CS 121 - Intro to Programming: Java - Lecture 3

Announcements

Read Chapters 1, 2 this week, but look at ch 3 (Tues).
First OWL assignment due tomorrow at 11
First programming assignment due Friday, 4:30
Office hours have been posted (today, 1-2..)
A typical Java program
A very, very simple application

main

Class G
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
Java’s Object Model

This subject will occupy us for a good deal of the next month or so!
The Object Model
We model “things” as objects
Objects have attributes, and behaviors
Trip (attributes: start, end, days, distance)
   (behaviors: getDays, setDays, distPerDay ..)
Horse (attributes: name, breed, age, height)
   (behaviors: getAge, setAge ..)
Car
House
Student
Tree
Where are all of these 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

getAddress

anotherMonth (make kid one month older)
Infant

InfantTester

main
public class InfantTester{

    public static void main (String[] args){
        Infant myKid = new Infant("Kit",4);
        System.out.println("name: " + myKid.getName());
        myKid.anotherMonth();
        System.out.println("my kid is now " + myKid.getAge());
    }
}
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 is increased, a car’s gas tank is filled, etc. When a mutator does what it does, the state of the calling object changes.
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 number = 7; // number now “holds” 7
number = 4; // number now “holds” 4
number = number + 2; // number now “holds” 6
```

Assignment is NOT equality!

Assignment is an **action** operator: **Compute** the RHS, Then **copy** the result to variable named on the LHS
Our current view of the OO programming landscape..
**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.

```java
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:
   ```java
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)