Lecture Five - Arrays

- Intro to Arrays
- Lab Grades program
- Sort operator
- Lab Grades 2 program
- Summer Hours program
- Two dimensional arrays
- Drill

Read Chapter 4
Arrays

- Arrays are variables that can contain many elements, rather than only one value
  - i.e. int only holds one integer value while an array of type int can store many integer values
- This makes it easier to group similar information
- Arrays can be of any data type (int, double, char, String, etc)
Arrays (continued)

- Creating and initializing an array can be done a couple different ways
  - An array with known data, in this case integers:
    - int[] intArray = {1, 300, 68};
  - An empty array with known length (the number of elements in the array), in this case length = 3.
    - String[] strArray = new String[3];
Arrays (continued)

- To access a certain value in array one must know the index number of that value
  - Example:
    - int[] intArray = {1, 300, 68}
    - In this case the index 0 contains the value 1, index 1 contains 300, index 2 contains 68
      - In code words intArray[0] is 1, intArray[1] is 300, intArray[2] is 68
      - Note how this is different from the length, the length of intArray is 3. You can get the length of the array in code by saying intArray.length

- Remember:
  - Indexes start from 0.
    - intArray[2] is 68
  - Length is the total number of elements
    - intArray.length is 3
Arrays (continued)

- Here is a visual look at an array

```
intArray[0] = 98
intArray[1] = 42
intArray[2] = 70
intArray[3] = blank
intArray[4] = blank

length = 5
```
Arrays (continued)

- **Pop Quiz:**
  - Suppose we have this segment of code:
    - `double[] doubleArray = {65.3, 36.1, 100, 90};`
  - What value would `doubleArray[2]` return?
  - What value would `doubleArray.length` return?
Arrays (continued)

**Pop Quiz:**

- Suppose we have this segment of code:
  
  ```java
  double[] doubleArray = {65.3, 36.1, 100, 90};
  ```

  - What value would `doubleArray[2]` return?
    - **Answer: 100**

  - What value would `doubleArray.length` return?
    - **Answer: 4**

- The next two slides contain code that shows implementations of arrays in a program called LabGrades.
// Lab Grades Code
import javax.swing.JOptionPane;
public class LabGrades
{
    public static void main(String[] args)
    {
        String studentname; // variable to hold the name of a student
        double[] labgrades = new double[7]; // an empty array of doubles to hold 7 lab grades
        double sum = 0; // will be the sum of the labgrades
        double average; // will be the average of the labgrades
        String grades; // will be used to simplify output later
        double lowest; // used to keep track of the lowest value
        int i; // counter for the for-loops
        // input
        studentname = JOptionPane.showInputDialog(null, "Please enter student name", "Input Name", JOptionPane.QUESTION_MESSAGE);
        for (i = 0; i < 7; i++) // goes for i = 0 to i = 6
        {
            labgrades[i] = Double.parseDouble(JOptionPane.showInputDialog(null, "Please enter Lab Grade "+(i + 1), "Input Grade", JOptionPane.QUESTION_MESSAGE)); // converts each inputted string to a double
        }
        ...
Lab Grades - Code

... // process

lowest = labgrades[0];
for (i = 1; i < 7; i++)
{
    if (lowest > labgrades[i])
    
        lowest = labgrades[i];

}

// give a starting value for lowest

// don't have to start at 0, because 0 is already stored in lowest

for (i = 0; i < 7; i++)
{
    sum += labgrades[i];
}

// += is short hand for sum = sum + labgrades[i]

sum -= lowest;
average = sum / 6;
grades = "Lab Grades : ";
for (i = 0; i < 7; i++)
{
    grades += " " + labgrades[i];
}

// subtract the lowest grade

// average of the 6 grades

// creating a string in order to organize the output better

together

JOptionPane.showMessageDialog(null,"Name: " + studentname + "n" + grades +"nAverage: " + average,

"Grades", JOptionPane.INFORMATION_MESSAGE);

}
Lab Grades Program

- What the code generally does:
  - Receives a student name and seven grades from the user
  - Calculates the average of the grades while ignoring the lowest value
  - Shows an outbox with the student’s name, the seven grades, and the average of the six highest

- Now let’s take a closer look at the code to see how all of this was done
public static void main (String args[]) {
    String studentname; // variable to hold the name of a student
    double[] labgrades = new double[7]; // an empty array of doubles to hold 7 lab grades
    double sum = 0; // sum of the labgrades
    double average; // average of the labgrades
    String grades; // simplify output of grades
    double lowest; // used to keep track of the lowest value
    int i; // counter for the for-loops
    ...
}

1. These are declarations just as you have seen before, each is noted on what they will do for the program.

2. The array labgrades is the only new thing in this section of the code. It is declared as an empty array of 7 doubles, indexed 0-6.
public static void main (String args[])
{
    . . .
    // input
    studentname = JOptionPane.showInputDialog(null, "Please enter student name", 
        "Input Name", JOptionPane.QUESTION_MESSAGE);
    for (i = 0; i < 7; i++) // goes for i = 0 to i = 6
    {
        labgrades[i] = Double.parseDouble(JOptionPane.showInputDialog(null, 
            "Please enter Lab Grade ", i + 1, "Input Grade", 
            JOptionPane.QUESTION_MESSAGE));
        // converts each inputted string to a double
    }
    . . .
}

3. In lesson 3 you used the `Double.parseDouble(...)` to convert a String into a double. Here there is no difference. You may be confused as to why the input dialogue is within the parenthesis. Just remember that the `JOptionPane.showInputDialog` returns a String the user inputs. Here the String the user inputs is immediately converted into a double without the middle steps of assigning holder variables.

4. The `for-loop` makes it easier for us to input numbers into the rest of the array. Remember `i` increments by one each time through the loop, meaning `labgrades[0]` takes a number the first time through, then `labgrades[1]` takes a new number the second time and so forth.
public static void main (String args[]) {
    ...
    // process
    lowest = labgrades[0]; // give a starting value for lowest
    for (i = 1; i < 7; i++) // don’t have to start at 0, because 0 is already stored in lowest
    {
        if (lowest > labgrades[i]) // if the known lowest value is greater than the current value
            lowest = labgrades[i]; // then lowest now equals the current value
    }
    for (i = 0; i < 7; i++)
    {
        sum += labgrades[i]; // += is short hand for sum = sum + labgrades[i]
    }
    sum -= lowest; // subtract the lowest grade
    average = sum / 6; // average of the 6 grades
    ...
}

5. The double lowest is assigned the value of labgrades[0], this is to insure there is a starting base for comparison when looking at the values stored in the array. Remember lowest was never given an initial value and we do not know what numbers are in labgrades, so why not assign it an easy variable for comparison?

6. The for-loop runs through all the numbers held in the array, starting at index 1. The if statement compares the lowest value to the current value, if lowest is greater, then obviously it is not the lowest value in the array. When the if statement is true the current value is stored in lowest. If the statement is false, it skips lowest = labgrades[i] and continues through the loop evaluating the next number in the array.

7. The following for-loop adds the numbers stored in the array together.

8. The last couple of lines subtract the lowest value from the sum and then average the six numbers.
public static void main (String args[]) {
    ...
    grades = "Lab Grades : ";
    // creating a string in order to organize the output better
    for (i = 0; i < 7; i++) {
        grades += " " + labgrades[i];
        // adding the numbers to a string, this is -not- adding the numbers together
    }
    JOptionPane.showMessageDialog(null,
    "Name: " + studentname + "\n" + grades + "\nAverage: " + average,
    "Grades", JOptionPane.INFORMATION_MESSAGE);
}

9. The only odd thing done in this part of the code is the concatenated string for outputting all of the lab grades back onto the screen. A loop puts a space then the number grade into the string so that in the output we can just use the variable ‘grades’ to display all of the numbers. (Note: \n in a String means the next line)

10. Run this program and compare
Lab Grades Program

- How could we have made the program simpler?
  - Used the API’s sort method for arrays to find which was the smallest value
    - The Array.sort(arrayName) sorts the elements in the array from lowest to highest.
    - Index 0 would have contained the lowest value and we could have simply ignored labgrades[0] in our summation of the grades.

- The next slide is code for Lab Grades 2. This program will display the grades from highest to lowest. Can you think of how to do it using sort?
// Lab Grades 2 Code
import javax.swing.JOptionPane;
import java.util.Arrays;

class LabGrades2 {
    public static void main(String[] args) {
        String studentname; // variable to hold the name of a student
        double[] labgrades = new double[7]; // an empty array of doubles to hold 7 lab grades
        double sum = 0; // will be the sum of the labgrades
        double average; // will be the average of the labgrades
        String grades; // will be used to simplify output later
        int i; // counter for the for-loops

        studentname = JOptionPane.showInputDialog(null, "Please enter student name", "Input Name", JOptionPane.QUESTION_MESSAGE);
        for (i = 0; i < labgrades.length; i++) { // goes for i = 0 to i = 6
            labgrades[i] = Double.parseDouble(JOptionPane.showInputDialog(null, "Please enter Lab Grade "+(i + 1), "Input Grade", JOptionPane.QUESTION_MESSAGE));
            // converts each inputted string to a double
        }
    }
}
... // process
    Arrays.sort(labgrades); // sorts the grades from lowest to highest
    // want to drop the lowest grade, located at element 0 of the labgrades array
    for (i = 1; i < labgrades.length; i++)
    {
        sum += labgrades[i];
    }
    average = sum / 6;
    // output
    grades = "Lab Grades : "; // creating a string in order to organize the output better
    for (i = labgrades.length-1; i >= 0; i--)
    // want to display highest first, therefore reversing our way through the array
    {
        grades += " " + labgrades[i];
        // adding the numbers to a string, this is -not- adding the numbers together
    }
    JOptionPane.showMessageDialog(null, "Name: " + studentname + "\n" + grades +"\nAverage: " + average, "Grades", 
    JOptionPane.INFORMATION_MESSAGE);
import javax.swing.JOptionPane;
import java.util.Arrays;

public class LabGrades2 {
    public static void main(String[] args) {
        …
        // process
        Arrays.sort(labgrades); // sorts the grades from lowest to highest
        // want to drop the lowest grade, located at element 0 of the labgrades array
        for (i = 1; i < labgrades.length; i++) {
            sum += labgrades[i];
        }
        average = sum / 6;
        …
    }
}

1. `Arrays.sort(labgrades)` sorts the numbers in the array from least to greatest. It makes it easy to pick out which value will be the lowest instead of the comparison we did before. The sort method places the lowest value in index 0 and the highest (in this case) at index 6.
2. The sort method is in the API, because of this we must add the import line `java.util.Arrays` in order to have access to the method (just as we had to do to use `JOptionPane`).
3. The for-loop sums up all of the numbers except the value in index 0, this way we do not have to subtract. The lowest value is already ignored through the for-loop starting at 1.
4. `labgrades.length` is 7 and we’ve used this knowledge for our loops. There may come a time when the amount of elements in an array is unknown to us, however we will always be able to use `length` to determine the size. This part of the code just shows ways `length` is commonly used.
public static void main(String[] args) {
    ...
    // output
    grades = "Lab Grades : ";
    // creating a string in order to organize the output better
    for (i = labgrades.length - 1; i >= 0; i--)
    // want to display highest first, therefore reversing our way through the array
    {
        grades += " " + labgrades[i];
        // adding the numbers to a string, this is -not- adding the numbers together
    }
    JOptionPane.showMessageDialog(null, "Name: " + studentname + "\nAverage: " + average, "Grades", JOptionPane.INFORMATION_MESSAGE);
}

5. The only different in this section of the code to the original Lab Grades is where the for-loop begins in ends. The task of this program was to display the grades from highest to lowest, however the sort function placed the lowest value in the index 0 position of the array. Loops we have been using so far started at i = 0, now what to do to fix it. Easy, start at the highest index and go down to 0 (by using i--). This way the highest will be displayed first and the lowest last in the order of grades. (i begins at 6 and the loop stops once i is less than 0)
Lab Grades 2 Program

- Remember:
  - A `for-loop` can start and end at any integer. A `for-loop` can increase by two each time around or decrease by four, just any way you can think of using it.
  - Arrays make it easy to store similar types of data in an organized manner.
- Arrays can store data in a very organized and common sense manner. We can make an array that can be the days in months of years. This organization is done through multidimensional arrays.
  - In the next slide there is a program which deals with uses of a multidimensional array.
Suppose you are asked to write a program to store the hours worked by an employee each day over a 10 week period (5 working days per week).

Then the program should display the hours worked for each week, and further, allow the user to ask to see the hours worked for any day of any week.

This requires a 2-dimensional array.

Over the next few slides we will discuss the code for this program. To keep it simple we will fill the array with random data, rather than allowing the user to enter the hours worked each day.
```
// Summer Hours Code
import javax.swing.JOptionPane;

public class SummerHours {
    public static void main(String[] args) {
        double[][] hoursWorked = new double[10][5]; // worked 5 days a week for 10 weeks
        double[] hoursPerWeek = new double[10]; // find out how many total hours per week
        int week = 0, day = 0;
        String output = ""; // week worked, day worked
        // making a nice output line for the JOptionPane
        // completely making up random hours for work
        for(week = 0; week < 10; week++) // going through every week
            { // if its an even week
                if(week % 2 == 0)
                    for(day = 0; day < 5; day++)
                        hoursWorked[week][day] = 8.2; // store 8.2 into this index
                else
                    for(day = 0; day < 5; day++)
                        hoursWorked[week][day] = 5.7; // store 5.7 into this index
            }
    }
```
// calculate the hours worked per week
output = "Hours Worked Per Week\n"; // starting the String output line already
for(week = 0; week < 10; week++) // go through every week
{
    for(day = 0; day < 5; day++) // go through every day
    {
        hoursPerWeek[week] += hoursWorked[week][day];
    }
    output += " Week " + (week+1) + "- Worked " + hoursPerWeek[week] + " hours\n";
}
// output
JOptionPane.showMessageDialog(null, output, "Summer Hours Worked", JOptionPane.INFORMATION_MESSAGE);

// getting information on what week and day the user wants to see the hours on
week = Integer.parseInt(JOptionPane.showInputDialog(null, "Please Enter Week (1 - 10)", "Input Week", JOptionPane.QUESTION_MESSAGE));

day = Integer.parseInt(JOptionPane.showInputDialog(null, "Please Enter Day (1 - 5)", "Input Day", JOptionPane.QUESTION_MESSAGE));

// outputting
JOptionPane.showMessageDialog(null, "Worked " + hoursWorked[week-1][day-1] + " hrs on day " + day + " of week " + week, "Summer Hours Worked", JOptionPane.INFORMATION_MESSAGE);
public static void main(String[] args) {
    double[][] hoursWorked = new double[10][5]; // worked 5 days a week for 10 weeks
    double[] hoursPerWeek = new double[10]; // find out how many total hours per week
    int week = 0, day = 0; // week worked, day worked, i and j are loop counters
    String output = ""; // making a nice output line for the JOptionPane
    ...

    hoursWorked is initialized the same as any other array, however this array has two dimensions. (See the next slide for a visual explanation)
Multidimensional Arrays

Remember the visual from before of `intArray`, now let's take a visual look at what `hoursWorked` would be:

Notice that the length of the first area is 10, and the other length is 5. This simply means that there is an array of 10 items and in each of those items is space for 5 elements.
// completely making up random hours for work
for(week = 0; week < 10; week++) // going through every week
{
    if(week % 2 == 0) // if its an even week
        for(day = 0; day < 5; day++)
            hoursWorked[week][day] = 8.2; // store 8.2 into this index
    else // if its not even
        for(day = 0; day < 5; day++)
            hoursWorked[week][day] = 5.7; // store 5.7 into this index
}

2. For this part of the code we are simply assigning “random” values to our multidimensional hoursWorked array. Basically, we are assuming that if it is an odd numbered week then we work 8.2 hours a day and if it is an odd week then we only work 5.7 hours a day.
// completely making up random hours for work
for(week = 0; week < 10; week++) // go through every week
{
    for(day = 0; day < 5; day++) // go through every day
    {
        hoursPerWeek[week] += hoursWorked[week][day];
    }
    output += "  Week " + (week+1) + "- Worked " +
        hoursPerWeek[week] + " hours\n";
}

3. Here we sum up the number of hours we worked each week for the 10 week period. Instead of having 10 variables for each week, we store the number of hours worked in our hoursPerWeek array.

4. After summing up the hours we create our output string. One thing that may seem strange is using (week+1) in the output statement. Remember than arrays start with index 0, so for the purposes of output we are calling the week at index 0 “Week 1”.

Summer Hours – Closer Look

// getting information on what week and day the user wants to see the hours on

    week = Integer.parseInt(JOptionPane.showInputDialog(null, "Please Enter Week (1 - 10)",
        "Input Week", JOptionPane.QUESTION_MESSAGE));

    day = Integer.parseInt(JOptionPane.showInputDialog(null, "Please Enter Day (1 - 5)",
        "Input Day", JOptionPane.QUESTION_MESSAGE));

// outputting

    JOptionPane.showMessageDialog(null,"Worked " + hoursWorked[week-1][day-1] +
        " hrs on day " + day + " of week " + week,
        "Summer Hours Worked", JOptionPane.INFORMATION_MESSAGE);

5. The final part of our program asks the user to enter a week and a day. Based on that input we then show the user the number of hours worked on that day. Again, remember that our index may not be the same as the number input by the user. So, if the user selects “Week 1” we really want the week at index 0 therefore we must use [week-1] and [day-1] as our indexes.
Drill 5

1. Ok, you should have typed and run the SummerHours program. If not, do that now. Then go back and modify the code to let the user enter the times worked for each day of each week. Make sure you use good prompts, telling the user exactly what it is you expect of him or her. The output should be the same as it was, but it should reflect the data entered by the user.

2. We all know we can use Strings to hold multiple characters but we can also use an array of chars. Create an empty array of 15 chars. Store your name in the array, and then use a for loop to output your name to the screen, one character at a time.

3. Create a 3x3 multidimensional int array that will serve as a tic-tac-toe board. Using two nested for loops (a loop within a loop), ask the user to enter 1 for X and 0 for O for row 1/column 1, row 1/column 2, etc. Store each value in the corresponding spot in the array. Finally output the tic-tac-toe board to the screen. For example:

```
X O X
O X O
O O X
```

- Now, see if you can figure out and output whether X's, O's, or the CAT won.