CS 121 - Intro to Programming: Java - Lecture 5

Announcements

Ch 3 embedded problems due Wed noon. Ignore trapdoor.
Programming Assignment 2 due Friday at 4 PM
OWL Ch 3 up, due Monday, 5 PM
So far we’ve touched on a number of low level issues in Java, having to do with assignment, types, etc:

String s = “top”;

int n = 12;

int j = 5;

    n = n + n;

    s = s + s;

    s = s + j;

    j = n + s;
The Object model

Objects underly Java programs
Their creation, evolution model real world situations
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());
    }
}

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 objects - it’s parameterized (pig, dog, etc)
A coordinating “song” class - the driver
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(); // where does its information come from?
    d.verse(); // where does its information come from?
    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{
private String name;
private String noise;

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); }
}
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);
    }
}

Scanner console = new Scanner(System.in);

Scanner object means: input from keyboard

---------------------

console.nextInt() --- looking for an int value

console.nextDouble() --- looking for a double value

console.next() --- looking for a String value

console.nextLine() --- looking for a whole line
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 RevLines{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.println("enter first phrase");
        String s = scan.nextLine();
        System.out.println("enter second phrase");
        String t = scan.nextLine();
        System.out.println(t);
        System.out.println(s);
    }
}

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();
    }
}

MacVerse m = new MacVerse(name,noise);
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
import java.util.Scanner;

public class MacVerseTester{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter an animal name");
        String animal = scan.next();
        System.out.println("Enter that animal's noise");
        String noise = scan.next();
        MacVerse someAnimal =
            new MacVerse(animal,noise);
        someAnimal.verse();
    }
}

Data tagging/flow of control(2) / object construction
/methods/packages/spelling conventions
Java and the notion of Encapsulation

• Encapsulation = Information Hiding
• Serves the user: think about nextDouble()
• Serves the implementor: just meet the API
• Underlying principle for code recycling
• Underlying principle for code hygiene
Working backwards

[a box of books has a count of books, and a weight of books]

```java
public class BookWork {
    public static void main() {
        BoxOfBooks b = new BoxOfBooks(43, 70.3);
        System.out.println("There are " + b.getCount() + " books.");
        System.out.println("They weigh " + b.getWeight() + " pounds.");
    }
}
```
public class BoxOfBooks{

}
public class BoxOfBooks{

    private int count;
    private double weight;

}
public class BoxOfBooks{

    private int count;
    private double weight;

    public BoxOfBooks(int c, double w){
        count = c;
        weight = w;
    }
}
}
public class BoxOfBooks{

    private int count;
    private double weight;

    public BoxOfBooks(int c, double w){
        count = c;
        weight = w;
    }

    public int getCount(){return count;}
}