Announcements

Arrays OWL assignment due Thursday
Exams back Tomorrow
Another programming assignment coming

I’ll give the “what should you make of your exam score” speech on Thursday
We’ve seen Java at two levels:

• the statement level - mechanisms for getting specific, often low-level jobs done- assignment, println, etc.

• the (class and) object level - mechanisms for modeling things (objects) according to an” Objects” model -(repositories of state) - served by methods (machinery for realizing behaviors) scheme

Now we’re back to a new and very important idea in statement-level thinking: arrays.

Arrays give us a new way to think about variables.
Think about: students in a class; seats on an airplane, rooms in a motel, positions in deli line

- Many similar, nearly anonymous, variables required
- There’s an indexing scheme for locating / identifying the variables in question:

Student 7
Seat 23B
Room 201
Deli-line position 77

- some indexing schemes are more natural than others
- some are two-dimensional
We seek a scheme that

• Let’s us define gobs of variables all at once
• Allows us to access / mess with / update these variables as a group in a systematic way
Note that if we have five kids (Infants) in a daycare ctr:

kid0, kid1, kid2, kid3, kid4

There’s no obvious way to wrap them in a loop and make each one month older:

```javascript
for (in j = 0; j < 5; j++)
    kidj.anotherMonth();
```

As stated, is a catastrophe
Variables in algebra

$x_0 \ x_1 \ y_3 \ and \ so \ forth$

Java notation just a variant:
$x[0], \ x[1], \ y[3]$

Algebra: $x_0 = 2 \times x_1$

----------------------------------

Java: $x[0] = 2 \times x[1]$;
Defining array variables

```
int[] nums = new int[6];  // array of 6 ints

nums[3] - the third one; at index 3

Infant[] kids = new Infant[5]; // 5 Infants

kids[0] - the zeroth one; at index 0
```
Recall the catastrophe:

```java
for(int j = 0; j < 5; j++)
    kidj.anotherMonth();
```

But this works!

```java
for(int j = 0; j < 5; j++)
    kid[j].anotherMonth();
```
Make value at index 3 an 8:  \( \text{nums}[3] = 8; \)
public class ArrayTest{
public static void main(String[] args){
    int[] firstArray = new int[10];
    for(int j = 0; j < 10; j++) {
        firstArray[j] = j*j;
    }
    System.out.println("here they come");
    for(int j = 0; j < firstArray.length; j++)
        System.out.println(firstArray[j]);
}
}
Arrays - the mental picture..

int[] firstArray = new int[10];

firstArray.length -> 10
firstArray[6] = 17;
Shorthand

```java
int[] nums = {2,4,6,8,10};

makes an array of 5 ints:
System.out.println(nums[4]); -> prints 10
```

If myKid, yourKid, jillsKid, leahsKid, nedsKid already exist as Infant objects, then this is ok:

```java
Infant[] someKids = 
   {myKid, yourKid, jillsKid, leahsKid, nedsKid};
```
Arrays are objects.

When you say “length” you are invoking a constant (public final value) associated with the array.

The size of an array is determined when “new” is invoked:

```java
int[] someArray = new int[66];
int[] nums;  // this is ok - variable is named
```

Array indices are always int, and always start at 0.

Array indices end at cell # (length - 1): same as String indexing.
An application

We’re going to write an application that rolls a pair of dice some number of times and reports the results as a profile of the rolls (e.g. how many 2, 3, 4, .. etc. came up).
Results: (10,000 tosses)

toss of 2 303
toss of 3 543
toss of 4 807
toss of 5 1123
toss of 6 1432
toss of 7 1630
toss of 8 1389
toss of 9 1129
toss of 10 808
toss of 11 557
toss of 12 279
Key idea:

Indices of a “scoreboard” array actually stand for dice toss outcomes.
import javax.swing.JOptionPane;

public class DiceExperiment {
    public static void main(String[] args) {
        String tossString = JOptionPane.showInputDialog("enter toss count");
        int tossCt = Integer.parseInt(tossString);
        Dice d = new Dice();
        d.multiToss(tossCt);
        d.showScoreboard();
    }
}

public class Dice{
    private int[] scoreboard = new int[13];

    public Dice(){ initializeScoreboard(); } }

public void initializeScoreboard(){
    for(int j = 0; j < 13; j++) scoreboard[j] = 0;
}

public int tossDie(){
    return (1+ (int)(6*Math.random()));
}

public int throwDice(){
    return(tossDie() + tossDie());
}
public void multiToss(int tossCount){
    int score;
    for (int j = 0; j < tossCount; j++){
        score = throwDice();
        scoreboard[score]++;
    }
}

public int[] getScoreboard(){return scoreboard; }

public void showScoreboard(){
    for(int j = 2; j < 13; j++)
        System.out.println("toss of "+j+" "+scoreboard[j]);
}
} // ends class
The Scoreboard

... 23 ...  

5 6 7 8 9 ...

score = throwDice();
scoreboard[score]++;

Suppose throwDice() returns 6 -> then what?
Arrays of objects

Infant[] kids = new Infant[10]; // array of 10 infants

Infant littleMikey = new Infant(“Mike”, 3);
kids[4] = littleMikey;
Places littleMikey into cell with index 4 of the kids array.

Kid at cell 2 has wrong name; should be Lilly
kids[2].setName(“Lilly”);

public void allOlder(Infant[] kiddo){
    for(int j = 0; j < kiddo.length; j++)
        kiddo[j].anotherMonth();
}
A typical array problem:

Find the name of the oldest kid in an array of Infants

Assume zeroth kid is the oldest - set aside her position (0), and her age

Walk down the array (may as well start with 1)
When you find someone older:
set aside her position (j), her age

When you’re done, j holds the index of the oldest kid (who could be older???)
Get that kid, return her name!
public String oldest(Infant[] kiddo) {
    if (kiddo.length == 0) return "no kids";
    int oldestSoFar = 0; // an array index
    int oldAge = kiddo[0].getAge();
    int curAge;
    for (int j = 1; j < kiddo.length; j++) {
        curAge = kiddo[j].getAge();
        if (curAge > oldAge) {
            oldAge = curAge;
            oldestSoFar = j; // location of oldest kid so far
        }
    }
    return (kiddo[oldestSoFar].getName());
}
public boolean anyBabies(Infant[] kiddo) {
    // are any kids less than 2 months old?
    boolean aBaby = false;
    for (int j = 0; j < kiddo.length; j++) {
        if (kiddo[j].getAge() < 2) {
            aBaby = true;
            break;
        }
    }
    return aBaby;
}
public boolean majorityOld(Infant[] kiddo, int a) {
// are strict majority in array older than age a?
    int old = 0;
    for (int j = 0; j < kiddo.length; j++) {
        if (kiddo[j].getAge() > a) {
            old++;
        } else old--;
    }
    return (old > 0);
}
Java’s “for-each” construction

Most for loops that apply to arrays march down an entire array of objects, either:

• Collecting information; or

• Altering the contents of objects
public class ArrayTest2{
    public static void main(String args[]) {
        Infant kid1 = new Infant("a",12);
        Infant kid2 = new Infant("b",12);
        Infant kid3 = new Infant("c",12);
        Infant[] kids = {kid1,kid2,kid3};
    }
}
```java
for (Infant kid : kids) {
    System.out.print(kid.getAge() + " ");
    kid.anotherMonth();
}
System.out.println();

for (Infant kid : kids) {
    System.out.print(kid.getAge() + " ");
}
```

Prints:

```
12 12 12
13 13 13
```
for (Infant kid : kids)

- Type tag
- variable colon
- array
A caveat

You can’t change the array (directly)

```java
int[] nums = {5,5,5,5,5};

for(int i : nums) System.out.print(i);
55555

for(int i : nums) i++;
// increase nums(?)

> for(int i : nums) System.out.print(i);
55555>
```
public boolean anyBabies(Infant[] kiddo)
{
  // are any kids less than 2 months old?
  boolean aBaby = false;
  for(Infant k : kiddo)
  {
    if (k.getAge() < 2)
    {
      aBaby = true;
      break;
    }
  }
  return aBaby;
}
public boolean majorityOld(Infant[] kiddo, int a) {
    // are strict majority in array older than age a?
    int old = 0;
    for (Infant k : kiddo) {
        if (k.getAge() > a) {
            old++;
        } else {
            old--;
        }
    }
    return (old > 0);
}
public String oldest(Infant[] kiddo){
    if (kiddo.length == 0) return "no kids";
    Infant oldKid = kiddo[0];
    for(Infant k : kiddo)
        if (k.getAge() > oldKid.getAge()) oldKid = k;
    return(oldKid.getName());
}
How would you write a (static) method to compute the average age of a bunch of kids in an Infant array?
public static double avgKid(Infant[] kids){
    int count = 0; int total = 0;
    for(Infant i : kids){
        total = total + i.getAge();
        count++;
    }
    if (count == 0) return 0.0;
    else return (double)total/count;
}

public static void main(String[] args) {

What does it mean???
What are the relative frequencies of letters in English?

We’ll work on a simplified version: what are the relative frequencies of letters in English, where we are working only with a typed text (you type in lines of text, end with two carriage returns)
General plan:
A Driver class (LetterFreq)
A principal class (Letters)
Think informally first:
• Keep a scoreboard
• Read lines
• For each line, march through its characters, and when you find a letter, increment the appropriate scoreboard cell.
Driver outline

Make a Letters object
While last line wasn’t blank
    read a new line
    process the line - inside Letters obj
Make your frequency report - from Letters obj
import java.util.Scanner;

global class LetterFreq{
  public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);
    String t = " ";
    System.out.println("EntrTxt-blankTo end");
    Letters letterStudy = new Letters();
    while(t.length() > 0) {
      t = scan.nextLine();
      letterStudy.process(t);
    }
    letterStudy.reportLetters();
  }
}
LetterFreq

Sends String for processing

Letters

returns, waiting for next String
public class Letters{

    private int[] scoreboard = new int[26];
    private int count = 0; // letter count

    public Letters(){
        for(int j = 0; j < 26; j++) scoreboard[j] = 0;
    }

    public void process(String s){  }

    public void reportLetters(){ }
}

public void process(String str){
    char ch; int pos;
    String s = str.toLowerCase();
    for(int j = 0; j < s.length(); j++){
        ch = s.charAt(j);
        if (Character.isLetter(ch)){
            count++;
            pos = ch-'a';
            scoreboard[pos]++;
        }
    }
}
}
public void reportLetters()
{
    int which = (int)'a';
    for(int i : scoreboard){
        System.out.println((char)(which)+ "  " + i);
        which++;
    }
}