IST 256
Lab Week 12, Tuesday, April 8, 2014

1. Practice Using Arrays and Classes.
Suppose that we have a class that represents people with two fields for name and age.

```
public class Person
{
        // fields for the name and age
        private String personName;
        private int age;

        // constructor initializes both fields
        public Person ( String startName, int startAge)
        {
                personName = startName;
                age = startAge;
        }

        // accessor functions for the name and age
        public String getName()
        {
                return name;
        }
        public int getAge()
        {
                return age;
        }
        ….
}
```

Suppose that we have a main method with an array declared of type Person, and that data is read into all 50 of the Persons in the array.

```
Person [ ] personArray = new Person [ 50 ];
```

a. Write the expression whose value is the name of the first person in the array.

b. Write the first line of an if statement that will test if the name of the first Person in the array is "Alan".

c. Write code that will compute the average of the ages of all 50 Persons in the array.

2. Finish the program from the lab last week

In the last lab, we wrote a project for StudentData that had one button that would read a file of data for class Student and store it into an array of Student, and another button that went through the array and displayed every student. If you are not done with this yet, finish this lab first.

3. Extending the program with Student Data

Today, we will extend the StudentData example with two more buttons. The first will display the tallest student, and the second will allow the user to search for a student by (partial) name.

a. Start by **opening the StudentData** project (or whatever you called it).

b. Extend the form interface.

On the form, we will keep the previous buttons and labels for reading the file and saving the data, and for displaying the students. We will add a button to display the tallest student and a label to show (only) the name of the tallest student. Now we will add a textfield to allow the user to type in a search name and another button to perform the search. The form can look something like this:

```
|__ Read Student File__|      File Status

|___Display Students___|              |___Display Tallest Student___|
        Students:                                  Tallest:

Enter (partial) search name:  |_____________|        (label and textfield)
        |__Search Student Names__|                       (button)
Results:                                                (multiline label)
```

For the new buttons, select Event -> action -> actionPerformed.

You may find that you explicitly need to resize the Students: label in the Design pane for all the results to show. Or you can rearrange the form elements so that they are to the right of the Students: label.

c. Add accessor methods to the Student data class

In the new buttons, we will need to get the student name from each student class instance. So we should add an accessor method getStudentName to the class. For the Tallest

Student button, we will also need to get the height for each student, so we should also add an accessor method getHeight to the class.

Go to the code for Student.java and add accessor methods called getStudentName() and getHeight(). Usually these methods are added after the Constructor method.

d. Add Code for the new Tallest Student button

This button will search the array of Students to find the index of the one which has the largest height in inches.

From the notes for ArrayAverageAndMax.txt, adapt the code for finding the largest element in an array to find the tallest student. Note that where the example code compares an array value, your Student code must get the array value, which is something of type Student, and then compare the height field of the student. Display the name of the student in the Tallest label.

e. Test your program.

Which student is displayed as the tallest? Is this the first or last of the tallest students?
Write your answers here:

f. Add Code for the new Search button

This button will search the array of Students to find ones whose name matches according to the following criteria:

the **search string can be found as a starting substring of the student's name** and **case should be ignored**.

Find and display all the students that match. If no students match, display a message to that effect.

- Declare a variable to hold the search name String and get the search name from the TextField.
- **Write a search that finds all students** that match the search criteria.
- Write the search criteria that combines ignoring case with the comparison method startsWith().
- Display the all the students that matched or the no match message.

g. Test your program.

Test your program by typing in example search strings that match and don't match.

Write here **example search strings that you tested**:

1. An example that didn't match anything:

2. An example that matched one student:

3. An example that matched more than one student:

4. An example that shows the search is ignoring case:

Submit this lab sheet together with your new version of the StudentDataGUI code that contains the two new button methods. (You may copy and past only the two new methods.) This is due by Tuesday, April 15.