Late rules in effect - late programs are reduced in value by half.
Tutorial grades - will be given manually.
Program 5 due Monday
Next OWL assignment due friday

Drop date: 5 PM monday
We’ve seen Java at two levels:

• the statement level - mechanisms for getting specific, often low-level jobs, done, e.g. assignment stmts, 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.

Basically, 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 the deli line at a super-market.

In all cases:

- Many variables required for representation
- 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
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;
```
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: \( x[0] = 2* x[1]; \)
int[] nums = {2,4,6,8,10} makes an array of 5 ints:
System.out.println(nums[4]);  -> prints 10

First 10 Fibonacci numbers:
1,1,2,3,5, 8, 13, 21, 34, 55

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];
Let’s work on a simple text processing application:

Read in a line of text from the keyboard, then report on the frequencies of the letters in the text.

Important: arrays are always numerically indexed, beginning at 0

We’ll start with some facts about letters…
public class Test{
    // all characters have a position in Unicode
    // Ascii comes first : 0 - 127
    // when an integer op is applied to a character, it
    // is automatically converted into an integer
    public static void main(String[] args){
        int i = (int) 'a';  int j = (int) 'A';
        System.out.println(i + " " + j); // positions of a,A
        System.out.println('a' - 'A');
        System.out.println('e' - 'a');
    }
}

----output
97 65 // positions of a,A (a comes first)
32
4
import java.util.Scanner;

public class LetterFreq {
    public static void main (String[] args) {
        final int LETTERCOUNT = 26;

        int[] letterChart = new int[LETTERCOUNT];
        for(int j = 0; j < LETTERCOUNT; j++)
            letterChart[j] = 0;  // initialize the chart

        char ch;
        Scanner s = new Scanner(System.in);
        System.out.println("Enter a line of text");
        String str = s.nextLine();  // read a phrase
str = str.toLowerCase();
    for (int pos = 0; pos < str.length(); pos++)
    {
        ch = str.charAt(pos); // the key loop
        if (ch >= 'a' && ch <= 'z')
            letterChart[ch-'a']++;
    }

    // Print results
    System.out.println("Letter scoreboard for phrase");
    for (int pos=0; c < letterChart.length; c++)
    {
        System.out.print ( (char) (pos + 'a') );
        System.out.println("--> " + letterChart[pos]);
    }
}
Enter a line of text
   ok, I will enter a line of text, and then we'll see if it's a line of the very very very highest quality!

   Letter scoreboard for phrase
   a--> 4
   b--> 0
   c--> 0
   d--> 1
   e--> 13
   ..........
   t--> 8
   u--> 1
   v--> 2
   w--> 2
   x--> 1
   y--> 3
   z--> 0
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, etc came up).

```java
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]);
}
Results: (10,000 tosses)

<table>
<thead>
<tr>
<th>Toss of 2</th>
<th>303</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toss of 3</td>
<td>543</td>
</tr>
<tr>
<td>Toss of 4</td>
<td>807</td>
</tr>
<tr>
<td>Toss of 5</td>
<td>1123</td>
</tr>
<tr>
<td>Toss of 6</td>
<td>1432</td>
</tr>
<tr>
<td>Toss of 7</td>
<td>1630</td>
</tr>
<tr>
<td>Toss of 8</td>
<td>1389</td>
</tr>
<tr>
<td>Toss of 9</td>
<td>1129</td>
</tr>
<tr>
<td>Toss of 10</td>
<td>808</td>
</tr>
<tr>
<td>Toss of 11</td>
<td>557</td>
</tr>
<tr>
<td>Toss of 12</td>
<td>279</td>
</tr>
</tbody>
</table>
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 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();
Find the name of the oldest kid in an array of Infants

Assume zeroth kid is the oldest - set aside her position (0), 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){
    int oldestSoFar = 0;  
    int oldAge = kiddo[0].getAge();
    int myAge;
    
    for(int j = 1; j < kiddo.length; j++){
        myAge = kiddo[j].getAge();
        if (myAge > oldAge){
            oldAge = myAge;
            oldestSoFar = j;
        }
    }
    
    return(kiddo[oldestSoFar].getName());
}