IST 256

Lab Week 10, Wednesday, March 31, 2010

(Make sure your name gets on an attendance sheet.)

1. Writing a program that experiments with a class

In this program, we will write a Java application that contains a class for keeping pet information and recommending how much food to give a pet. The main program will experiment with creating and using objects of the pet class.

a. Start by **creating a Java application** and name it something like TestPet. As usual, NetBeans will bring up a program source code window where there is a class called Main and a main method where we will put our code to read a file.

b. Create a Pet class: In the left pane of NetBeans, find the TestPet project and right click on the top line that says TestPet. In the menu, select New -> Java class. In the new class window:

give the **class name** as Pet
select the **package** of the class to be testpet

Note that in the left pane, there are now two programs listed under the Source Packages/testpet – Main.java and Pet.java.
(In the past, for the sake of simplicity, we have allowed new jFrame Forms to go into the default package, but from now on, we will always place new classes and forms in the package with the project name.)

c. Write the Pet class.

After the class header and inside the curly brackets for this class, we will add all the code for the class. First add the fields of the class. We will declare two variables as fields for the name of the pet and its weight in pounds.

```
// fields for this class are the name and weight (in pounds)
   public String name;
   public int weight;
```

After this code in the Pet class, we will add a Constructor method.

```
// constructor initializes both fields
   public Pet(String startname, int startweight)
   {
      name = startname;
      weight = startweight;
   }
```

Next, we will add a method to change the weight of the pet. Note that this method can just assign a new value to one of the fields.

```
// method to change the weight of the pet
   public void setWeight(int newweight)
   {
      weight = newweight;
   }
```

And finally, we had another method that can be used to find a recommended amount of food to feed this pet on any day. To find the amount of food, you must pass a string representing a level of activity of the pet on that day.

```
// method to recommend how much pet food to give your pet
   //    depending on the day's activity level
   //    levels are "low", "normal", "high"
   public double recommendFood(String activityLevel)
   {
      // initialize food to 0 ounces
      double food = 0;
      if (activityLevel.equals("low"))
      {
         food = weight * 2.0;
      }
      if (activityLevel.equals("medium"))
      {
         food = weight * 2.2;
      }
      if (activityLevel.equals("high"))
      {
         food = weight * 2.4;
      }
      return food;
   }
```

d. Write the main method

Double click on the Main.java program and add code to test the class. First, we create two instances of the class Pet, called myPet and yourPet, and we print out the fields of them both. Copy the following code into the main method of the Main class, right after the TODO comment.

```
// variables for two pets
        Pet myPet, yourPet;
        // amount of food
        double amount;

        // create a Pet for my pet
        myPet = new Pet("Tiger", 20);

        // show values of my pet
        System.out.println("My pet name is " + myPet.name +
                " and weight is " + myPet.weight);

        //create a Pet for your pet
        yourPet = new Pet("Fluffy", 10);

        // show values of my pet
        System.out.println("Your pet name is " + yourPet.name +
                " and weight is " + yourPet.weight);
        System.out.println();
```

Note the use of myPet.name and yourPet.name, for example, to show the values of the name fields in the two class instances.

Run the program and observe its results in the output pane of the NetBeans Window.

Continue by using the methods setWeight and recommendFood. Here are some examples to try:

```
// recommended food if my pet activity is medium
        amount = myPet.recommendFood("medium");
        System.out.println("With medium activity, pet food is " + amount);

        // My pet loses weight
        myPet.setWeight(18);
        // recommended food if my pet activity is medium
        amount = myPet.recommendFood("medium");
        System.out.println("With medium activity, pet food is " + amount);
```

Try using different values for the pet weight and activity level and view the results. Write here one test case, giving the call to setWeight and recommendFood, and showing the result.

2. Writing a GUI application that uses a class

For our second example using a class, we will define a class for Students and write a Java GUI application that can read student data from the form and create and use an instance of the Student class.

a. Start by **creating a Java GUI application** and name it something like TestStudent. Then for that project,

create a new jFrameForm to get the GUI window and
put it into **package teststudent**, and
set the GUI to be the main class of the project.

b. Create a Student class: In the left pane of NetBeans, find the TestStudent project and right click on the top line that says TestStudent. In the menu, select New -> Java class. In the new class window:

give the **class name** as Student
select the **package** of the class to be teststudent

c. Write the Student class.

1. First write the code to declare four field variables to keep student data for a semester:
a String called *name* and three int variables called *grade1, grade2* and *grade3.*

2. Now create a Constructor method that fills in the student name, but sets all the grades to 0. Here is the code you can use:

```
// constructor initializes name field and sets grades to 0
   public Student(String startname)
   {
      name = startname;
      grade1 = 0;
      grade2 = 0;
      grade3 = 0;
   }
```

3. Next we need the code to set values for the three grade variables. These methods should be called setGrade1, setGrade2, and setGrade3. Here is the code for setGrade1:

```
// update methods for each grade
   public void setGrade1(int newgrade)
   {
      grade1 = newgrade;
   }
```

Write the methods for setGrade2 and setGrade3 into your program.

4. Add this code for a method that computes the average of the three grades:

```
// find the average of the three grades
   public double findGradeAverage()
   {
      int sum = grade1 + grade2 + grade3;
      double average = sum / 3.0;
      return average;
   }
```

d. Create the form interface.

On the form, first add four labels and four textfields where the student can give information. Then add a button and label for the result. The form can look something like this:

```
Student name:              |_____________________|
Enter first grade:         |_____________________|
Enter second grade:        |_____________________|
Enter third grade:         |_____________________|
        |___Find Grade Average___|                    (button)
                Results:                             (label)
```

For the button, select Event -> action -> actionPerformed.

e. Add code for the Grade Average button.

In the interests of keeping the lab short, you can use this code for the button:

```
// create a student with  the name from the first TextField
Student st = new Student(jTextField1.getText());

// add the grades from the other three textfields,
//    checking if there is any number conversion error
try
{
   st.setGrade1 (Integer.parseInt(jTextField2.getText()));
   st.setGrade2 (Integer.parseInt(jTextField3.getText()));
   st.setGrade3 (Integer.parseInt(jTextField4.getText()));
}
catch (Exception e)
{ // do nothing and grade will remain 0
}
// get the grade average and display it
double av = st.findGradeAverage();
jLabel5.setText("grade average = " + av);
```

Test this program on several values. Be prepared to give one set of values that you tested and the results.