CS 121 – Intro to Programming: Java - Lecture 4
Course home page:
http://twiki-edlab.cs.umass.edu/bin/view/Moll121/WebHome

OWL Homework due tomorrow
Programming project due Friday
Captive labs? Software problems?

Topics:
General application structure
Primitives
String class
Views of classes
   APIs
   Driver code as specification
Scanner class
Old MacDonald does Java

Old Macdonald had a farm // first two lines make chorus
ei, ei, o;
and on that farm he had a pig
ei ei O
With an oink oink here
And a oink oink there
Here a oink there a oink
Everywhere a oink oink
Old Macdonald had a farm
ei, ei, o;

Old Macdonald had a farm
ei, ei, o;
and on that farm he had a dog
ei ei O (etc...)
Overall structure

Chorus

Pig verse

Chorus

Chorus

Dog verse

Chorus

Who are the players?

Chorus object

Verse object - it’s parameterized (pig, dog, etc)

A coordinating “song” class
public class MacSong{
    public static void main(String[] args){
        MacChorus m = new MacChorus();
        MacVerse p = new MacVerse("pig", "oink");
        MacVerse d = new MacVerse("dog", "woof");
        m.chorus();
        p.verse();
        m.chorus();
        m.chorus();
        m.chorus();
        d.verse();
        m.chorus();
    }
}

Three objects made - a chorus object, and two verse objects. The verse objects differ in that their attributes hold different values. The verse method exploits this to give different verses.
public class MacChorus{

public void chorus()
{
    // a method that serves MacChorus objects
    System.out.println("Old Macdonald had a farm");
    System.out.println("ei, ei, o;");
    }
}
}
public class MacVerse{

String name;    String noise;    // attributes

public MacVerse(String animalName, String animalNoise){
    name = animalName;
    noise = animalNoise;   }

public String getName(){return name;}
public String getNoise(){return noise;};

public void verse(){
System.out.println("and on that farm he had a " + name);
System.out.println("ei ei O");
System.out.println("With a " + noise + " " + noise + " here");
System.out.println("And a " + noise + " " + noise + "there");
System.out.println("Here a " + noise + " there a " + noise);
System.out.println("Everywhere a " + noise + " " + noise);  }
}
}
Primitive Data Types

• objects are Java’s main currency

• Too tedious for them to be the only currency - making numbers a kind of object is a pain, however.

• Primitive data types integers (4), floats(2), char, boolean. That these aren’t actually objects will turn out to be a pain, too.

Statement like these are fairly common:

    int count = 0;
    boolean chewsTobacco = false;
    char averageGrade = ‘C’; // note the single quotes
Strings - A very important class.

String greeting;
greeting = new String(“ola”);
greeting2 = new String(“howdy”);
greeting = greeting2;
System.out.println(greeting);

Some caveats:
1) Strings are not primitives (unlike float, double, int, etc)
2) There’s a shorthand for String creation:
greeting = “ola”; // works fine
3) As a class, String comes with extensive functionality
String pupName = “spot”;
int len = pupName.length(); // len assigned 4
char what = pupName.charAt(1); // what assigned ‘p’
char what = pupName.charAt(0); // what assigned ‘s’
String huh = pupName.concat(“less”); // spotless
String bigHuh = pupName.toUpperCase(); // SPOT

Where do I find out about the String class… (hold on)
Multiple views of a class. Most obvious - the class source code:

```java
public class Infant{

    private String name;
    private int age;  // in months

    public Infant(String who, int months){
        name = who;
        age = months;
    }

    public String getName(){return name;}

    public int getAge(){return age;}

    public void anotherMonth(){age = age + 1;}
}
```
Underlying principle - Encapsulation:
Hide information (there’s too much of it - only keep track of what you really need to know)

And all you really need to know, in many cases, is how use a class - not the details of how a class is implemented (how it does what it does).
For example, we can get by just fine building and manipulating Infant objects if we just use the Infant class API, and don’t bother explicitly with the class at all.

This IS big
The entire Java API is online, at

http://java.sun.com/j2se/1.5.0/docs/api/

This is the link we use in the online book, but you can look at it any time, for example to look up the String class’s API, etc.
Java™ 2 Platform Standard Edition 5.0
API Specification

This document is the API specification for the Java 2 Platform Standard Edition 5.0.

See:
Description

### Java 2 Platform Packages

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.applet</td>
<td>Provides the classes necessary to create an applet and the classes to communicate with its applet context.</td>
</tr>
<tr>
<td>java.awt</td>
<td>Contains all of the classes for creating user interfaces and for presentation and behavior.</td>
</tr>
<tr>
<td>java.awt.color</td>
<td>Provides classes for color spaces.</td>
</tr>
<tr>
<td>java.awt.datatransfer</td>
<td>Provides interfaces and classes for transferring data between applications.</td>
</tr>
<tr>
<td>java.awt.dnd</td>
<td>Drag and Drop is a direct manipulation gesture found in many GUI systems that provides a mechanism to transfer information between applications associated with presentation elements in the GUI.</td>
</tr>
<tr>
<td>java.awt.event</td>
<td>Provides interfaces and classes for dealing with different types of events.</td>
</tr>
<tr>
<td>java.awt.font</td>
<td>Provides classes and interface relating to fonts.</td>
</tr>
<tr>
<td>java.awt.geom</td>
<td>Provides the Java 2D classes for defining and performing operations in two-dimensional geometry.</td>
</tr>
<tr>
<td>java.awt.im</td>
<td>Provides classes and interfaces for the input method framework.</td>
</tr>
<tr>
<td>java.awt.im.spi</td>
<td>Provides interfaces that enable the development of input method Java runtime environment.</td>
</tr>
</tbody>
</table>
A Third View (sort of) - you may write code that tells how you want a class to work before you actually create the class, then this is a sort of specification for the class.

Consider the following BathTub driver that some enterprising plumber might want..
public class TubTester{
public static void main(String[] args){
    System.out.println("his & hers tubs in house: ");
    BathTub b= new BathTub("king", 60);
    BathTub t = new BathTub("queen", 70);
    System.out.println("Total water capacity: " +
    (b.getCapacity() + t.getCapacity() ));
    System.out.println("total water weight of larger: "
    + t.totalWt());
}
}
BathTub class has:

A constructor that takes a “name” and a (water) capacity

Has get methods for these

Has a totalWt method also - reports weight of water in full tub
public class BathTub{
    private String name;
    private int capacity; // in gallons
    final double GALLON_WT = 8.345; //gal of wtr-lbs

    public BathTub(String n,int cap){
        name = n;
        capacity = cap;
    }

    public String getName(){return name;}
    public int getCapacity(){return capacity;}

    public double totalWt(){
        return (GALLON_WT * capacity);
    }
}

import java.util.Scanner;

class Adder {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter 2 decimal numbers");
        double num1 = scan.nextDouble();
        double num2 = scan.nextDouble();
        System.out.println("Sum of " + num1 + num2 + " is ");
        System.out.println(num1 + num2);
    }
}