CS170 **Lab 9**
Fall 2012 **Single dimensional arrays**

Due: Midnight tonight

Objectives of this lab:

- Write methods utilizing return values and input parameters
- Practice translating English specifications for programs into algorithms and code.
- Practice traversing single dimensional arrays
- Understand command line arguments and how they can be incorporated into a program

Lab preparation:

- Start a terminal application and prepare your lab6 directory:

```
mkdir ~/cs170/lab9
cp ~cs170003/share/lab9/*  ~/cs170/lab9
cd ~/cs170/lab9
ls
```

- You should see a file named `ArrayPractice.java`
- If you do not see these files, ask you TA for help.

Exercises: Command line arguments:

- Open the file `ArrayPractice.java` using gedit:

```
gedit ArrayPractice.java &
```

- Inspect the code in the main method. Note that the program is designed to use command line arguments. The program is designed to be run with an integer command line argument. Examples:

```
java ArrayPractice 5
java ArrayPractice 8
```

- The command line argument specifies how large an array to create. In the first example above, the array should contain 6 elements, and in the second example, the array should contain 8.
- Modify the `main` method to use the command line argument correctly, given the TODO comment in the `main` method.
  - Remember that all command line arguments are strings and are passed to the `main` method via the String array variable `args`.
  - You must convert the command line argument to an integer. Review the `Integer.parseInt` function if you do not remember how to use it.
  - This integer should then be used to make an array of the appropriate size.
  - The code in the `main` method will populate (fill-in) the array with random numbers between 0 and 99 for you.

Exercise 2: Arrays and Methods

- In the file, you will write 3 methods which practice array manipulation:
  - `printReverse`
  - `collapse`
  - `minGap`
- The `printReverse` method

- Inspect the function call to the `printReverse` method in the `main` method and the function header.
- Note that the function is given an array as an input parameter, but the function does not return anything. We can tell because '`void`' is specified as the return type in the function header.
- This function should print out the array in reverse order. This is, the array is

21	76	82	34	17	59

Your function should print:

```
[59, 17, 34, 82, 76, 21]
```

- The `minGap` method:
  - Inspect the function call to the `minGap` method in the `main` method and the function header.
  - Note that the function is given an array as an input parameter, and the function returns an `int.`
  - The "gap" between two elements (values) in an array is defined as absolute value of the 2nd value minus the first.
    - Given the array above, the gaps would be:
      - `55 (a[1] – a[0])`
      - `6 (a[2] – a[1])`
      - `48 (absolute value of a[3]-a[2])`
      - `17 (absolute value of a[4]-a[3])`
      - `42 (a[5] – a[4]`
  - The "minimum gap" of an array is the smallest gap between two adjacent elements. So in the example array above, the minimum gap is 6 (82 – 76).
  - You can use the `Math.abs` method to take the absolute value. This function takes an integer as a parameter and returns the absolute value of that number.
  - Your function should return the minimum gap of the input parameter array.
- The `collapse` method:
  - Inspect the function call to the `collapse` method in the `main` method and the function header.
  - Note that the function is given an array as an input parameter, and the function returns an array of integers`.`
  - To "collapse" an array, we add adjacent elements and put the values into a smaller array which is half the size of our original array.
  - So in the above array above, we would collapse the array by making a new array of length 3. In this array, the values would be:
    - `97 (a[0] + a[1])`
    - `116 (a[2] + a[3])`
    - `76 (a[4] + a[5])`
  - Your function should return this new, "collapsed" array.
  - If the original array has an odd number of values, you should simply ignore the last element in the original array. In other words, you should collapse only n-1 elements of the original

array. So if the original array was:

26	91	42	80	33

Your function should return the array:

117	122

- **Turn-in your work:**
  - Submit your work using the following commands:

```
cd ~/cs170/lab9
/home/cs170xxx/turnin-lab    Scoping.java       lab9
```

  - You will need to replace the 'xxx' in the above commands with the appropriate section number.
  - Make sure you see the “success” message. If you do not, ask your TA for help.