IST 256
Lab Week 9, Tuesday, March 22, 2011

1. Understanding Arrays

Assume that a program has the following statements:

```
int [ ] values = new int[5];

for (int index=0; index < values.length; index++)
{
        if ( index == 0 )
        { values [ index ] = 10;    }
        else
        { values [ index ] = values [ index – 1 ]  + 2;   }
}
```

After this loop executes, write down the numbers in the array *values*.

values[0]	
values[1]	
values[2]	
values[3]	
values[4]	

2. Writing a method

Suppose that we have the following code that is used to compute the price of books and add in sales tax.

```
int numBooks = 5;
double pricePerBook = 20.00;
double taxRate = .08;
double bookCost;

bookCost = pricePerBook * numBooks;
bookCost = bookCost + (bookCost * taxRate);
```

Design and write a method definition that will compute the cost of any number of items at any price and with any amount of sales tax, and return the final cost of the items with tax.

Write a method call to compute the price of books, given the original variables.

3. Writing a program that reads and writes words to a file
In this program, we will write a Java application that can read words from the file "xanadu.txt" and write them to a file "xanaduwords.txt" where each word appears on a line by itself.

a. Start by **creating a Java application** and name it something like FileWriteWords. 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 Resources folder in the project. Right click on the FileWriteWords project name in the project pane in NetBeans and create a new **Java package** called Resources. We will put this package under the src folder in the project (the default).

b. Save a data file for the program to read. From your browser, save the file called xanadu.txt from last week and put it into the folder for your NetBeans project. (In Firefox, to save a file, right click on the link and select Save Link as …) Then navigate to the folder where you keep your projects and go into the newly created src/Resources folder for FileWriteWords and save the xanadu.txt file there. (Or copy the file from your NetBeans project FileReadWords from last week.)

c. Write the application that reads and writes words.
At the top of the program, right after the package statement, add import statements:

```
import java.io.*;
import java.util.Scanner;
```

From today's web page click on the link called FileWriteWords. Copy the part of the main method from the line that starts "Scanner s … " on down to the end, but not including the last two brackets.

Paste this program into your NetBeans main method. Fix the tabs, spacing and matching brackets. Observe the use of the output file to write out the words.
Run the program and observe its results in the output pane of the NetBeans Window.

d. Look at the output file
Observe that the output file appeared under the Resources folder in the NetBeans project FileWriteWords. Open the file xanaduwords.txt in NetBeans and observe its contents.

e. Cause an error.
In the program, find the line with the outFile.close() method and cause it to not be executed by "commenting it out", which means to change the line to:

```
// outFile.close();
```

Rerun the program and observe the contents of the output file.

f. Change the file close statement back to the correct one.

2. Extending the program that read temperatures from a file

In this lab, we will work more on the program that was written last time, called FileReadTemps.

Look at this program to review what it does so far:

- Button to read the file: reads a file of temperatures, with one temperature per line, and stores them in an array
- Button to compute average: computes the average of the temperatures in the array and displays it

a. Add buttons to the GUI

Add a button to the GUI to find the maximum element of the array and another button to write a report to a file. The GUI should also have a label to display the maximum and for file status. It should look something like this:

|__ Read Temperatures from File Feb05temps.txt__| File Status

|__ Compute Average Temperature__| Average

|__ Find Hottest Temperature__| Hottest

|__ Write Temperature Report__| File Status

Rename the buttons and make event -> action -> actionPerformed methods.

b. Write the button method to compute the maximum of the array

To find the hottest temperature, we will search the array to find the maximum value of the array. Our technique will be to have an index variable that keeps track of the index of the hottest temperature that we have found so far as we go through the array. For each element, we compare it with the hottest so far, and if it is bigger, we make its index be the hottest so far. At the end, we will have found the maximum.

Here is the code to put into the button method:

```
int indexSoFar;
int hottestTemp;
// to begin, the index of hottest temperature so far is at 0
 indexSoFar = 0;
// loop over remaining array to keep finding index
//          of new hottest temperature so far
for(int i = 1; i < numtemps; i++)
{
   if (temperatures[i] > temperatures[indexSoFar])
  {
      indexSoFar = i;
   }
}
hottestTemp = temperatures[indexSoFar];
hottestLabel.setText("Hottest temperature:  " + hottestTemp);
```

Test your program.

c. Add a temperature to the file

Find the Resources folder in the Project pane and double-click on the Feb05Temps.txt file to open it into NetBeans. Add another temperature at the bottom of the file.

What number did you put?

Now what is the average of the temperatures in the array?

And the hottest temperature?

Note that NetBeans saved the .txt file when you ran the program.

d. Write a temperature report to an output file for the fourth button.

Now we are going to write the average and hottest temperatures in a report to an output file.

Note that we don't need to add another import statement because we already added import java.io.*;

In order to use the average and hottest temperature values in our report, the fourth button needs to be able to get the values of those variables. Therefore, we must move the declarations of the variable "average" and the variable "hottestTemp" to the class level, so that they are global to the buttons.

Also move the line with the creation of a decimal formatter from inside the Average button to the class (global) level.
Inside the fourth button method:

1. a try catch block
2. inside the try block, create an output Stream to a file:

```
// create an output file
BufferedWriter outStream = new BufferedWriter(
        new FileWriter("src/Resources/Feb05tempsReport.txt"));
```

3. write the header line for the report:

```
// write a title line in the file with one blank line
outStream.write("February 2005 Temperature Report");
outStream.newLine();  outStream.newLine();
```

4. write the average to the report in the file:

```
// write the average temperature
outStream.write("Average temperature for the month:  ");
outStream.write(df.format(average));
outStream.newLine();
```

5. write the hottest temperature to the report in the file:

   (write your own code similar to the code to write the average.)

6. the last thing inside the try block is to close the output file:

```
outStream.close();
```

7. inside the catch block, write code to handle any IO exceptions

   (you can use the same exception handling as for the read file code.)

Test your program and view the results of the file.

This lab is due by the beginning of class on Tuesday, March 29.