

LCS-ES Group Meeting #5

Exercise 1: Syntax

For this exercise, you will divide into groups of 2 or 3 to play *concentration*, which will help you review the syntax for different Java constructs.

Each group will get one set of cards; the green ones have an English description of some kind of Java code, the yellow ones have a definition of the syntax for that code, and the pink ones have an example of that kind of code. First, look at the cards and decide which ones match (note that multiple cards from one group may match a single card from another group).

Now put all the cards face down and take turns turning over two cards of different colors at a time. If you turn over matching cards, you take them and go again. The game ends when someone has 4 matching pairs.

Exercise 2: Writing a Java Method

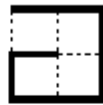
Last week you did a long exercise involving the Artist class. One of the problems was to write code to draw a clockwise, square spiral in an n-by-n square. You may not have written actual code or you may have written code that works only for a particular n. This week we will write the general method. The method header is given below:

```
public void drawSpiral(int n)
```

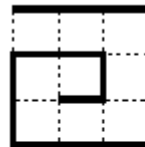
And here are the pictures of some spirals to remind you what they look like.



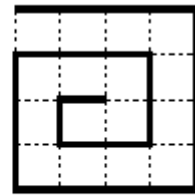
1-by-1 inch square



2-by-2 inch square



3-by-3 inch square



4-by-4 inch square

Exercise 3: Google search with logical operators

Google searches for Webpages that satisfy a condition entered by the user.

How does Google handle interpret search conditions? If you enter

health coffee

in the textbox, does Google search for all Webpages that have **both** words or either word?

Does entering “health and coffee” alter the results?

Does entering “health or coffee” alter the results?

Is Google case-sensitive in dealing with logical operators?

Suppose you enter “health” “coffee” is this equivalent to using a logical operator? If so, which one?

Is there a NOT (!) logical operator in Google?

Exercise 4: The String class

The String class has many useful methods including the following:

```
String substr(int beginIndex, int endIndex)
```

Returns a new string that is the substring of this string that starts at `beginIndex` and ends at index `endIndex - 1` (the index of the first character is 0). Error if `endIndex` is greater than the length of this string.

```
int indexOf(int ch)
```

Returns the index within this string of the first occurrence of `ch`. If no such character occurs in this string, then -1 is returned.

```
int compareTo(String anotherString)
```

Returns 0 if this string is the same as `anotherString`; returns a negative integer if this string comes before `anotherString` in lexicographic order (dictionary order); returns a positive integer if this string comes after `anotherString` in lexicographic order.

```
boolean equals(Object anObject)
```

Returns true if `anObject` is a String that represents the same sequence of characters as this string; otherwise returns false.

To get some practice using the String methods, play the following game (in pairs). (The cards to use are on the last four pages; they need to be cut up into individual cards.)

- Each person thinks of a word, 3 to 5 letters long. Whoever guesses their opponent's word first wins!
- Each person takes 2 cards.
- Alternate turns. When it's your turn, pick one card, then play one of your 3 cards. If it has a blank (e.g., `s.indexOf(__)`), you get to fill in the blank with whatever you want. Your opponent tells you what value their word, `s`, would return if the method call on your card were made. For example, if your opponent is thinking of the word "hat" and you play `s.substr(1, 2)`, your opponent must say "a"; if you play `s.substr(2, 4)`, your opponent must say "error" (because "hat" has only 3 letters).
- You win if you play an `s.equals` card and your opponent says "true", or if you play an `s.compareTo` card and your opponent says 0 (i.e., you guessed their word).