Announcements
• Today is drop-dead day for dropping classes
• Diffy, Ch7 embedded problems due Monday
• Ch OWL hwk due Thursday
• Program 5 up soon
• Final 12/18
• Another very short survey about to be posted
• Midterms back? Come to my office. Solutions posted soon
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 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

```java
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
```

Indexing system like char/string
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();
```
$k = 7;$

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..

```java
int[] firstArray = new int[10];

firstArray.length -> 10
firstArray[6] = 17;
```
These sorts of expressions are possible:

```java
firstArray[4] = 9*firstArray[4];
firstArray[3] = 11;

int j = firstArray[3]/2;
firstArray[j] = 9*firstArray[j/2];
```
Shorthand

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

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

If `myKid`, `yourKid`, `jillsKid`, `leahsKid`, `nedskid` already exist as Infant objects, this is ok:
```java
Infant[] someKids =
    {myKid, yourKid, jillsKid, leahsKid, nedsKid};
```
First 10 Fibonacci numbers:
1, 1, 2, 3, 5, 8, 13, 21, 34, 55

Fibonacci #s: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

```java
int[] fibos = new int[10];
fibos[0] = 1;
fibos[1] = 1;
for(int j = 2; j < fibos.length; j++)
    fibos[j] = fibos[j-1] + fibos[j-2];
```
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 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(tossCt);
        d.multiToss();
        d.showScoreboard();
    }
}
public class Dice{
    private int[] scoreboard = new int[13];
    private int tossCt;

    public Dice(int tosses)
    { tossCt = tosses;
        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 score;
    for (int j = 0; j < tossCount; j++){
        score = throwDice();
        scoreboard[score]++;
    }
}

// Note: cells 0, 1, never get used
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

Note: we ignore cells 0 and 1
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 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){
    // what's name of oldest kid?
    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; // loc 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;
}
p. boolean majorityOld(Infant[] kiddo, int a){
    // are strict majority in array older than a?
    int old = 0;
    for(int j = 0; j < kiddo.length; j++){
        if (kiddo[j].getAge() > a)
            old++;
        else old--;
    }
    return (old > 0);
}