IST 256
Lab Week 13, Thursday, April 17, 2014

1. Designing a Class

Suppose that we have a file Pictures.txt that contains information about images. Each line of the file has the name of the image, the name of the photographer, the number of prints sold and the type (either Color or B&W for black and white). An example file might look like:

```
Mountain,Adams,40,B&W
Workers,Salgado,31,B&W
Flower,Mapplethorpe,25,Color
Baseball,Allen,200,Color
Woman,Lange,38,B&W
```

a. First design a class that is suitable to keep the data from each line of this file. The name of the class should be Picture and it should have fields for imagename, photographer, numprints, and medium. These fields should all be public and there should be one Constructor method that initializes the fields. No other methods are required at this time.

b. Write the declaration of a one-dimensional array of objects of the type Picture defined by the class in part a. The array should allow up to 12 elements.

2. Extend your StudentData project in NetBeans to have more information and a subclass.

a. First we will extend our Student Data class to have the additional information for students types. For student type, every student will have a string for whether they are "freshman", "sophomore", "junior" or "senior". We will add a field for studentClass to the Student java class. This information will be considered as part of the student's type.

Here is our redesigned Student java class, omitting the DormAddress, which you are also putting in this week:

```
/*
 *  Class to represent data about each student
 */
public class Student {
//   name, gender (either M or F), age in years, height in inches, student type
    protected String studentName, gender;
    protected int age, height;
    protected String studentClass;

    // Constructor gives an initial value to all fields
    //     (using a different style notation for initial values)
    public Student(String _name, String _gender,
                              int _age, int _height, String _studentClass)
    {
       studentname = _name;  gender = _gender;  age = _age;
       height = _height;  studentClass = _studentClass;
    }
    // accessor methods
    public String getStudentName ()
    {
          return studentname;
    }
    // other accessor methods remain here
    // method to get all student type information – in this case, we only have the class
    public String getStudentType ()
    {
          return studentClass;
    }
   public String toString()
    {
       String result = "Student: " + studentName + ", gender: " + gender
             + ", age: " + age + ", height(inches): " + height
             + " dorm: " + dormAddr.toString() + "class:" + studentClass;
       return result;
    }}
```

b. Suppose that some students have additional information because they have financial aid packages. In the real world, there is probably a lot of information that goes into having financial aid, so it would make sense to have different subclasses of students. In this lab, we will simulate that by having one additional piece of information, the type of financial aid, which will be a string whose value is either 'loan' or 'scholarship'. This will be considered part of the student type information to be shown about a student on the GUI.

We will create another data class called StudentFA for students with financial aid. This class will extend the Student class and will have one additional field called typeFA for type of financial aid.

c. Add data to the file studentFile.txt (or whatever you called the student data file). To the end of every line of studentFile data first add

- a comma and one of the words freshman, sophomore, junior or senior

Then add one of the following:

- a comma
- a comma followed by the word loan or the word scholarship

In the latter case, these students will be represented as financial aid students

d. Make the changes to the Student class sketched above to add the studentClass field and the getStudentType method. Note that we changed all the fields from private to protected so that they can be used by the StudentFA subclass.

The subclass needs a "default" parent constructor, so you will also need to add to the Student class:

an additional no argument constructor:

```
public Student() { }
```

e. Now create the StudentFA class by making a new class in the same src package as the Student class.

- To the header line of the StudentFA class, add that it extends Student (Note that when add this, you will get a red underlined error that is just NetBeans saying that you haven't completed it yet. Just ignore the red and continue to complete the class.)
- Add an additional field for a String that will contain type of financial aid, called typeFA.
- Add a constructor function that initializes all fields in Student and also initialize this new field.
- Give a different method for getStudentType() that returns a String with both the studentClass and the typeFA in one string, for example studentClass + ", " typeFA. (This method will override the parent Student method getStudentType().
- Add a different method for toString() that adds the FA type to the end of the string. (This method will override the parent Student method toString().

f. I think that the only changes necessary for the StudentDataGUI code is in the button that reads the file.

- add an additional two variables for reading the student class and the (optional) financial aid from the file and add calls the scanner next() method to read them from the file
- change your code for creating a student to create either a regular Student or a StudentFA as follows:

```
// in both cases, the type of the object can be the parent class Student
Student oneStudent;
if (typeFA.equals(""))
{
    // add these items to a new Student object
    oneStudent = new Student(name, gender, age, height, dorm, room, studentClass);
}
else
{
   // if typeFA number, add these items to a new StudentFA object
   oneStudent =
          new StudentFA(name, gender, age, height, dorm, room, studentClass, typeFA);
}

// add this student data to the array
studentArray[numStudents] = oneStudent;
```

Note that it's o.k. to assign an object created as a StudentFA to a variable of type Student.

This lab is due on Tuesday, April 22. Please submit your changed Student.java class and your new StudentFA class.