# CS 1331 Homework 4

**Due Friday Sept 28, 2012 8:00PM**

## Introduction

This homework will cover string manipulation, more on class design, and some GUI components.

Be sure to name your classes as required by the instructions. Also be sure to use good coding style and indentation, and to use appropriate and descriptive variable names.

Do not forget about the collaboration policies detailed in homework 1, or about javadocing and commenting detailed in homeworks 2 and 3.

## 4.1: Location Class

Create a class called `Location`, which will represent a location and do some useful things for us. It will need:

1. 2 string instance variables to store the city and state. They should have the appropriate visibility / protection.
2. A constructor that takes in 2 strings corresponding to the 2 parts of a location. You may assume that the 2 parameters for the constructor are not null or empty.
3. 2 getters for the instance variables, following standard Java getter/setter naming conventions.
4. The following methods:
    a. `normalize:` changes the location to standard capitalization. For example, if the original location has a city value of “aTlanTA” and a state value of geOrGIa”, after “`normalize`” is called, it should be “Atlanta” and “Georgia”.
    b. `toString:` returns the location as a String, using an abbreviation for the state. For example, if the location has a city value of “Atlanta” and a state value of “Georgia”, then the method should return “Atlanta, GA”. For the purposes of this assignment you may assume the state abbreviation is always the first and last letter of the state, capitalized.
    c. `isSameState:` takes in another `Location` object and determines if the location represented by this location object comes from the same state as the other location. This method should ignore the capitalization of the state (i.e. it should treat “Georgia” and “geORgiA” as the same state).

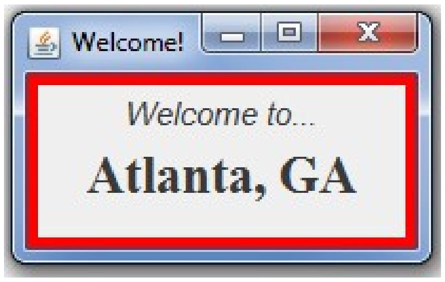
Don’t forget to javadoc the class, constructor, and methods, javadocing parameters where applicable (see the previous homeworks for an examples of this).

## 4.2: Welcome Sign Class

Create a class called `WelcomeSign` that creates a `JFrame` and `JLabels` based upon a `Location` object. It should have:

1. An instance variable to store the `Location`, and an instance variable to store the `JFrame`. They should have the appropriate visibility / protection.
2. A constructor that takes a `Location` parameter for the `WelcomeSign` object to show.
3. In the constructor, create a `JFrame` and store it into the instance variable from step 1. Add `JPanels` and `JLabels` as appropriate to display the location from the `Location` object passed to the constructor. The location displayed should be the result of a `toString` performed on the location. Don't forget to set the default close operation on the `JFrame`.

   An example would be:

4. You can use `Font` and `Border` objects to achieve the effects in the above example:
   ```
   label.setFont(new Font("Arial", Font.ITALIC, 15));
   label2.setFont(new Font("Times New Roman", Font.BOLD, 25));
   panel.setBorder(BorderFactory.createLineBorder(Color.red, 5));
   ```
   See: http://java.sun.com/javase/6/docs/api/java/awt/Font.html
5. Lastly, it should have a method named "`showWelcomeSign`" that will actually show the `JFrame`.

6. Don't forget to javadoc the class, constructor, and methods, javadocing parameters where applicable (see the previous homeworks for an examples of this)

## 4.3: Welcome Sign Generator Class

Finally, create a class named `WelcomeSignGenerator` with a main method that brings together a `Location` object and a `WelcomeSign` object:

1. Prompt the user for the city and the state.
2. Create a `Location` object, using the values collected above.
3. Normalize the location.
4. Create a `WelcomeSign` object.
5. Show the welcome sign.

## Turn-in Procedure

Turn in the following files on T-Square. When you're ready, double-check that you have *submitted* and not just saved as draft.

- Location.java
- WelcomeSign.java
- WelcomeSignGenerator.java

All .java files should have a descriptive javadoc comment.

Don't forget your collaboration statement. You should include a statement with every homework you submit, even if you worked alone.

## Verify the Success of Your HW Turn-In

Practice "safe submission"! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.
2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.
3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.
4. Recompile and test those exact files.
5. This helps guard against a few things.
    a. It helps insure that you turn in the correct files.
    b. It helps you realize if you omit a file or files.**
       (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)
    c. Helps find last minute causes of files not compiling and/or running.

**Note: Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework (past the grace period of 2 am) will not be accepted regardless of excuse. Treat the due date with respect. The real due date and time is 8 pm Friday. Do not wait until the last minute!

# Extra Credit

1. Make the user prompt a dialog box (an actual Swing pop-up).
2. Make the sign font flash colors on a timer.