CS 121 - Intro to Programming:Java - Lecture 5

Announcements

Ch 4 Embedded problems due Friday, 10 AM
Programming Assignment 2 due Friday at 5 PM
Third OWL assignment up, due Monday, 10 AM
Fourth OWL assignment due (next) Thursday, 10 AM
Midterm posted (W evening 10/14)

TA Office hours: see website, CourseAdministration. Note: there are office hours today, 3:45 - 5:00.

Email discussion topics!
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
MacChorus

MacVerse

main

MacSong
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); }
}
```java
import java.util.Scanner;

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
Working backwards

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{

    private int count;
    private double weight;

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

    public int getCount(){return count;}
}

Classes, Objects, and Time

When a class definition is written, we don’t know exactly which objects will be created according the definition - that is, we almost always have a blueprint with unspecified parts. Example: the MacVerse blueprint does not identify an animal, nor does it identify a noise. We use instance variables (attribute names) as place holders for these to-be-specified features.

“When she got home, Jill hung up her coat.”
Arithmetic and Operator Precedence

1) $+, -, \ast$ behave in standard way. Division / different

2) $5/3 = 1$ (but $5.0/3 = 5/3.0 = 1.6666$), $10/4 = ?$

3) In the absence of parentheses, $\ast, /$, have higher precedence than $+,-$
So: $(3 + 5 \ast 2) = 13$, $(7 - 4 / 2) = ?$

4) mod (remainder) operator $\%$: $10 \% 3 = 1$, $10 \% 7 = ?$

$(3 + (7/2)) = ?$
$(2 \ast 3 - 1 + 5 / 3) = ?$
$(20 \% (9 \% 4)) = ?$
$(5 \% 0) = ?$
What is 365 % 10

What is 365 / 10

What is (365 / 10) % 10

What is (365 / 100)
Conditional, Looping Statements in Java

Conditional statements and looping statements are flow of control constructions

At a primitive level, Java programs are made up of statements, and it often makes sense to

1) have statements repeat in a systematic way; and
2) have statements execute conditionally
Conditionals first - Consider:

\[
\text{if (n % 2 == 0) System.out.println("n is even");}
\]

Lots going on here: statement says: “if the remainder after dividing n by 2 is equal to (==) 0, then report that n is an even number

Master this difference:

\[
(n \% 2 == 0) \quad \#1
\]

\[
n = 0; \quad \# 2
\]
if (n % 2 != 0) System.out.println("n is odd");
else System.out.println("n is even");

An important point: (n % 2 == 0) is a **boolean** expression (returns a boolean value) --
a **boolean** must go into the test slot of an **if** stmt!

Nothing else will do!!
Write an application that reads two Strings for the keyboard, and then announces (prints, or echoes) the longer of the two.

In case of a tie - either will do.

(One way to proceed: do an easier problem first)