WES-CS GROUP MEETING #2

Exercise 1: Java Identifiers and Naming Conventions

What are the different kinds of things in a Java program that get names?

For each of the following,
- Say whether it's a valid Java name (identifier).
- If it is, does it follow the naming conventions discussed in class?
- If it does follow the naming conventions, what kinds of names should it be used for?
- If it doesn't follow the naming conventions, how could you fix it?

xyz	this Is Fine
WillThisWork?	public
letterCount	Public
fire-fly	number$1
MAX_VAL	Sandwich
mustang	abiggreenpickle
true	go#home
3people	big_bear

Exercise 2: Using Objects and Methods to Make Anagrams

This exercise is something like the "picasso" one we did last week: again, we'll assume that we have an object with methods that we can call, and we'll see how to use a sequence of method calls to perform some tasks.

This time, instead of an *Artist* object named *picasso*, we'll assume that we have a *Word* object named *word* that represents a sequence of letters. The methods that *word* provides each rearrange the letters in the sequence in some way. The methods all have one or two integer arguments that specify *positions* in the sequence. Because Java (and many other modern programming languages) starts counting from zero, we'll do that for the positions, too. For example, if *word* represents the sequence of letters CANDY, we'll say that the letter C is in position zero, the letter A is in position one, and so on.

Below are descriptions of the methods; because they all perform tasks (rather than computing and returning values) the descriptions all start with the keyword *void*.

void moveToFront(int pos)
   Move the letter in position *pos* to the beginning of the sequence (i.e., to position zero).

void moveToEnd(int pos)
   Move the letter in position *pos* to the end of the sequence.

void swap(int pos1, int pos2)
   Swap the letter in position *pos1* with the letter in position *pos2*.

void reverse(int start, int finish)
   Reverse the order of the letters in positions *start* to *finish*.

Part (a).

Here are some examples of method calls; in each case, the letter or letters that will be moved are shown in bold in the first column. Work with a partner to write the results in the third column, then compare your answers with another pair.

Original sequence in `word`	Method call	Sequence after the call
POST	`word.moveToFront(2)`	
STOP	`word.moveToEnd(0)`	
TO**P**S	`word.swap(0,2)`	
S**PORT**ED	`word.reverse(1,4)`	

Part (b).

Now you're going to start with the name of your Team Leader (DAVIDHOFFERT) as the value of *word*, and your task is to execute one of the sequences of method calls below to discover an interesting anagram of your Team Leader's name.

SEQUENCE 1 | SEQUENCE 2 | SEQUENCE 3

word.moveToEnd(3)   word.moveToEnd(5)   word.moveToFront(8)
word.moveToEnd(2)   word.moveToEnd(3)   word.moveToFront(8)
word.reverse(0,2)   word.moveToEnd(2)   word.moveToFront(8)
word.swap(3,4)   word.moveToFront(1) word.moveToFront(8)
word.swap(4,11)   word.reverse(1,3)   word.moveToFront(7)
word.swap(5,7)   word.swap(5,10)   word.reverse(6,10)
word.swap(6,8)   word.swap(6,10)   word.swap(0,1)
word.swap(8,10)   word.swap(7,8)   word.swap(2,5)
word.swap(9,10)   word.swap(8,9)   word.swap(3,5)

Part (c).

Find an anagram for your partner (you'll get better anagrams if you use their whole name, not just their first name). You can find one yourself, or use the anagram server at:

http://www.wordsmith.org/anagram/

Assume that *word* originally represents your partner's name; write down a sequence of method calls that transforms the name into your anagram, then give the sequence to your partner so that they can use it to discover your anagram.

Exercise 3: Algorithms

An *algorithm* is a clear, step-by-step description of how to perform a task. For example, here's an algorithm for determining whether a positive integer N is prime:

Step 1: Let k equal 2.

Step 2: If k is greater than $N/2$, stop: N is prime.

Step 3: If N is evenly divisible by k, stop: N is not prime.

Step 4: Let k be $k+1$.

Step 5: Go to Step 2.

Part (a).

Below is an incomplete algorithm for reversing the letters in word (for example, turning STOP into POTS, or HELLO into OLLEH) using only the *swap* method.

Step 1: Let j equal 0 and let k equal the number of letters in the word -1.

Step 2: If _____ stop.

Step 3: word.swap(j, k)

Step 4: Let j equal _____ and let k equal _____.

Step 5: Goto Step 2.

Your job is to fill in the blanks.

Part (b).

Now assume we only have the *moveToFront* method. Write an algorithm for reversing the letters in a word.

If you finish early, try it again assuming that you only have the *moveToEnd* method.

Exercise 4: Methods that return values

The *Artist* methods we used last week, and the *Word* methods we just used to make anagrams all performed actions. A method can also compute and return a value. For example, suppose we have several *Dessert* objects that each represent the recipe for a chocolate dessert, and that each *Dessert* object has the following methods (each of these methods computes and returns an integer value, so the descriptions all start with the keyword *int*):

int numEggs()
> Returns the number of eggs needed to make 1 batch of this dessert.

int numCupsFlour()
> Returns the number of cups of flour needed to make 1 batch of this dessert.

int numOzChocolate()
> Returns the number of ounces of chocolate needed to make 1 batch of this dessert.

We'll also assume that we have an *Inventory* object that represents the ingredients we have in the house, and that an *Inventory* object provides the following method:

int numUnits(String ingredientName)
> Returns the number of units (cups, ounces, or whatever is appropriate) of the given ingredient that we have in the house.

Now we'll assume we have three *Dessert* objects named *brownies*, *cookies*, and *fudge*, and one *Inventory* object named *onHand*. If we want to know how many eggs are needed to make 1 batch each of brownies, cookies, and fudge, we could evaluate the following Java expression:

brownies.numEggs() + cookies.numEggs() + fudge.numEggs()

and compare the result (an integer value) to the value of the expression

onHand.numUnits("eggs")

which would also be an integer value -- the number of eggs we currently have in the house. If the value we got by adding the results of the three calls to *numEggs* is less than or equal to the value returned by the call to *numUnits*, we can make all three desserts!

Part (a).

What expressions could we compare to find out whether we have enough cups of flour in the house to make *two* batches of fudge?

How about two batches of fudge *and* three batches of brownies?

Part (b).

Suppose we want to make as many batches of cookies as possible. We know that we have plenty of eggs and flour, but we're a bit short on chocolate. What expression would tell us how many batches of cookies we can make? (Hint: If we have N ounces of chocolate in the house, and each batch of cookies requires M ounces of chocolate, then what expression gives the number of batches of cookies we can make?)

Exercise 5: Logical Thinking

Here are two logic puzzles. Try solving them now. If you don't finish, you can keep working on them during the week if you want to. We'll talk about the answers next time.

1. What are the ages?
   Two mathematicians are sitting together in a building. They don't have anything to do at the moment, so one says to the other, "Try guessing the ages of my three children. I'll give you a hint - the product of their ages is 72." The other mathematician says, "That's not enough information. Tell me more." The first mathematician says, "Their ages add to be the number of this building." The other mathematician goes outside, looks at the building number, comes back, and says, "That still isn't enough information. Tell me more." The mathematician says, "My youngest child's name is Anne."

   What are the ages of the children?

2. How many handshakes?
   A woman and her husband attend a party with four other couples. Some people shake hands with other people. Of course, no one shakes their own hand or the hand of the person they came with. When the woman asks the other (9) people present how many different people's hands they shook they all gave a different answer. Question (this is *not* a trick!): How many different people's hands did the woman's husband shake?