a solution using interfaces and make a UML diagram describing it
  - If you're having trouble, first look at the different types of discounts in the current code (these are likely your classes).
  - Then look at what information each of those items needs to provide (these are likely the methods on your interface).
- Get myself or a TA to check out your UML
- Once we sign off – start coding
  - You only need 1 computer for this one.
  - Follow the steps in the Main comment

Hints

1) Your interface will likely be called Discount
2) You should have 2 classes implementing Discount, one for each of the current types of Discounts in the code
3) You'll need to add an ArrayList<Discount> (or some other storage method to main)