CS 2334
Project 5: Graphical User Interfaces

November 18, 2015

Due: 1:29 pm on Monday, Dec 7, 2015

Introduction

In project 4, you created a graphical user interface that interacts with a user and displays the requested Mesonet data. This project will extend your experience into the realm of graphics. In particular, you will display min/max/average statistical information spatially by painting information onto the counties of Oklahoma.

Your implementation from the prior projects will serve as important components for this project, with some modification. In addition, we will be releasing some classes within a few days of this assignment being given.

Your final product will:

1. Load in files that describe the set of measures taken (the variables) at the stations, and the set of stations.
2. Allow the user to specify a data file to load.
3. Allow the user to select a variable of interest, and the year(s), month(s) and day(s) of interest.
4. Report the minimum, maximum and average of the selected statistic over the range of years, months and days that have been specified. These statistics will be painted onto a county-by-county map of Oklahoma
5. Upon clicking on a county, your GUI will pop-up a window that describes the county and all of the stations contained within the county.

Learning Objectives

By the end of this project, you should be able to:

1. Represent shape information on a 2D plane.
2. Use shape information for rendering of the shapes.
3. Use shape information to detect when a shape is selected by a mouse click.
4. Compute min/max/average statistics over multiple stations within a county
5. Open pop-up windows that contain data.
6. Continue to exercise good coding practices for Javadoc and for testing

Proper Academic Conduct

This project is to be done in the groups of two that we have assigned. You are to work together to design the data structures and solution, and to implement and test this design. You will turn in a single copy of your solution. Do not look at or discuss solutions with anyone other than the instructor, TAs or your assigned team. Do not copy or look at specific solutions from the net.

Strategies for Success

- The UML is a guide to the new classes and methods that you will implement.
- When you are implementing a class or a method, focus on just what that class/method should be doing. Try your best to put the larger problem out of your mind.
- We encourage you to work closely with your other team member, meeting in person when possible.
- Start this project early. In most cases, it cannot be completed in a day or two.
- Implement and test your project components incrementally. Don't wait until your entire implementation is done to start the testing process.

- Write your documentation as you go. Don't wait until the end of the implementation process to add documentation. It is often a good strategy to write your documentation **before** you begin your implementation.

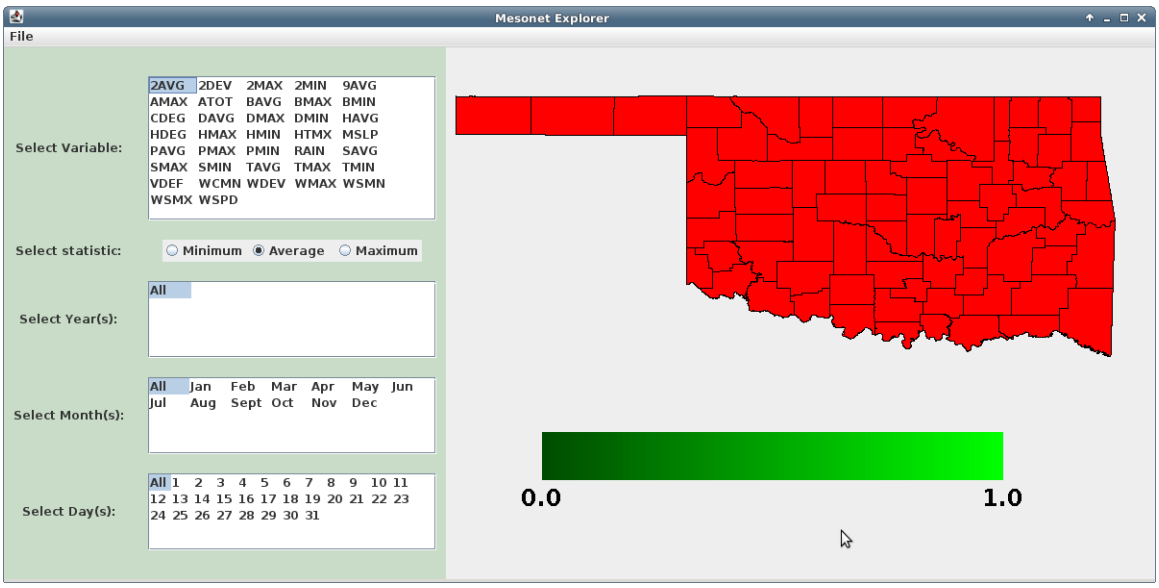
Preparation

- We will be providing some of the classes implementation. However, this will only become available after the final deadline for project 4.
- Import the existing project4 implementation into your eclipse workspace: http://www.cs.ou.edu/~fagg/classes/cs2334/projects/project2/project4-initial.zip

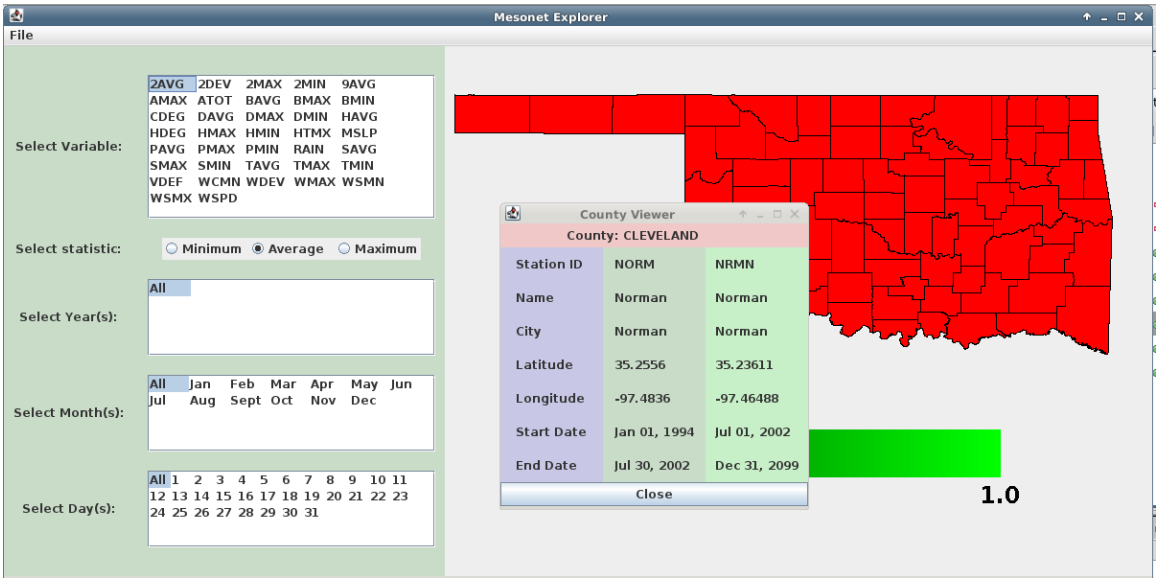
Example Interactions

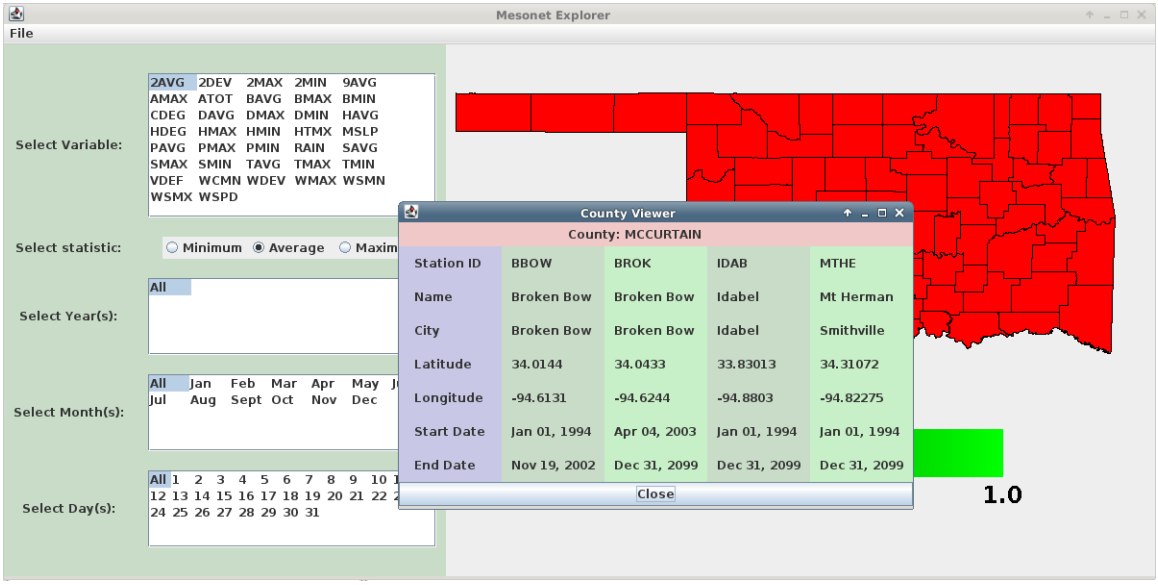
Below is a set of screen-shots for our implementation. Your implementation may have a different look. However, it must have the essential functionality, as described in the next section.

When your program starts up, it will immediately load the station, variable and county configuration files, but will not load a data file. Given the loaded information, here is the initial state of the interface:



When the user clicks on one of the counties, information about the county and the stations within the county are displayed in a separate pop-up window. This window is non-modal and more than one can be opened at once.





Once a data file is selected and opened, the statistical information can be displayed. The user has the ability to select:

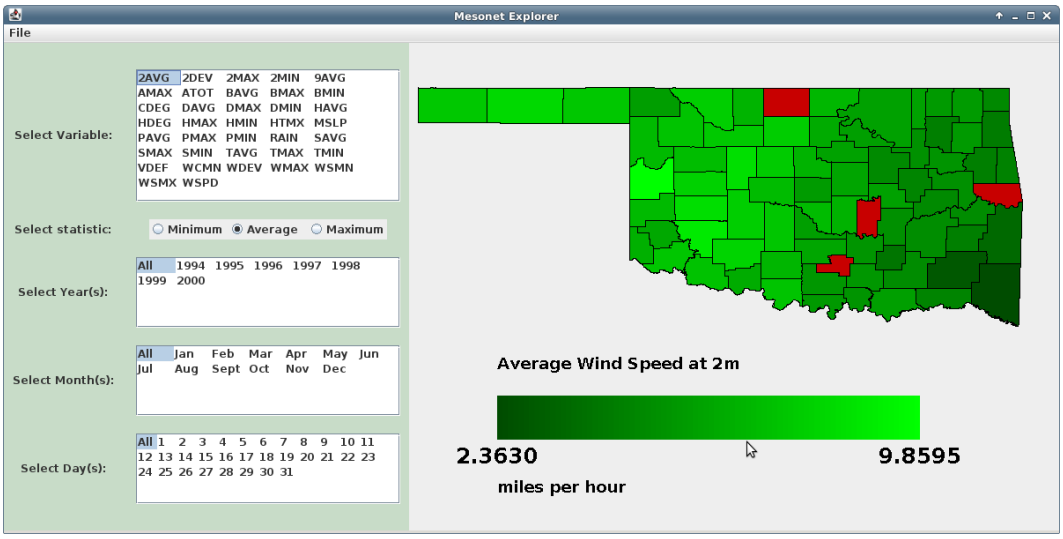
- The measurement of interest (the variableId)

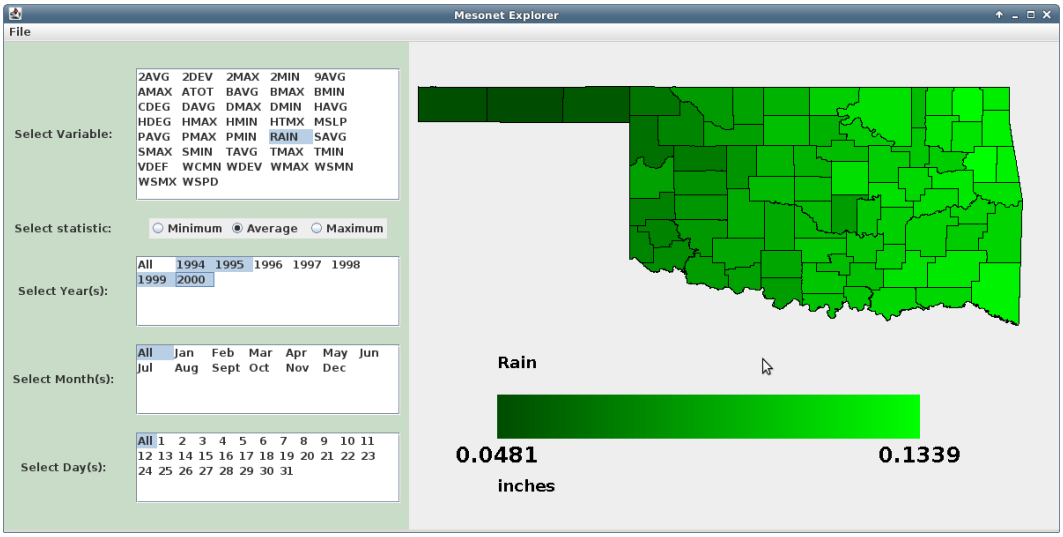
- The statistic of interest (minimum, average or maximum)
- The years of interest. Any combination of years can be selected.
- The months of interest. Any combination of months can be selected.
- The days of interest. Any combination of days can be selected.

When a selection is made, the statistical information is painted onto the state-level representation:

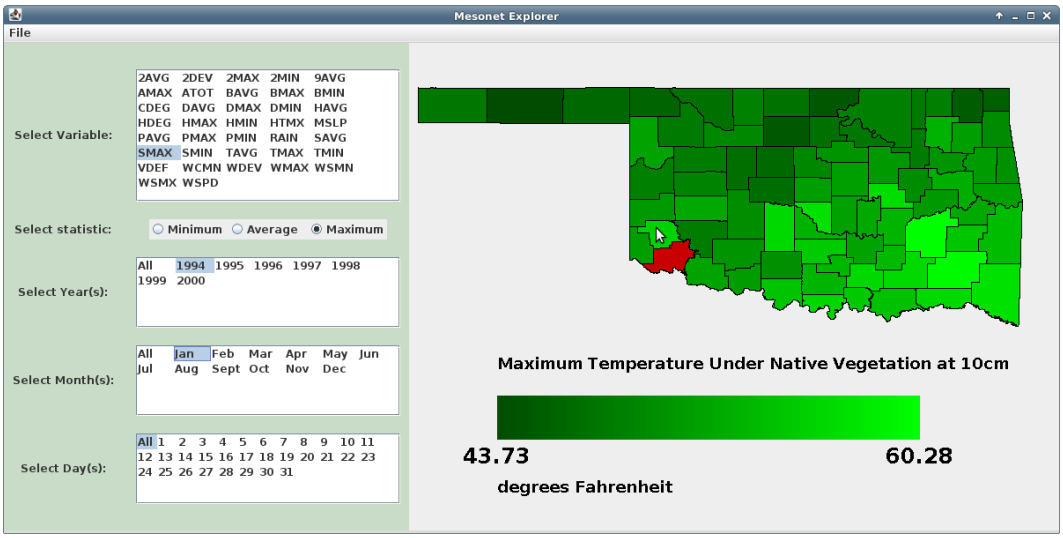
- Each county is represented and painted according to the selected statistic and data. Counties with no or only invalid data are painted with a color that is distinct from that used for valid data.
- A color bar indicates how values are represented using specific colors. The range of values captured by the color bar is determined by the minimum and maximum values for the selected statistic across all of the counties (so, we effectively use our entire color range). The color bar displays these minimum and maximum values.
- Labels that describe the selected statistic and its units.

Subsets of years may be selected, including discontinuous ranges:

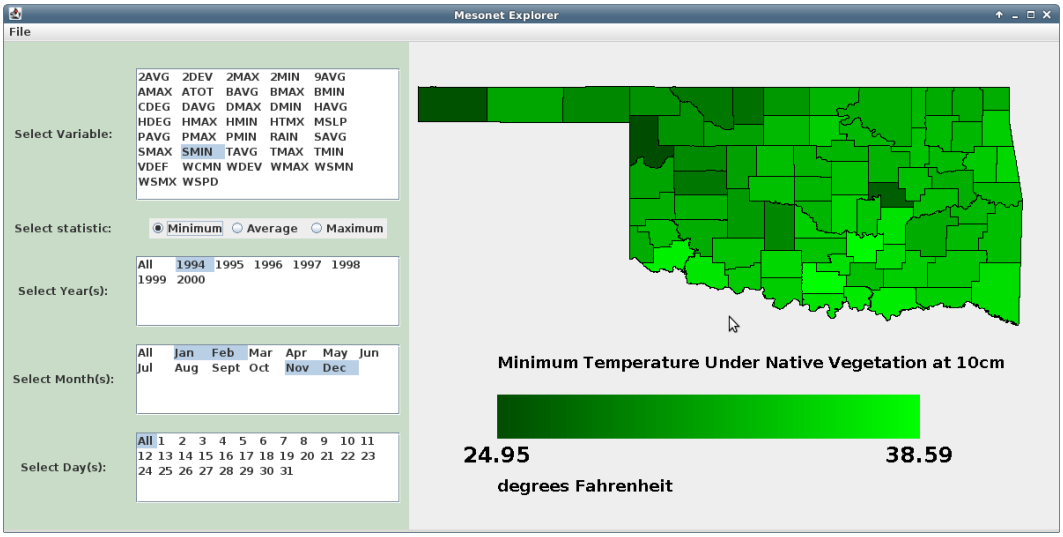


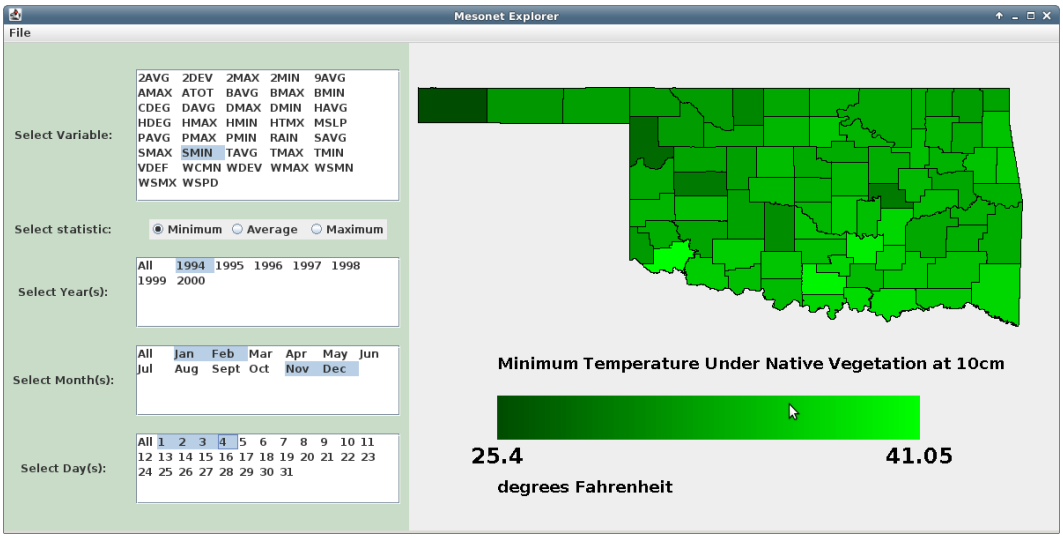


Subsets of months may be selected:

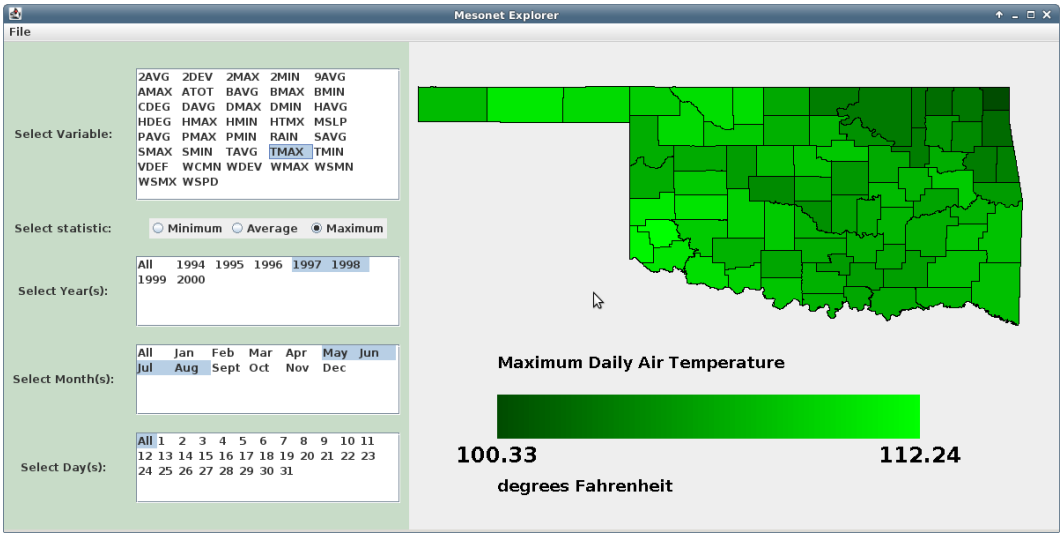


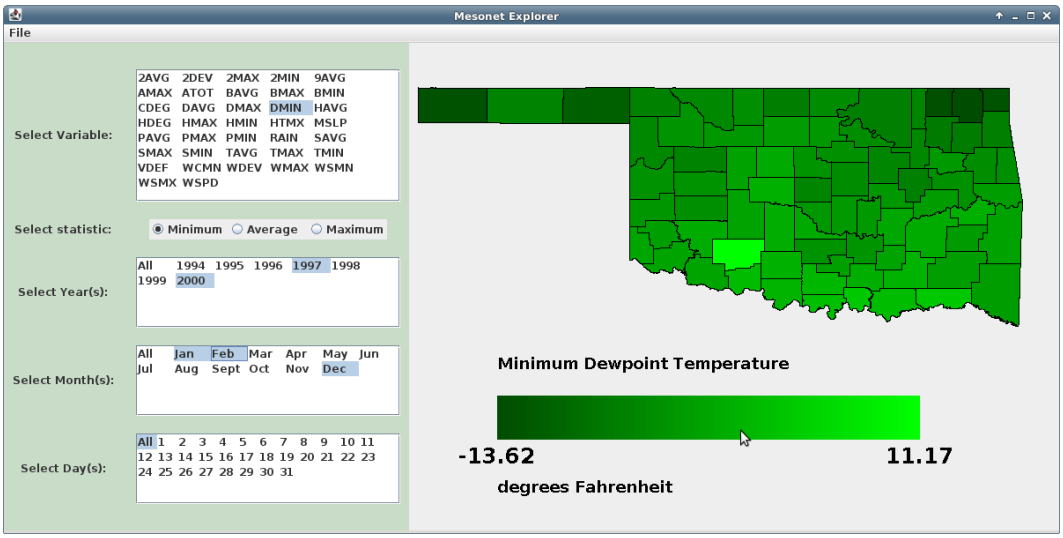
As can days:

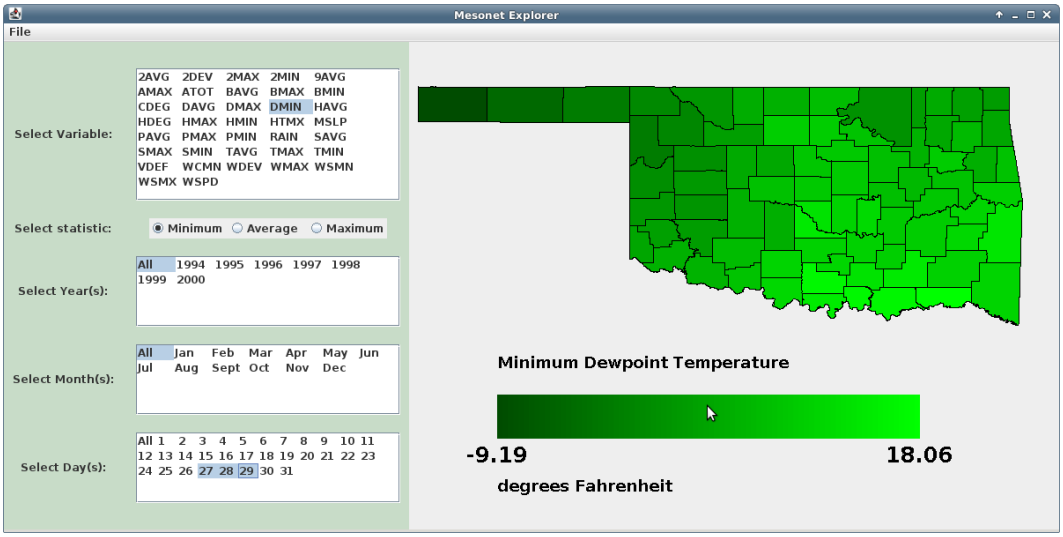

Mesonet Explorer
File
Select Variable:
2AVG 2DEV 2MAX 2MIN 9AVG
AMAX ATOT BAVG BMAX BMIN
CDEG DAVG DMAX DMIN HAVG
HDEG HMAX HMIN HTMX MSLP
PAVG PMAX PMIN RAIN SAVG
SMAX SMIN TAVG TMAX TMIN
VDEF WCMN WDEV WMAX WSMN
WSMX WSPD
Select statistic:
Minimum Average Maximum
Select Year(s):
All 1994 1995 1996 1997 1998
1999 2000
Select Month(s):
All Jan Feb Mar Apr May Jun
Jul Aug Sept Oct Nov Dec
Select Day(s):
All 1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31
Minimum Temperature Under Native Vegetation at 10cm
24.95
38.59
degrees Fahrenheit


Mesonet Explorer
File
Select Variable:
2AVG 2DEV 2MAX 2MIN 9AVG
AMAX ATOT BAVG BMAX BMIN
CDEG DAVG DMAX DMIN HAVG
HDEG HMAX HMIN HTMX MSLP
PAVG PMAX PMIN RAIN SAVG
SMAX SMIN TAVG TMAX TMIN
VDEF WCMN WDEV WMAX WSMN
WSMX WSPD
Select statistic:
Minimum Average Maximum
Select Year(s):
All 1994 1995 1996 1997 1998
1999 2000
Select Month(s):
All Jan Feb Mar Apr May Jun
Jul Aug Sept Oct Nov Dec
Select Day(s):
All 1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31
Minimum Temperature Under Native Vegetation at 10cm
25.4
41.05
degrees Fahrenheit

A few other examples:

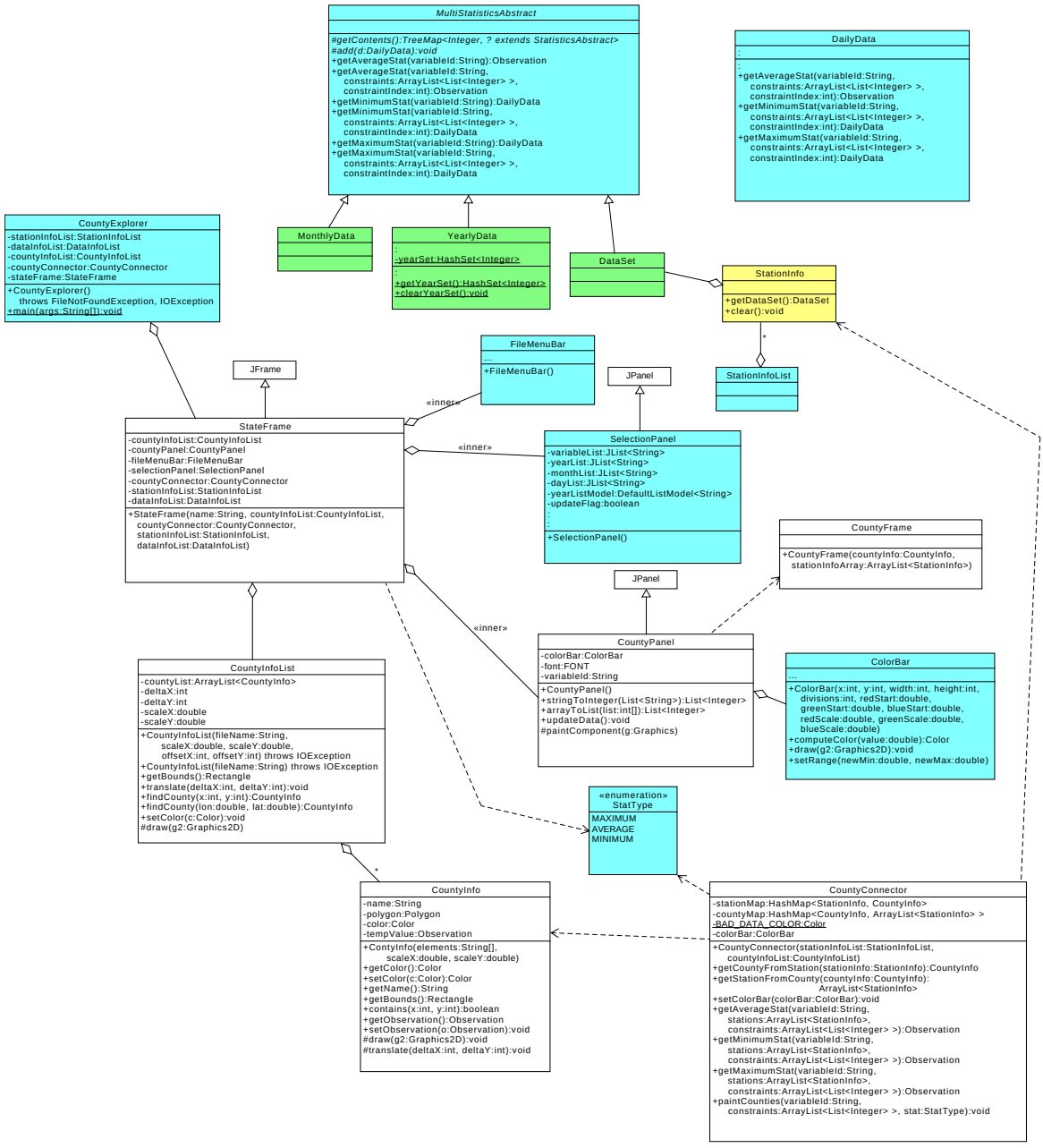





Mesonet Explorer
File
Select Variable:
2AVG 2DEV 2MAX 2MIN 9AVG
AMAX ATOT BAVG BMAX BMIN
CDEG DAVG DMAX DMIN HAVG
HDEG HMAX HMIN HTMX MSLP
PAVG PMAX PMIN RAIN SAVG
SMAX SMIN TAVG TMAX TMIN
VDEF WCMN WDEV WMAX WSMN
WSMX WSPD
Select statistic:
Minimum
Average
Maximum
Select Year(s):
All 1994 1995 1996 1997 1998
1999 2000
Select Month(s):
All Jan Feb Mar Apr May Jun
Jul Aug Sept Oct Nov Dec
Select Day(s):
All 1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31
Minimum Dewpoint Temperature
-9.19
18.06
degrees Fahrenheit

UML Design

Below is an outline of what has changed from project 4:



The colors denote classes that: we provide (*cyan*), you bring forward from project

4 (*green/yellow*) and that you provide the details of (*white*).

Class Design Outline

Your project 4 code will largely stay the same. Your key foci are described below.

The following are required:

- **CountyInfo** describes a single county, including its shape and its rendered color. This class also provides services for drawing the shape of the county and for determining whether a point is contained within the county. Complete the constructor. Complete the *draw()* and *contains()* methods.
- **CountyInfoList** describes all of the counties in Oklahoma. The constructor is responsible for reading the county data from a CSV file, creating the list of counties and handling the transformation between longitude/latitude coordinates and screen coordinates (this transformation involves both a scaling and a translation). Complete the constructor. Complete the *getBounds()*, *translate()*, *draw()*, and *findCounty()* methods.
- **CountyConnector** provides the association between a county and the stations contained in the county. In addition, this class provides the methods for deciding the color with which to paint each county. Complete the implementation of the constructor and the *paintCounties()* method.

The details of the GUI side of the project are left to you. We have provided partial implementations of **StateFrame** and **CountyFrame**. You may start with our implementations or you may implement your own. I you follow our path:

- **StateFrame.CountyPanel** handles the display of the statistical information to the user. Complete the constructor. Also complete the *paintComponent()* and *updateData()* methods.
- **CountyFrame** handles the pop-up window that is generated when someone clicks on a county in the CountyPanel. Complete the implementation of the constructor

Notes

- In project 4, we could select a subset of years by temporarily creating a new DataSet. Here, we can also select a subset of months and days. Rather than creating a new **DataSet** object (with new **YearlyData** and **MonthlyData** objects), we will explicitly tell *getXstat()* which years/months/days to consider in computing the statistic. This is handled in a new method, *StationInfo.getXStat()*, that not only takes as input a *variableId*, but also takes a constraint list (where X is one of Average, Minimum and Maximum). This constraint list is a List of Lists. The first of these lists (element 0) describes which years will be included in the computation; the second (element 1) describes which months and the third describes which days. In turn, this method calls equivalent methods in the **StatisticsAbstract** classes.

- We are assuming a simply linear transformation between longitude/latitude and screen coordinates. We will choose the scale ahead of time (as fixed parameters), but will decide on the offsets after all of the counties have been created.

- The county name and shape information is defined in the *polygons.csv* file. Each row contains data for one county: the name of the county, followed by a sequence of longitude/latitude pairs.

Final Steps

1. Generate Javadoc using Eclipse for all of your classes.

2. Open the *project5/doc/index.html* file using your favorite web browser or Eclipse (double clicking in the package explorer will open the web page). Check to make sure that all of your classes are listed and that all of your documented methods have the necessary documentation.

Submission Instructions

- All required components (source code and compiled documentation) are due at 1:29 pm on Monday, December, 7th (i.e, before class begins).

- Prepare your submission file by creating a project4.zip file. This file must include your entire project, including: src, and doc. **Do not include your data directory**
- Submit your zip file to the project5 folder on D2L.

Grading: Code Review

All groups must attend a code review session in order to receive a grade for your project. The procedure is as follows:

- Submit your project for grading to the D2L Dropbox, as described above.
- Any day following the submission, you may do the code review with the instructor or the TAs. For this, you have two options:
    1. Schedule a 15-minute time slot in which to do the code review. We will use Doodle to schedule these (a link will be posted on D2L). You must attend the code review during your scheduled time. Failure to do so will leave you only with option 2 (no rescheduling of code reviews is permitted).
    2. "Walk-in" during an unscheduled office hour time. However, priority will be given to those needing assistance in the labs and project.
- Both group members must be present for the code review.
- During the code review, we will discuss all aspects of the rubric, including:
    1. The results of the tests that we have executed against your code.
    2. The documentation that has been provided (all three levels of documentation will be examined).
    3. The implementation. Note that both group members must be able to answer questions about the entire solution that the group has produced.
- If you complete your code review before the submission deadline, you have the option of going back to make changes and resubmitting (by the deadline). If you do this, you will need to return for another code review.
- The code review must be completed by Friday, December 11th to receive credit for the project.

References

- The Java API: https://docs.oracle.com/javase/8/docs/api/

Rubric

The project will be graded out of 100 points. The distribution is as follows:

Implementation: 45 points

Program formatting: 10 points

(10) The program is properly formatted (including indentation, curly brace and semicolon locations).

(5) There is one problem with program formatting.

(0) The program is not properly formatted.

Data types and method calls: 10 points

(10) The program is using proper data types and method calls.

(7) There is one error in data type or method call selection.

(4) There are two errors in data type or method call selection.

(0) There are three or more errors in data type and method call selection.

Required Methods: 15 points

(15) All of the required methods are implemented.

(10) One required method is not implemented

(5) Two required methods are not implemented.

(0) Two or more required methods are not implemented.

GUI Design: 10 points

(10) All required GUI elements are included.

(7) One key GUI element is missing

(4) Two key GUI elements are missing.

(0) Three or more key GUI elements are missing.

Proper Execution: 30 points

Output: 15 points

Two (2) points will be deducted for every test that your program fails (note that these are tests that we provide).

Execution: 15 points

(15) The program executes with no errors.

(8) The program executes, but there is one minor error.

(0) The program does not execute.

Documentation and Submission: 25 points

Project Documentation: 4 points

(4) The java file contains all of the required documentation elements at the top of the file.

(3) The java file is missing one of the required documentation elements.

(2) The java file is missing two of the required documentation elements.

(0) The java file is missing more than two of the required documentation elements.

Method-Level Documentation: 9 points

(9) Every method contains all of the required documentation elements ahead of the method prototype.

(6) The method documentation is missing one of the required documentation elements.

(3) The method documentation is missing two of the required documentation elements.

(0) The method documentation is missing more than two of the required documentation elements.

Inline Documentation: 9 points

(9) Every method contains appropriate inline documentation.

(6) There is one missing or incorrect line of inline documentation.

(3) There are two missing or incorrect lines of inline documentation.

(0) There are more than two missing or incorrect lines of inline documentation.

Submission: 3 points

(3) The correct zip file name is used and has the correct contents.

(2) The correct zip file name is used, but one required component is missing.

(0) An incorrect zip file name is used or more than one required component is missing.