Reading:

1. Deitel & Deitel, Chapter 6

Objectives:

1. To learn from mistakes made on the midterm.
2. To learn how to use, create, and customize Java methods

Concepts:

1. Midterm Solutions
2. Java Methods
3. Overview of Lab 2

Outline:

1. Midterm Solutions
2. Java Methods
   a. Coercion
   b. Promotion
   c. Java API packages
3. Random Number Generation
4. Scope
5. Method Overloading
6. Recursion
7. Recursion vs. Iteration

Reference:

1. Midterm Solutions
2. Java Methods

- Introduction
  - Programming modules in Java
    - Methods and classes
  - Divide and conquer

- Methods
  - Invoked by a method call
  - Returns a result to calling method (caller)
    - Similar to boss(caller) asking a worker (called method) to complete a task
  - Allow programmers to modularize programs
  - Manageable program development
  - Software reusability
  - Avoid repeating code
  - Local variables
    - Declared in method declaration
  - Parameters
    - Communicates information between methods via method calls
  - General format of method declaration
    ```
    return-value-type method-name( parameter1, parameter2, ..., parameterN )
    {
      declarations and statements
    }
    
    can also return values -- return expression;
    ```
  - Coercion of arguments
  - Promotion Rules
    - How to convert types without data loss

- Java API packages
  - Classes grouped into categories of related classes
  - Promotes software reuse
  - Import statements specify classes used in Java programs
    - ie. import javax.swing.JApplet;

3. Random-Number Generation

- Java random-number generators
  - Math.random()
    - (int)(Math.random() * 6) - produces integers from 0-5

- Craps Simulation
  - Craps example program
4. Scope

- Craps Simulation
  - Portion of program that can reference an entity by its name
  - Basic scope rules
    - Scope of a parameter declaration
    - Scope of a local-variable declaration
    - Scope of a label in a labeled break or continue statement
    - Scope of a local-variable declaration that appears in the initialization section of a for statement’s header
    - Scope of a method or field of a class

5. Method Overloading

- Several methods of the same name
- Different parameter set for each method
  - Number of parameters
  - Parameter types

6. Recursion

- Recursive method
  - Calls itself (directly or indirectly) through another method
  - Method knows how to solve only a base case
  - Method divides problem
    - Base case
    - Simpler problem
      - Method now divides simpler problem until solvable
  - Recursive call
  - Recursive step
  - Example using recursion: the Fibonacci series

7. Recursion vs iteration

- Iteration
  - Uses repetition structures (for, while or do…while)
  - Repetition through explicitly use of repetition structure
  - Terminates when loop-continuation condition fails
  - Controls repetition by using a counter

- Recursion
  - Uses selection structures (if, if…else or switch)
  - Repetition through repeated method calls
  - Terminates when base case is satisfied
  - Controls repetition by dividing problem into simpler one
  - More overhead than iteration
  - More memory intensive than iteration
  - Can also be solved iteratively
  - Often can be implemented with only a few lines of code
Objectives:

1. Demonstrate knowledge of methods
2. To create and use methods that perform specific operations.

Assignment:
Write a Java program that does the following:

1. Create an integer array with the following values \{1,1,3,5,7,3,29,56,134\}.
2. Use separate methods to analyze this array of values by finding the mean, median, max, min, and mode of the list of numbers formed by the array. Your program should:
   a. Use separate method calls to find each value.
   b. Make the method finding the maximum value in the array recursive.
   c. Print the value.

Your code should be concise and well commented. You are graded on neatness as well as correctness.
Objectives:

3. Demonstrate knowledge of methods
4. To create and use methods that perform specific operations.

Assignment:
Write a Java program that does the following:
3. Create an integer array with the following values {1,1,3,5,7,3,29,56,134}.
4. Use separate methods to analyze this array of values by finding the mean, median, max, min, and mode of the list of numbers formed by the array. Your program should:
   a. Use separate method calls to find each value.
   b. Make the method finding the maximum value in the array recursive.
   c. Print the value.

Your code should be concise and well commented. You are graded on neatness as well as correctness.

```java
class lab2solution {
  double[] array = {1,1,3,5,7,3,29,56,134};
  public static void main(String[] args){
    double mean = getMean();
    double median = getMedian();
    double max = getMax();
    double min = getMin();
    double mode = getMode();
    System.out.println("MEAN: " + mean);
    System.out.println("MEDIAN: " + median);
    System.out.println("MAX: " + max);
    System.out.println("MIN: " + min);
  }
  public double getMean(){
    double sum = 0;
    for(int i=0; i< array.length; i++){
      sum += array[i];
    }
    return sum/array.length;
  }
  public double getMedian(){
    return array[array.length/2];
  }
  public double getMax(){
    int maxOccurrences = 0;
    double mode = array[0];
    for(int i=0; i< array.length; i++){
      ...
    }
  }
  public double getMin(){
    return array[0];
  }
  public double getMode(){
    ...
  }
}
```
double num = array[i];
int occurrences = 0;

for(int j=0; j< array.length; j++){
    if (num == array[j]){
        occurrences++;
    }
}
if( occurrences > maxOccurrences) {
    maxOccurrences = occurrences;
    mode = array[i];
}
}
return mode;
}

public double getMin(){
    double min = array[0];
    for(int i=0; i< array.length; i++){
        if(array[i]<min){
            min=array[i];
        }
    }
}
}

public double getMax(){
    double max = getMax(array[0], array.length()-0);
}

public double getMax( double num, int c){
    while(c>0){
        c++;
        double max = getMax(array[c], array.length-c);
        if(max>num){
            return max;
        }
    }
}
}