CSSE 220 Day 3

Strings, Arrays,
Object intro

Announcements

- Don't forget the Python *vs.* Java comparison document in the Resources folder on the Web.
- Any questions on:
    - Syllabus
    - Java
    - Reading from the textbook
    - Homework
    - etc.

In all your code:

- Write appropriate comments:
  - Javadoc comments for public fields and methods.
  - Explanations of anything else that is not obvious.
- Give explanatory variable and method names:
  - Use name completion in Eclipse, Alt-/ to keep typing cost low and readability high
- Use local variables and static methods (instead of fields and non-static methods) where appropriate.
  - "where appropriate" includes any place where you can't explicitly justify doing otherwise.
- Use Ctrl-Shift-F in Eclipse to format your code.

HW1 Program Solutions

```
public static int maxOfThree (int a, int b, int c) {
    int larger = (a>b) ? a : b;
    return (larger > c) ? larger : c;
}

 public static int monthsToReach(double target,
                                 double monthlyDeposit,
                                 double annualInterestRate) {
     double total = 0;
     double monthlyInterestRate = annualInterestRate / 12;
     int month = 0;
     while (total < target) {
         double interest = total * monthlyInterestRate;
         total = total + interest + monthlyDeposit;
         month++;
     }
     return month;
 }
```

Primitives *vs.* Objects

- What is the main difference between primitive types and object types?
- Consider these two code snippets (assume that we have import java.awt.Point; at the top of the file.

```
int a = 3, b = 2;
b = a;
a = 4;
System.out.println(a + "  " + b);

Point p1 = new Point(4, 5), p3, p4;
p3 = new Point(p1.x, p1.y);
p4 = p1;
System.out.println("p3==p1?  " + (p3==p1) + "     " +
                  "p3.equals(p1)?  " + p3.equals(p1));
p3.y = 500;
p4.x = 100;
System.out.println(p1 + " " + p3 + " " + p4);
```

Characters and Strings

- char is a (primitive) integer type that represents a single character.
-
```
char c1='a', c2 ='\n', c3='\\', c4=65;
System.out.printf("*%c*%c*%c*%c*%c*\n",
                  c1, c2, c3, c4, c4+1);
```
- output from the above:
```
*a*
*\*A*B*
```
- String is an object type that represents a sequence of zero or more characters.
- 'a' vs "a". Draw the pictures.

The String class

- A Java String object is immutable.
- I.e., once created, you cannot change its length or the individual characters in the String.
- String constants are enclosed in double quotes.
- + is the concatenation operator
- Every class has a toString() method, which returns a String representation of an object.

String Declarations and Operations

```
String s1 = new String();
System.out.println("*" + s1 + "*");
String s2 = "";
System.out.println("1 ==? " + (s1==s2));
String s3 = "abc";
String s4 = "ab" + "bc".substring(1);
String s5 = "ab".concat("c");
System.out.println("2 ==? " + (s3==s4));
System.out.println("3 ==? " + (s4==s5));
System.out.println("1 equals? " + (s1.equals(s2)));
System.out.println("2 equals? " + (s4.equals(s3)));
System.out.println("3 equals? " + (s4.equals(s5)));
s1 = "AbCdEfG";
System.out.println(s1.length() + " " + s1.charAt(3) + " " +
    s1.toLowerCase() + " " + s1.substring(2, 5) + "\n" +
    s1.replace("bC", "XYZ")  + " " + s1.indexOf('C') + " " +
    s1.substring(0,3).equalsIgnoreCase(s3));
```

Later: look at the String.format () method.

It is like printf(), but it returns the formatted string instead of printing it.

Some static String Methods to Write

- printReverse(s) prints the String s in reverse order.
- reverse(s) returns a String that is the reverse of s.
- multiply(s, i) returns a String that contains i copies of s, where i>=0.

Array Basics

```
int [] nums = new int[5];
for (int i=0; i<nums.length; i++)
   nums[i] = i*2 + 1;
System.out.println(nums);

for (int j : nums)
   System.out.print(j + " ");

System.out.println();
int [] moreNums = nums;
moreNums[3] = 100;
System.out.println(nums[3]);
moreNums = nums.clone();
moreNums[2] = 1000;
System.out.println(nums[2]);
int [] stillMore = {2, 4, 6, 8};

Point [] pts = {new Point(0,0), new Point(3,4), new Point(5, 6)};
Point [] pts2 = new Point[6];
pts[1].translate(-2, 2);
System.out.println(pts[1]);
```

Write these methods:

public static void printIntArray(int[ ] a)

public static void reverseObjArray(Object[ ] a)

How many objects are created by the declaration of pts2?

2-dimensional and ragged arrays

```
int[][] table = new int[4][3];
int[][] table2 = {{1, 4}, {2, 3}, {-2, 4}};
int[][] ragged = new int[4][];
ragged[0] = new int[2];
ragged[0][1] = 4;
ragged[1] = new int[1];
ragged[1][0] = 6;
```

- Practice later: write methods to find the sum of the elements of a 2D ragged array of ints

"Resize" an array

- An array is inherently fixed-length.
- But we can get the effect of a "growable array":
  - Have two variables, arr, and size.
  - initialize arr to be an array of 5 elements
    - I choose 5 because that is what Mark Weiss does.
  - When we want to add a new element at the end:
    - if size == arr.length
      - call resize to give us an array twice as big.
    - Put the new element in arr[size] and increment size.
    - Code:

```
if (size == arr.length)
   arr = resize(arr, size, size*2);
arr[size++] = newValue;
```

Write resize( )

Why *2 instead of +1?
You'll answer that question mathematically on the first day of 230 (if not sooner)

ArrayList: a class that implements a resizeable array-like structure

- Full name: java.util.ArrayList
- Methods include
  - add(element)
  - add(index, element)
  - get(index)
  - size()
  - clear()
  - remove(object)
  - remove(index)
  - set(index, element)
  - toArray()
  - trimToSize()