CS 121 – Intro to Programming: Java - Lecture 4

Professor Robert Moll (+ TAs)
CS BLDG 276 - 545-4315
moll@cs.umass.edu

Course home page:
http://twiki-edlab.cs.umass.edu/bin/view/Moll121/WebHome

Announcements
Second OWL assignment up, due Tuesday;
Second Programming assignment up, due next Friday (9/19)
No office hours next Monday; I’ll be a little late for mine today
Worked examples in text (esp at end of Ch 2); Reading stats;
TA Office hours: M 12-4; TU 1-3:40; W 4-6; TH 1-4; F 12-4, in (back room next to) LGRT 223
```java
public class InfantTester{

    public static void main (String[] args){
        Infant myKid = new Infant("Lizzie",4);
        int lizAge = myKid.getAge();
        System.out.println("my kid's name is " +
                myKid.getName());
        myKid.anotherMonth();
        System.out.println("my kid is now " +
                myKid.getAge() + " months");
    }
}
```

Variables are “live” inside main: myKid,lizAge
* Methods that return something create/adjust contents directly (L 4,5,6,8);
* void methods don’t return anything, may write to screen,or adjust internals of objects(L6,7,8)
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;}
}

Name: Lizzie
Age: 4

myKid.anotherMonth();

Name: Lizzie
Age: 5

myKid [1]                                                           myKid [3]

myKid [1]

4

[2]  lizAge = myKid.getAge();
Objects associated with 2 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, etc.; involve `return` stmt

Get behaviors do NOT change the calling object

**Mutating behaviors** - These alter the calling object - a name is changed, an age is increased, a car’s gas tank is filled, etc. When a mutator does what it does, the `state` of the calling object changes.
The cell model and assignment

```java
int j = 5;
int k = 10;
int m = 2;

j = j + k;
k = 2 * k + j;
m = m + k;
```
The egg carton problem

public class EggCarton{
    private int eggs = 12;
    private String owner;

    public EggCarton(String name){
        owner = name;
    }

    public int getEggs(){return eggs;}
    public String getOwner(){return owner;}

    public void haveAnEgg(){eggs = eggs - 1;}  // no test for zero eggs!
}
Assignment Statements and Identifiers

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

```plaintext
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; // long is 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; String greeting2;
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 standard class

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.
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 an applet uses 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 painting graphics and images.</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 and within applications.</td>
</tr>
<tr>
<td>java.awt.dnd</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>java.awt.event</td>
<td>Provides interfaces and classes for dealing with different types of events fired by AWT components.</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 on objects related to 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 methods that can be used with any Java runtime environment.</td>
</tr>
<tr>
<td>java.awt.image</td>
<td>Provides classes for creating and modifying images.</td>
</tr>
<tr>
<td>java.awt.image.renderable</td>
<td>Provides classes and interfaces for producing rendering-independent images.</td>
</tr>
<tr>
<td>java.awt.print</td>
<td>Provides classes and interfaces for a general printing API.</td>
</tr>
<tr>
<td>java.beans</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);
    }
}