Third OWL assignment up, due next week
Second programming assignment due next week. See website for details.

Please do Survey!

Honors section: first meeting today at 4, in LGRC A311.

This week we’ll talk about Ch 3. Read Ch 4 for next Tuesday

Due dates: Program 1- end of day today; Program 2 - next Wed
OWL homework, Ch 3 homework: beginning of class next Tuesday

Office hours: TA Office hours: (in LGRT 213): M 12:30 - 2:30; Tu 1-5; W 3-5; Th 10-11, 2:45-3:45; F 12:30 - 4:30
A typical Java program
A very, very simple application

Class G

main
public class Howto{
    // a baby intro example
    public static void main(String args[]) {
        System.out.println("Welcome to 121");
        System.out.println("3 + 5");
        System.out.println(3 + 5);
    }
}

Prints:
Welcome to 121
3 + 5
8
Our current view of the OO programming landscape..
The Object Model

We model “things” as objects

Objects have attributes, and behaviors

The “state” of an object - the snapshot values of its attributes

“Things” in the world change; as a reflection, objects change state
Where are an object’s characteristics written down?

Answer: In a class declaration or definition

A class definition for an object is different from an object, in the same way that the blueprint for a house is different from a house.

Still -

How do you make an object?

How do you invoke its behaviors?
Infant objects

Attributes

name

age (in months)

Behaviors

getName

getAge

anotherMonth (make kid one month older)
Broadly, two kinds of behaviors:

**Tell or get behaviors** - how old’s the kid, what’s the kid’s name, what’s the kid’s weight in ounces, what’s the kid’s weight in kilos. These do queries, calculations, reports.

Tell behaviors do not change the calling object

**Mutating behaviors** - These alter the calling object - a name is changed, an age increases, a car’s gas tank is filled, etc. When a mutator does what it does, the state of the calling object changes.
public class InfantTester{

    public static void main (String[] args){
        Infant myKid = new Infant("Kit",4);
        int kitAge = myKid.getAge();
        System.out.println("name: " + myKid.getName());
        myKid.anotherMonth();
        System.out.println("age: " + kitAge);
    }
}

public class Infant{

    private String name; // name, age are Infant attributes
    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;}
}
Assignment Statements and Identifiers

An identifier is the name of a variable (or method, or class..)

```java
int num = 7; // num now “holds” 7
num = 4;   // num now “holds” 4
num = num + 2; // num now “holds” 6
```

Assignment is NOT equality!

Assignment is an **action** operator: **Compute** the RHS,
Then **copy** the result to the variable named on the LHS
Primitive Data Types
• objects are Java’s main currency
• Too tedious for them to be the only currency
• Primitive data types (8): integers (4), floats(2), char, boolean.

Statement like these are fairly common:

long count = 0;     // like int, but larger range

double bigPapiAvg = 0.324;  // numbers with decimal pts

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); // prints howdy

Some caveats:
1) Strings are not primitives (unlike double, int, boolean)
   String is a class in the package java.lang

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 is assigned ‘p’

char ch = pupName.charAt(0); // ch is 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. One: 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 Java 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.
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.

(There’s an example in the text where we do this with the TreeHouse class)

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><code>java.applet</code></td>
<td>Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.</td>
</tr>
<tr>
<td><code>java.awt</code></td>
<td>Contains all of the classes for creating user interfaces and for painting graphics and images.</td>
</tr>
<tr>
<td><code>java.awt.color</code></td>
<td>Provides classes for color spaces.</td>
</tr>
<tr>
<td><code>java.awt.datatransfer</code></td>
<td>Provides interfaces and classes for transferring data between and within applications.</td>
</tr>
<tr>
<td><code>java.awt.dnd</code></td>
<td>Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.</td>
</tr>
<tr>
<td><code>java.awt.event</code></td>
<td>Provides interfaces and classes for dealing with different types of events fired by AWT components.</td>
</tr>
<tr>
<td><code>java.awt.font</code></td>
<td>Provides classes and interface relating to fonts.</td>
</tr>
<tr>
<td><code>java.awt.geom</code></td>
<td>Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.</td>
</tr>
<tr>
<td><code>java.awt.im</code></td>
<td>Provides classes and interfaces for the input method framework.</td>
</tr>
<tr>
<td><code>java.awt.im.spi</code></td>
<td>Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.</td>
</tr>
<tr>
<td><code>java.awt.image</code></td>
<td>Provides classes for creating and modifying images.</td>
</tr>
<tr>
<td><code>java.awt.image.renderable</code></td>
<td>Provides classes and interfaces for producing rendering-independent images.</td>
</tr>
<tr>
<td><code>java.awt.print</code></td>
<td>Provides classes and interfaces for a general printing API.</td>
</tr>
<tr>
<td><code>java.beans</code></td>
<td>Contains classes related to developing beans - components based on the JavaBeans™ architecture.</td>
</tr>
</tbody>
</table>
public class LooseLeaf{
    // models a looseleaf notebook, with count of blank sheets
    private int blankPages;
    private String name;  // notebook owner

    public LooseLeaf(String who, int blanks) {
        blankPages = blanks;
        name = who;
    }

    public int getBlankPages() {
        return blankPages;
    }

    public String getName() {
        return name;
    }

    public void setBlankPages(int amount) {
        blankPages = amount;
    }
}
public class LooseLeafTester{

    public static void main(String[] args){

        LooseLeaf jacksBook = new LooseLeaf("Jack", 50);
    }

}
public class LooseLeafTester{
    public static void main(String[] args){
        LooseLeaf jacksBook = new LooseLeaf("Jack", 50);
        LooseLeaf jillsBook = new LooseLeaf("Jill", 20);
    }
}
public class LooseLeafTester{
    public static void main(String[] args){
        LooseLeaf jacksBook = new LooseLeaf("Jack", 50);
        LooseLeaf jillsBook = new LooseLeaf("Jill", 20);
        int jacksBlanks = jacksBook.getBlankPages();
    }
}

public class LooseLeafTester{
    public static void main(String[] args){
        LooseLeaf jacksBook = new LooseLeaf("Jack",50);
        LooseLeaf jillsBook = new LooseLeaf("Jill",20);
        int jacksBlanks = jacksBook.getBlankPages();
        jacksBook.setBlankPages(jacksBlanks - 20);
        int jillsBlanks = jillsBook.getBlankPages();
    }
}
public class LooseLeafTester{
    public static void main(String[] args){
        LooseLeaf jacksBook = new LooseLeaf("Jack", 50);
        LooseLeaf jillsBook = new LooseLeaf("Jill", 20);
        int jacksBlanks = jacksBook.getBlankPages();
        jacksBook.setBlankPages(jacksBlanks - 20);
        int jillsBlanks = jillsBook.getBlankPages();
        jillsBook.setBlankPages(jillsBlanks + 20);
    }
}
public class LooseLeafTester{
    public static void main(String[] args){
        LooseLeaf jacksBook = new LooseLeaf("Jack",50);
        LooseLeaf jillsBook = new LooseLeaf("Jill",20);
        int jacksBlanks = jacksBook.getBlankPages();
        jacksBook.setBlankPages(jacksBlanks - 20);
        int jillsBlanks = jillsBook.getBlankPages();
        jillsBook.setBlankPages(jillsBlanks + 20);
        System.out.println(jillsBook.getBlankPages());
    }
}
public class Car{

    private String make; // manufacturer
    private double fuelCapacity;
    private double fuelAmount;

    // the Car constructor
    public Car(String what, double cap, double amt){
        make = what;
        fuelCapacity = cap;
        fuelAmount = amt;
    }
}
// the Car methods

public String getMake(){
    return make;
}

public double getCapacity(){
    return fuelCapacity;
}

public double getFuel(){
    return fuelAmount;
}

public void setFuel(double amt){
    fuelAmount = amt;
}

public double unusedCap(){
    return (fuelCapacity - fuelAmount);
}
public class CarTester{
    public static void main(String[] args){
        Car mine = new Car("Ford",15,9.5);
        double amt = mine.unusedCap();
        System.out.println("fill-up cost " + 2.65*amt);
    }
}
The Scanner class is in the package java.util.

You need to load it in - it’s not automatically available, as the classes in java.lang are.

The import statement loads classes (makes them available to your program):

import java.util.Scanner;

Import statements go outside and before your class definitions.
import java.util.Scanner; // makes Scanner available

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);
    }
}
import java.util.*;

public class Paste{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.println("enter first string");
        String s = scan.next();
        System.out.println("enter second string");
        String t = scan.next();
        System.out.println("paste them together");
        System.out.println(s + t);
    }
}
}
import java.util.*;

public class Infant2Tester{

    public static void main (String[] args){
        Scanner s = new Scanner(System.in);
        System.out.println("Enter kid name");
        String name = s.next(); // gets a String
        System.out.println("Enter kid age");
        int howOld = s.nextInt(); // gets a String
        System.out.println("Enter kid age");
        int howOld = s.nextInt();
        Infant who = new Infant(name,howOld);
        System.out.println(who.getName()+ " is "+ who.getAge());
    }
}

> java Infant2Tester
Enter kid name
Enter kid age
Dana is 19
Infant

Scanner

Infant2Tester
import java.util.*;

public class VerseMaker{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.println("enter animal name");
        String name = scan.next();
        System.out.println("enter animal noise");
        String noise = scan.next();
        MacVerse m = new MacVerse(name,noise);
        m.verse();
    }
}

java VerseMaker

enter animal name (donkey entered)
enter animal noise (hee-haw entered)

and on that farm he had a donkey
ei ei o
With an hee-haw hee-haw here
And a hee-haw hee-haw there
Here a hee-haw there  a hee-haw
Everywhere a hee-haw hee-haw