Java Basics

Session 4

Overall Schedule

- Session 1: Java installation, key programming concepts, overview of Java, structure of a Java program
- Session 2: Eclipse IDE,Variables, data types, mathematical operations, Strings
- Session 3: Decisions and control flow
- Session 4: Arrays and methods

Overall Schedule

- Session 5: Classes, objects, and inheritance
- Session 6: Interfaces, the List and Comparable Interfaces

Session 4

Session Topics

- Basic Console Input
- Methods
- Arrays

Basic Console Input

The Scanner Class

- Introduced in Java 5 to improve reading from external sources, or streams
- Most commonly used to read from the console and from files
- We'll look at one, particular way of working with it
  - Eliminates some issues
  - We won't look at dealing with bad use input.

A Simple Scanner Example

```
import java.util.Scanner;

public class SimpleScannerExample {
	public static void main(String [] args) {
		Scanner kb = new Scanner(System.in);
		System.out.print("Please enter your name: ");
		String name = kb.nextLine();
		System.out.println("Hello, " + name + ".");
	}
}
```

A Guessing Game Example

```
public class GuessingGameExample {
	public static void main(String [] args) {
		Scanner kb = new Scanner(System.in);
		int target = (int)(Math.random() *100 +1);
		int guess = 0;
		do {
			System.out.print("Enter a guess 1-100: ");
			guess = Integer.valueOf(kb.nextLine());
			if (guess > target) System.out.println("Too High");
			if (guess < target) System.out.println("Too Low");
		}while(guess != target);
		System.out.println("You got it!");
	}
}
```

Methods

Method Declaration Syntax

access [static] *type name*(*parameter list*) {

    method body

    [return *type;*]

}

- access: Just "public" for now.
- static: Will discuss more on Wednesday. Needed for now.
- type: data type of the return or void
- name: Any legal name for the method. camelCase, like variables.
- parameter list: list of type-name declarations
- method body: whatever it does
- return: Value, if any, returned from the method

Simple Method Example

```
public class SimpleMethodExample {
	public static void main(String [] args) {
		int num = 1;
		while (num <= 10) {
			int doubled = doubler(num);
			System.out.println(num + " x 2 = " + doubled);
			num = num + 1;
		}
	}
	public static int doubler(int in) {
		int doubled = in * 2;
		return doubled;
	}
}
```

Method Practice

```
public class MethodPractice {
	public static void main(String[] args) {
		//create a Scanner object here
		double input = 0;
		do {
			System.out.print("Enter a number (negative to end): ");
			input = //get the input from Scanner
			if (input >= 0) {
				double negRoot = negativeSqrt(input);
				System.out.println("The negative square root of "
					+ input + " is " + negRoot + ".");
			}
		}while (input >= 0);
	}
//create the negativeSqrt method here
}
```

Arrays

Arrays

- An array is a contiguous block of memory containing multiple items of the same type
  - primitive values
  - object references
- The individual elements of the array are accessed by the [] operator, indexed from 0.
  - myArray[3] = 5; sets the 4th value in myArray to 5
  - an array being accessed by index is exactly the same as a variable of the same type

Declaring an array

- An "empty" array of 10 ints:
  - int [] myArray = new int[10];
  - This style of creation initializes to the type's default value. 0, false, or null. (see session 2)
- An array of 10 int values:
  - int [] myArray = {4, 8, 2, 3, 2, 9, 143, 7, 2, 0, -32}
  - In Java, a "partially initialized" array uses this but with sufficient default values to fill in the size.

Example: Method to find the average of the values in a double array

```
public static double average(double [] arr) {
    double sum = 0;
    for(int i = 0; i < arr.length; i++) {
        sum += arr[i];
    }
    return sum / arr.length;
}
```

The "foreach" or "enhanced for" loop

```
public static double average(double [] arr) {
    double sum = 0;
    for(double element : arr) {
        sum += element;
    }
    return sum / arr.length;
}
```

The foreach loop

- Traverses all of the values in an array or collection.
- Creates a *copy* of each element in the array
  - Cannot change the values in the array
  - *Can* change the properties of a mutable object in the array
    - We don't have any mutable objects just yet. We'll see this later.

Array and Input Practice

- Ask the user how many values they have to enter, using Scanner to accept their input.
- Create an array of that size
- Ask the user for each of the values and out them into the array
- Loop through the array to find the largest value

Max values

Finding the largest value in an array is similar to keeping a running total in that you set a max value to the first value in the array and, for each subsequent value, ensure that it continues to hold the max value found so far.

Quick Look – Multidimesional Arrays

- The type of an array can be an array, making a 2- or multi-dimensional array
  - int [][] myArray = new int[5][10]
  - Creates an array of 5 arrays of 10 ints, or a 5x10 array of ints.
  - int [][] myArray = {{1,2,3},{8,7,6},{3,2,1},{9,2,3}}
  - Creates a populated array of 4 arrays of 3 ints each. A 4x3 array.
  - int [][] myArray = new int [5][];
  - Creates an array of 5 uninitialized int arrays.
  - myArray[0] = new int[6] sets the first of these to an array of 6 ints.
  - imyArray[1] = new int[9] sets the second to an array of 9 ints.
  - Multi-dimensional arrays need not be evenly sized.

Quick Look – Multidimesional Arrays

- Access follows the same rules – someArray[#] gives you a variable of that type.
  - myArray[2] gives you the third int array.
  - myArray[2][3] gives you the 4th value in the 3rd array.
  - int [] my1D = myArray[3] gives you a 1D array to work with that's the same as myArray[3].
    - Any changes to my1D[x] reflect in myArray[3][x] and vice versa.

Idiomatic 2D array loops

```
for(int i = 0; i < myArray.length; i++)
      for(int j = 0; j < myArray[i].length; j++) {
            loop code
      }
}
for(int [] i : myArray)
      for(int j : i) {
            loop code
      }
}
```