Review day!

Resources

Read my Arrays and other Collections (at the water park) for a visual analogy of the different types of collections.

See my Stuff from Early Module One presentation

Vocabulary

Crowdsourcing a list of terms and concepts

Abstraction
Loose Coupling
Method
Constructor
New
Static
Final
Encapsulation
Polymorphism
Parsing
Overflow & Truncation
Pointer or Reference
Null Pointer
Null
Void
Reference vs. Primitive
Scope
Parameter
Argument
Chaining
Immutable
Function (method with a return value)

Code Talk

Command Line Programs

Some examples, gosh darn it, even if we have to write them ourselves.

I Have 99 Problems (But Only Limited Time)

Let's take a look at some of the problems from the past two weeks

```
/*
Given a string of even length, return a string made of the middle two chars,
so the string "string"
yields "ri". The string length will be at least 2.
middleTwo("string") → "ri"
middleTwo("code") → "od"
middleTwo("Practice") → "ct"
*/
public String middleTwo(String str) {
  // Instantiate the thing to return
  String middle;
  int offsetToMiddle = (str.length()-2)/2;

  // body
  middle = str.substring(offsetToMiddle, offsetToMiddle+2);


  return middle;
}
```

```java
/*
Given a string, return the count of the number of times that a substring
length 2 appears in the string and
also as the last 2 chars of the string, so "hixxxhi" yields 1 (we won't count
the end substring).
last2("hixxhi") → 1
last2("xaxxaxaxx") → 1
last2("axxxaaxx") → 2
*/
public int last2(String str) {
  int last = 0;

  // Check the parameters for validity
  if (str == null || str.length() < 3) {
     return 0;
  }

  // Save the last pair for comparison
  String lastPair = str.substring(str.length()-2);

  // Loop through comparing, but stop before the end so we don't include the
last pair
  for (int i = 0; i < str.length()-2; i++) {
     String thisPair = str.substring(i, i+2);
     if (lastPair.equals(thisPair)) {
        last++;
     }
  }

  return last;
}
```

```
/*
Given two lists of Integers, interleave them beginning with the first element
in the first list followed
by the first element of the second. Continue interleaving the elements until
all elements have been interwoven.
Return the new list. If the lists are of unequal lengths, simply attach the
remaining elements of the longer
list to the new list before returning it.
interleaveLists( [1, 2, 3], [4, 5, 6] )  ->  [1, 4, 2, 5, 3, 6]
*/


public List<Integer> interleaveLists(List<Integer> listOne, List<Integer>
listTwo) {
  List<Integer> newList = new ArrayList<>();

  // Validation of parameters
  if (listOne == null || listOne.size() == 0) {
     return listTwo;
  }
  else if (listTwo == null || listTwo.size() == 0) {
     return listOne;
  }

  // Interleave, skipping elements if the list is too short
  for (int i = 0; i < listOne.size() || i < listTwo.size(); i++) {
     if (i < listOne.size()) {
        newList.add(listOne.get(i));
     }

     if (i < listTwo.size()) {
        newList.add(listTwo.get(i));
     }

  }

return newList;
}
```