Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ APCS A (Lab Exercises – 4.4)

**Representing Names**

1. Write a class *Name* that stores a person's first, middle, and last names and provides the following methods:

* public Name(String first, String middle, String last)—constructor. The name should be stored in the case given; don't convert to all upper or lower case.
* public String getFirst()—returns the first name
* public String getMiddle()—returns the middle name
* public String getLast()—returns the last name
* public String firstMiddleLast()—returns a string containing the person's full name in order, e.g., "Mary Jane Smith".
* public String lastFirstMiddle()—returns a string containing the person's full name with the last name first followed by a comma, e.g., "Smith, Mary Jane".
* public boolean equals(Name otherName)—returns true if this name is the same as otherName. Comparisons should not be case sensitive. (Hint: There is a String method *equalsIgnoreCase* that is just like the String method *equals* except it does not consider case in doing its comparison.)
* public String initials()—returns the person's initials (a 3-character string). The initials should be all in upper case, regardless of what case the name was entered in. (Hint: Instead of using *charAt*, use the *substring* method of String to get a string containing only the first letter—then you can upcase this one-letter string. See Figure 3.1 in the text for a description of the *substring* method.)
* public int length()—returns the total number of characters in the full name, not including spaces.

2. Now write a program TestNames.java that prompts for and reads in two names from the user (you'll need first, middle, and last for each), creates a Name object for each, and uses the methods of the Name class to do the following:

a. For each name, print

* first-middle-last version
* last-first-middle version
* initials
* length

b. Tell whether or not the names are the same.