CS 121 - Intro to Programming:Java - Lecture 4

Announcements

Introductory Survey - Please do it!

Midterm 10/14, 7 PM (see syllabus)

First programming assignment on OWL, due Thur at 5

Ch 2 OWL hwk due Friday at 5

Ch3 Embedded problems due Monday, 10 AM

Office hours posted, me + TAs

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

Check CourseWork link at left for due dates

ClassDiscussions09 - repository of discussion scripts
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 (L 6, 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;}
}

myKid.anotherMonth();

lizAge = myKid.getAge();
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;
```
An egg carton class

```java
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;}

    public void haveSomeEggs(int eatCt){ // note param
        eggs = eggs - eatCt;
    }
}
```
Assignment Statements and Identifiers
An identifier is the name of a variable (or method, or class..)

```cpp
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 = 0.224;  // nums 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 (Java library) class

2) There's a shorthand for String creation:
   greeting = "ola"; // works fine

3) String class comes with extensive functionality
String pupName = "spot";

int len = pupName.length(); // len assigned 4

char ch = pupName.charAt(1); // ch 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:

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 about)

Often all you need to know:

how use a class - not the details of how a class is implemented.
For example, just use the

**Infant class API,**

don’t bother explicitly with 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.

How about the API for the String class?
**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 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, counts 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);
    }
}
A **package**: a bundle of classes with a common general purpose

There are dozens of standard packages in Java

You can make your own packages

When you run standard Java, only the package `java.lang` is automatically loaded

Classes in other packages are loaded as needed

Load by need feature makes Java the recycling engine we touted at beginning of term
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):

```java
import java.util.Scanner;
```

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

public 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.Scanner;

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