Assignment A2 CS100J Fall 2007 Due 20 September, IN CLASS

This assignment is the same as "Drawing Objects II", in the Labs given on page .1 of ProgramLive (the CD). It's on the next two pages. This assignment requires you to draw a few objects (manila folders). Please hand it in at the BEGINNING of lecture on Thursday, 20 September. It shouldn't take more than 1/2 hour to do this assignment.

Remember: Any class that does not extend another class automatically extends class Object, which is in package java.lang. So, writing

pablic class Whatever { ...}

has exactly the same effect as

public class Whatever extends Object { ...}

Class Object has two important methods: function equals(Object) and function toString().

It is important that you know how to draw an instance of a class yourself, following our conventions. Only then can you fully understand howclasses and instances of classes work in Java.

One semester, when we gave this homework, some students did miserably. We explained what they did wrong and gave them another chance to learn. So that you don't have this problem --we won't be giving second chances--, here are notes that we gave out to the students who did things wrong. After completing this assignment, read through these notes and see whether you made the same mistakes. If so, correct them.

Note 1. You did not follow directions at all. We do not see, separately, an instance of Ex, an instance of Sub, an instance of SubSub1, etc. You may have put them all together in one picture, but that is not what we asked for and it shows no understanding of drawing folders. 20 points off.

Note 2. In a folder like one for SubSub1, you placed all the components together in one place. You did not put those for Ex on top, then those for Sub next, and finally those for SubSub1 at the bottom. 50 points off.

Note 3. You did not draw field p of Subsub2 as a variable --either with a line after it or a box after it. 5 points off.

Note 4. You left off the part for superclass Ex and/or Sub in your diagrams. You did not follow directions. 20 points off.

Note 5. You did not draw variables correctly. They should be drawn with the name of the variable followed either by an underline with the value on it or by a box with the value in it. 5 points off.

Note 6. Method what and field Zero in class Ex are static. Therefore, they should not be drawn in each folder of class Ex --they belong separately in the file drawer for Ex. 5 points off.

Note 7. In drawing a folder for SubSub2, you put in it a place for SubSub1 components. That is not correct. SubSub2 extends Sub, not SubSub1. 15 points off.

PGL-1. Drawing objects II

Lesson Page 4-1

Introduction

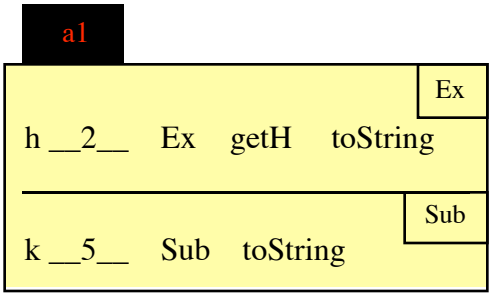
This Lab gives you practice in drawing instances of subclasses. You won't need a computer, but get out a few sheets of paper.

Step 1. Draw an object

Consider the following classes:

```
public class Ex {
  public static final int ZERO= 0;
  private int h;

  public Ex(int ph) {
      h= ph;
  }

  public int getH() {
      return h;
  }

  public String toString() { ... }

  public static int what(int x) { ... }
}

public class Sub extends Ex {
  private int k;

  public Sub(int pk) {
      k= pk;
  }

  public String toString() { ... }
}
```

Below is an instance of class `Sub`. Below the horizontal line are all the instance variables and methods that are declared in `Sub`; above the line are all the instance variables and methods that are declared in superclass `Ex`. Variables `h` and `k` have been given arbitrary values.



On your paper, draw an instance of class `Ex` and another instance of class `Sub`.

Step 2. More subclasses

Consider also these two subclasses of class `Sub`:

```
public class SubSub1 extends Sub {
  public SubSub1() {
      super(5);
  }

  public int hPlus1() {
      return getH()+1;
  }

  public String toString() { ... }
}

public class SubSub2 extends Sub {
  private int p;
}
```

Draw an intance of class `SubSub1` and an instance of class `SubSub2`.

Step 3. Taking class `Object` into account

Class `Object` is automatically the superclass of all classes that don't explicitly extend a class. For our purposes, class `Object` defines two instance methods: `equals` and `toString`.

Redraw all the objects that you drew so far (on another sheet of paper), showing explicitly the contribution of class `Object` to each object. If you do not know what to do, listen to the last activity on lesson page 4-2.