CS 2334
Project 5: Graphical User Interfaces

November 16, 2016

Due: 1:29 pm on Monday, Dec 5, 2016

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 small modifications.

Your final product will:

1. Load in files that describe the set of Mesonet stations, the measures taken (the variables) at each stations, and the counties.

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. Allow the user to select one of the minimum, average or maximum statistic to compute.

5. Report 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

6. 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.

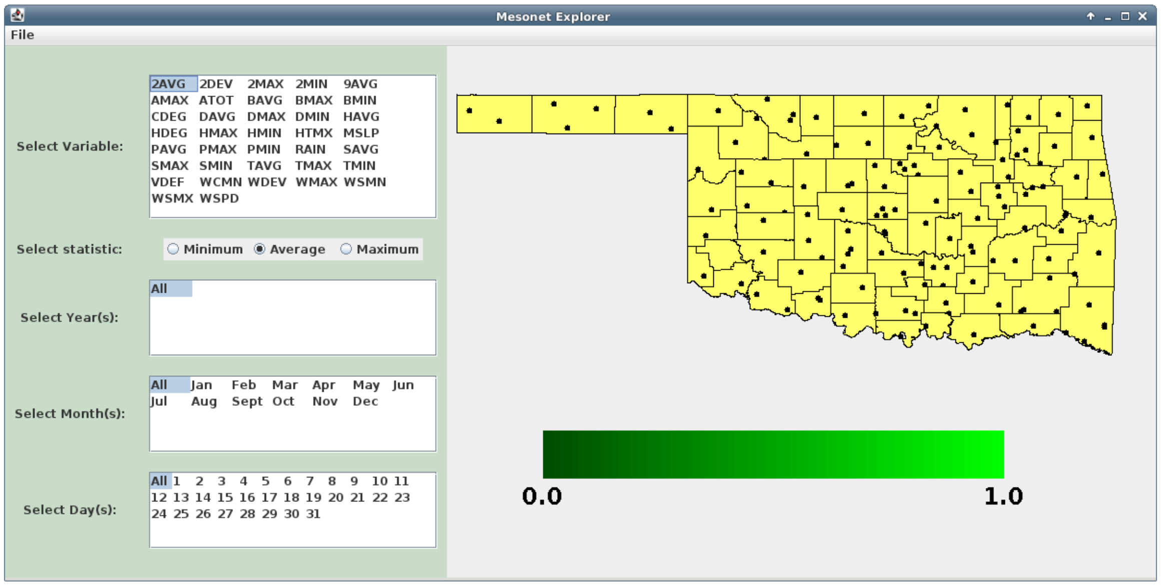
- Implement and test your project components incrementally. Don't wait until your entire implementation is done to start the testing process.
- Write your documentation as you go. Don't wait until the end of the implementation process to add documentation. It is often a good strategy to write your documentation **before** you begin your implementation.

Preparation

- This description and supporting materials are available at: `http://cs.ou.edu/~fagg/classes/cs2334/projects/project5`
- We will be providing parts of our project 4 implementation on Canvas after the project 4 deadline.
- In Eclipse, copy your *project4* folder to a new *project5* project. Within this project, your data should be located in the *data* directory (folder). The data will be the same as for the last project.
- Download project5.zip from the project web site. This zip file contains the following files, which should be copied into your project5 project:
  - **CountyExplorer**: main program
  - **StateFrame**: primary interface. **complete this implementation**
  - **ColorBar**: class for drawing a color bar and for translating scalar values into colors
  - **CountyDefinition**: class for describing the shape of a county and the stations located within it. **complete this implementation**
  - **CountyDefinitionList**: describes all of the counties. **complete this implementation**
  - **CountyFrame**: a class that represents pop-up windows that describe individual counties
  - **MultiStatisticsWithDaysAbstract**: an abstract class that extends **MultiStatisticsAbstract** and now is the parent class for **DataSet**, **DataYear** and **DataMonth**
  - **StatType**: an enumerated data type
  - **countyShapes.csv**: a file containing a list of coordinates that represent the boundary of each county

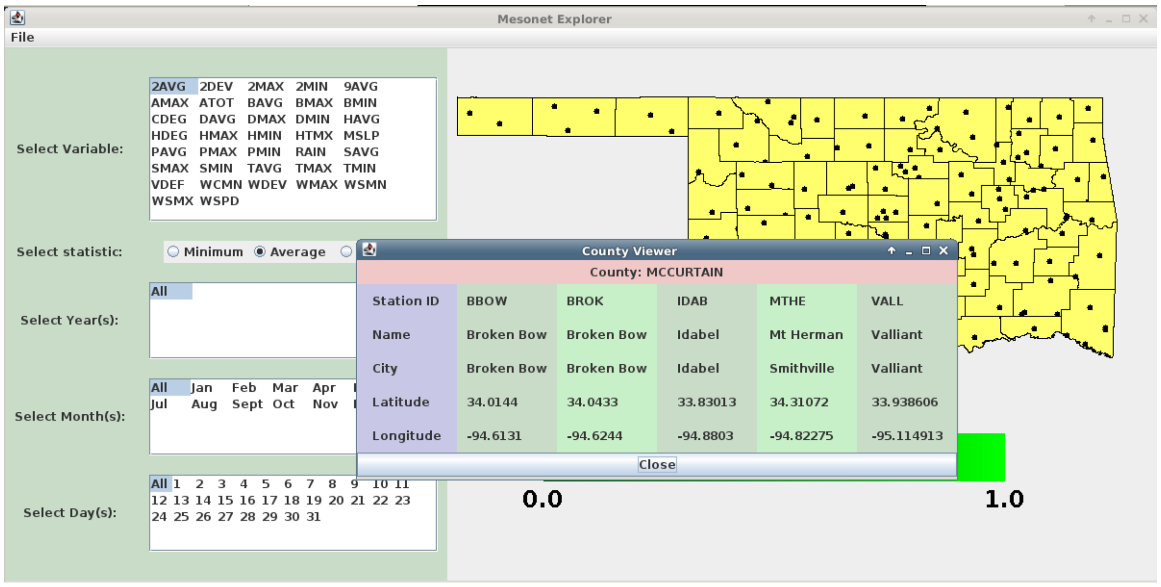
Example Interactions

Below is a set of screen-shots for our implementation. Your implementation will look very similar. 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:



Note that the county outlines are drawn, as well as the station locations (indicated with small circles).

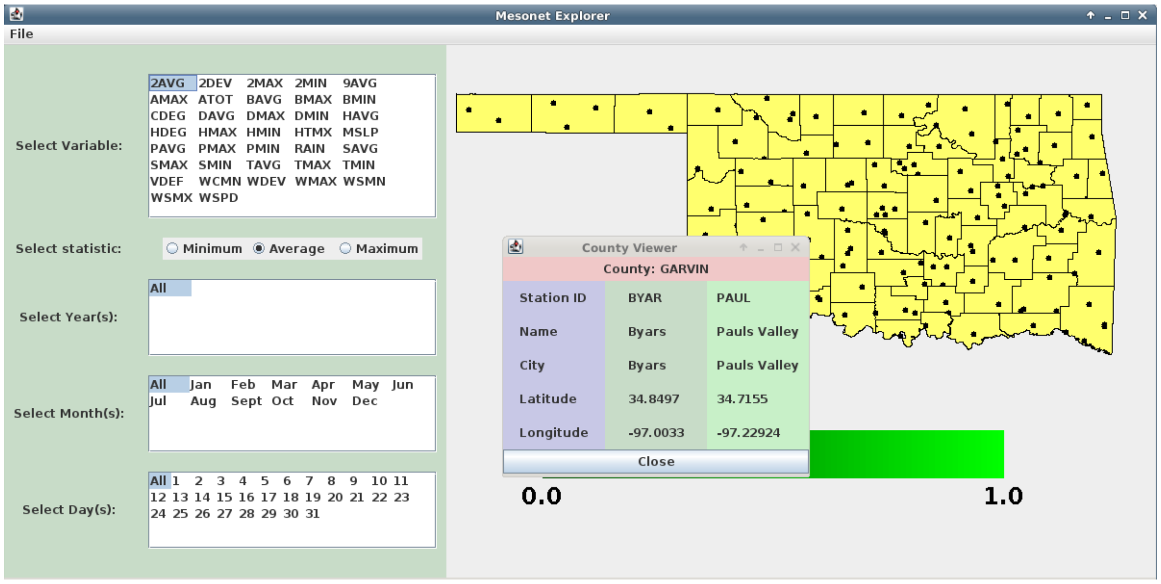
When the user clicks on one of the counties, information about the county and the stations within the county are displayed in a separate pop-up window. This window is non-modal and more than one can be opened at once.



Select statistic: Minimum [✓ Average]

County: MCCURTAIN

Station ID	BBOW	BROK	IDAB	MTHE	VALL
Name	Broken Bow	Broken Bow	Idabel	Mt Herman	Valliant
City	Broken Bow	Broken Bow	Idabel	Smithville	Valliant
Latitude	34.0144	34.0433	33.83013	34.31072	33.938606
Longitude	-94.6131	-94.6244	-94.8803	-94.82275	-95.114913



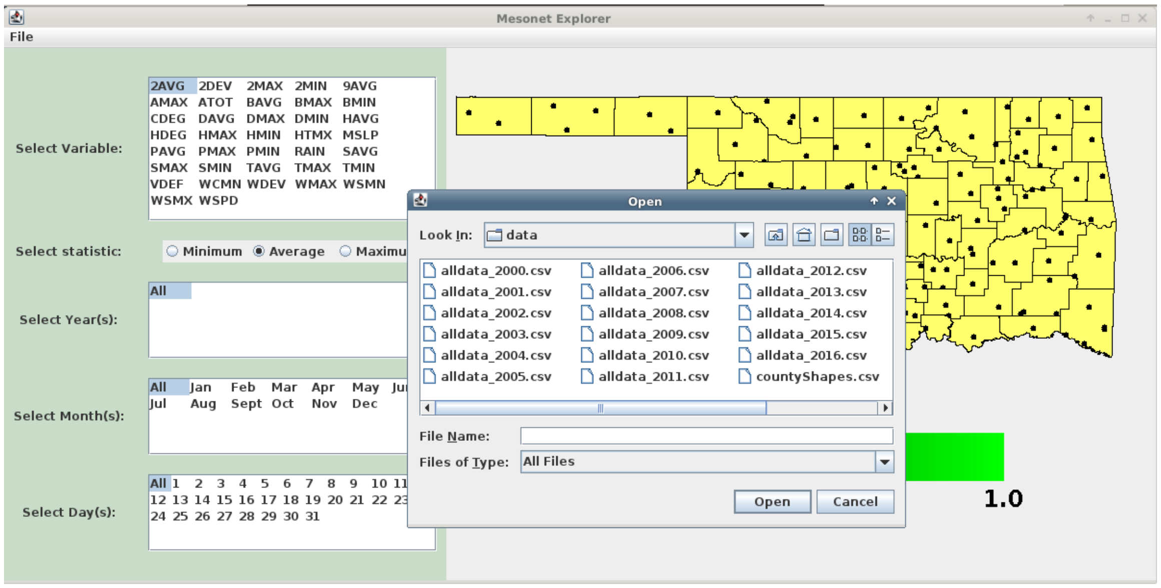
Select statistic: Minimum [✓ Average] Maximum

County: GARVIN

Station ID	BYAR	PAUL
Name	Byars	Pauls Valley
City	Byars	Pauls Valley
Latitude	34.8497	34.7155
Longitude	-97.0033	-97.22924

Note that if the user clicks outside of the state, then no pop-up windows are generated.

When the user selects **File/Open text file**, the interface brings up a pop-up file chooser:



Minimum [✓ Average] Maximu

Look In: data

alldata_2000.csv alldata_2006.csv alldata_2012.csv
alldata_2001.csv alldata_2007.csv alldata_2013.csv
alldata_2002.csv alldata_2008.csv alldata_2014.csv
alldata_2003.csv alldata_2009.csv alldata_2015.csv
alldata_2004.csv alldata_2010.csv alldata_2016.csv
alldata_2005.csv alldata_2011.csv countyShapes.csv

File Name:

Files of Type: All Files

If a valid file is selected, then it is loaded into the internal data structure. If the file is not valid in some way, then an appropriate dialog box explaining this fact to the user must be opened.

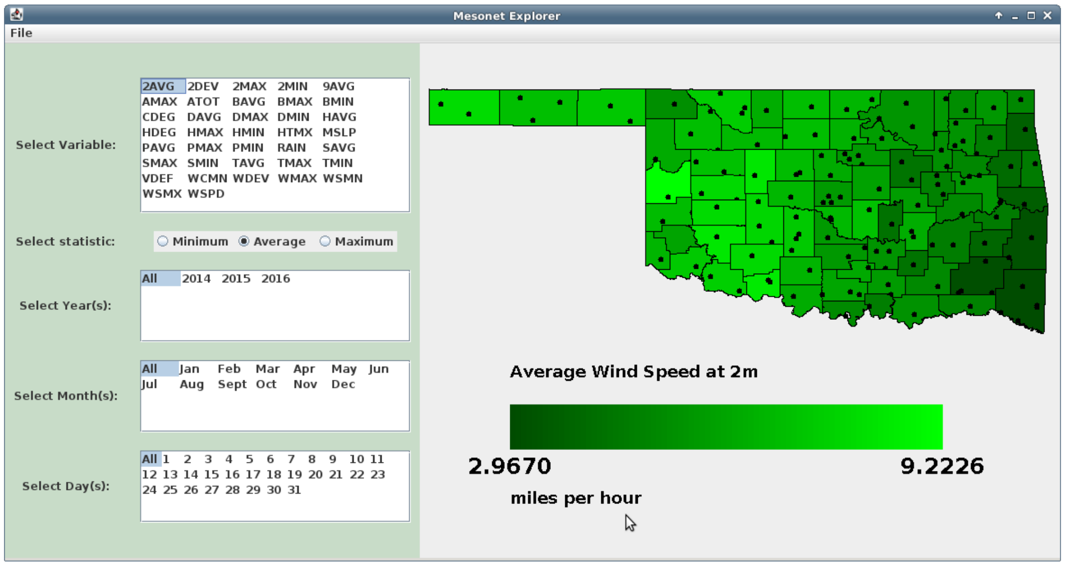
Once a data file is selected and opened, the statistical information can be displayed. The user has the ability to select:

- 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.

With all years, months and days selected, the interface looks like this:



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] 2014 2015 2016

Select Month(s): [✓ All] Jan Feb Mar Apr May Jun Jul Aug Sept Oct Nov Dec

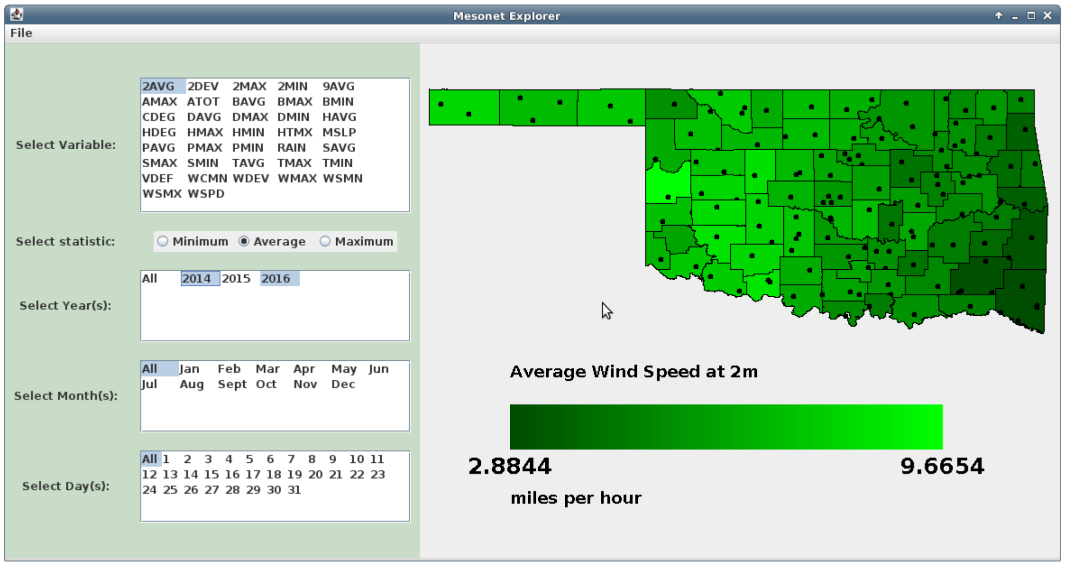
Select Day(s): [✓ All] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Average Wind Speed at 2m

2.9670 9.2226

miles per hour

Subsets of years may be selected, including discontinuous ranges:



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 [✓ 2014] 2015 [✓ 2016]

Select Month(s): [✓ All] Jan Feb Mar Apr May Jun Jul Aug Sept Oct Nov Dec

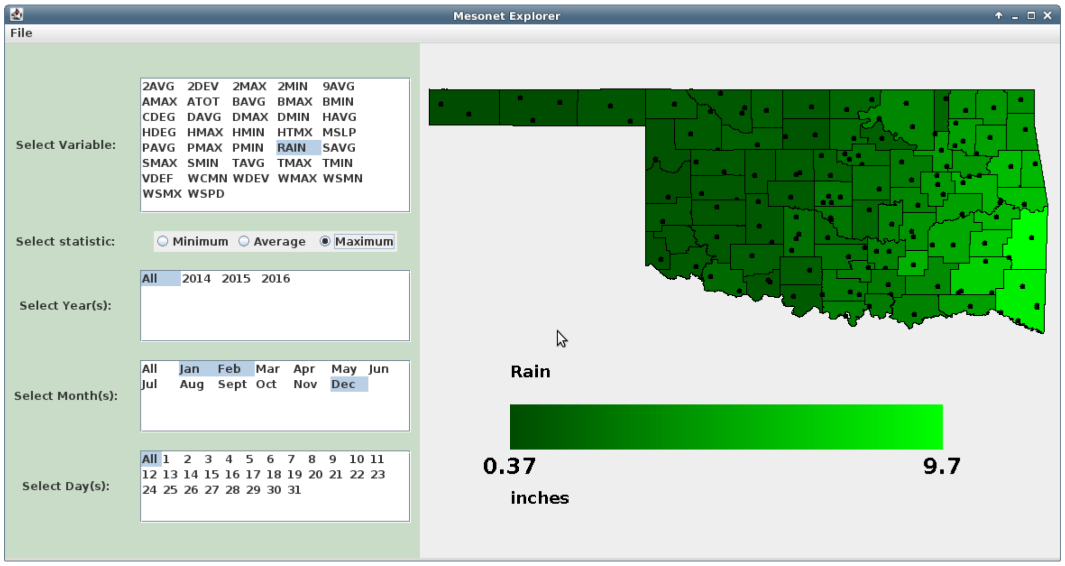
Select Day(s): [✓ All] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Average Wind Speed at 2m

2.8844 9.6654

miles per hour

Subsets of months may be selected:



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] 2014 2015 2016

Select Month(s): All [✓ Jan] [✓ Feb] Mar Apr May Jun Jul Aug Sept Oct Nov [✓ Dec]

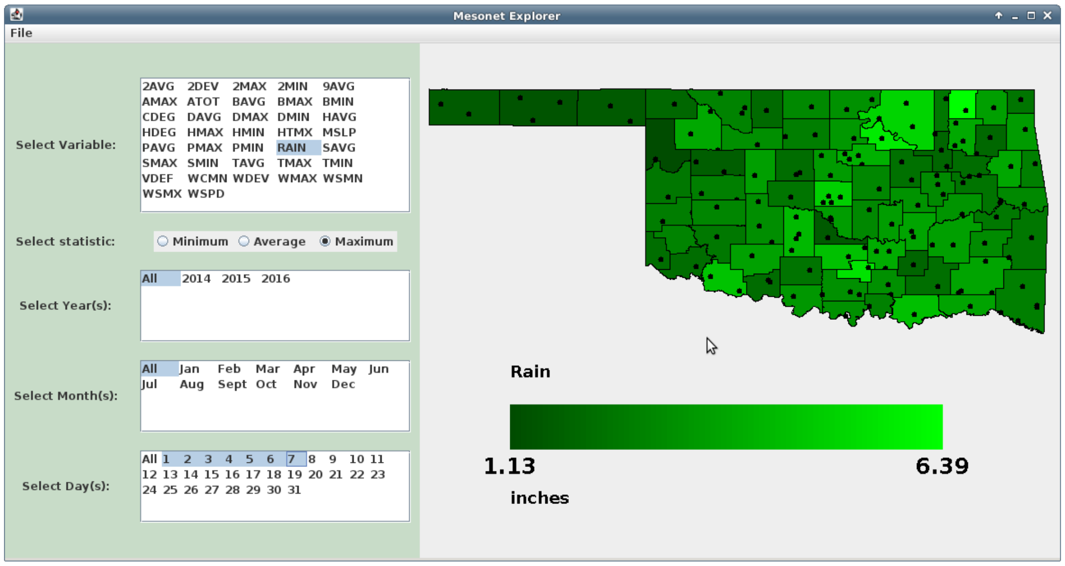
Select Day(s): [✓ All] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Rain

0.37 9.7

inches

As can days:



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] 2014 2015 2016

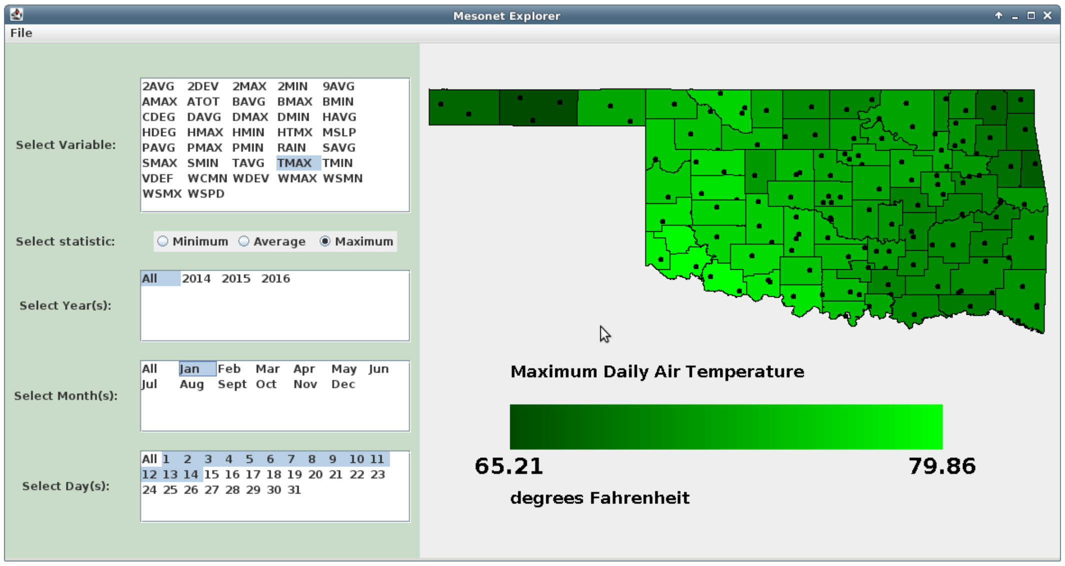
Select Month(s): [✓ All] Jan Feb Mar Apr May Jun Jul Aug Sept Oct Nov Dec

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

Rain

1.13 6.39

inches



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] 2014 2015 2016

Select Month(s): All [✓ Jan] Feb Mar Apr May Jun Jul Aug Sept Oct Nov Dec

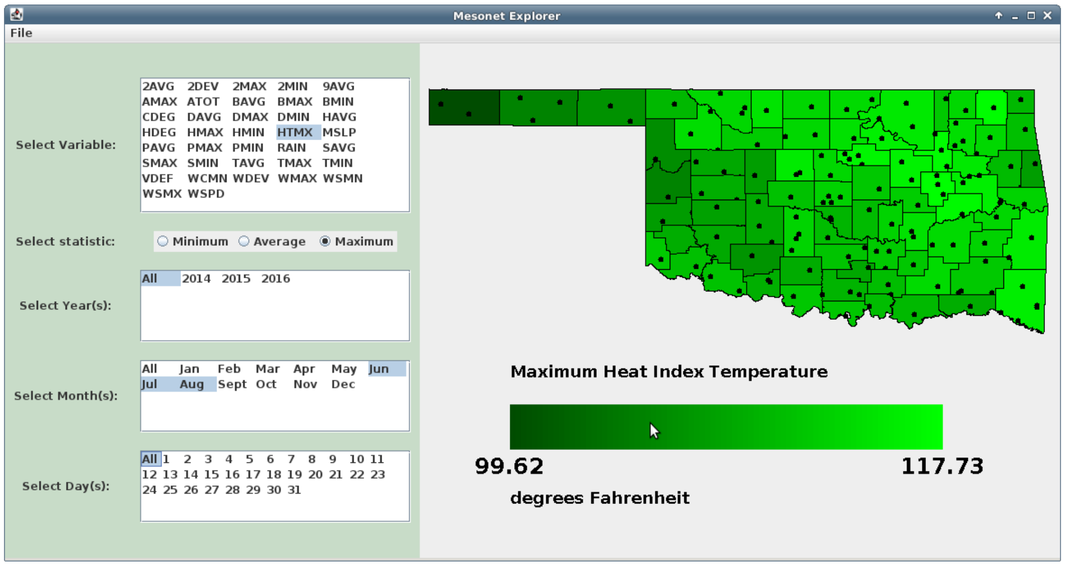
Select Day(s): All [✓ 1] [✓ 2] [✓ 3] [✓ 4] [✓ 5] [✓ 6] [✓ 7] [✓ 8] [✓ 9] [✓ 10] [✓ 11] [✓ 12] [✓ 13] [✓ 14] 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Maximum Daily Air Temperature

65.21 79.86

degrees Fahrenheit

A few other examples:



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] 2014 2015 2016

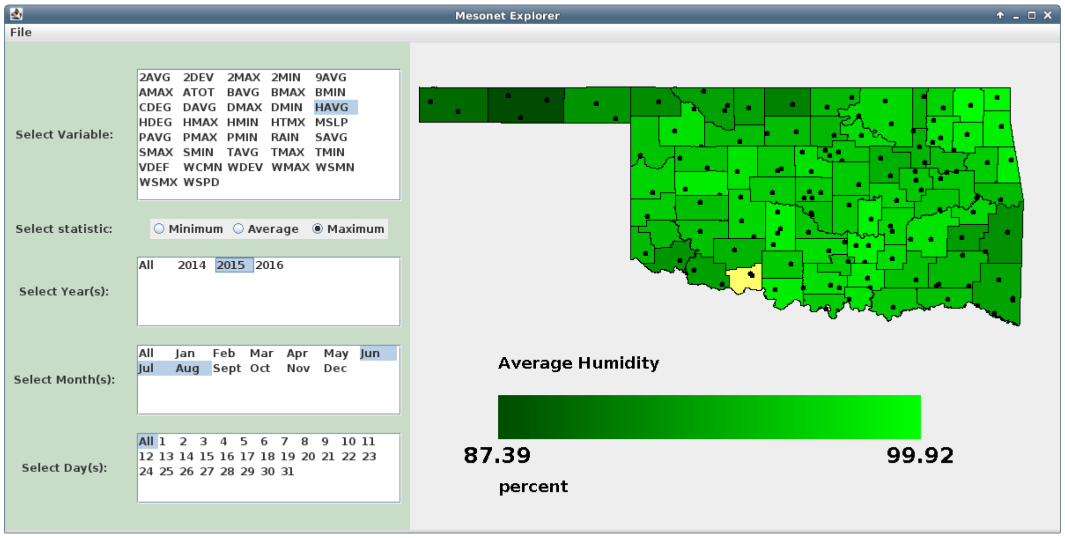
Select Month(s): All Jan Feb Mar Apr May [✓ Jun] [✓ Jul] [✓ Aug] Sept Oct Nov Dec

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

Maximum Heat Index Temperature

99.62 117.73

degrees Fahrenheit



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 2014 [✓ 2015] 2016

Select Month(s): All Jan Feb Mar Apr May [✓ Jun] [✓ Jul] [✓ Aug] Sept Oct Nov Dec

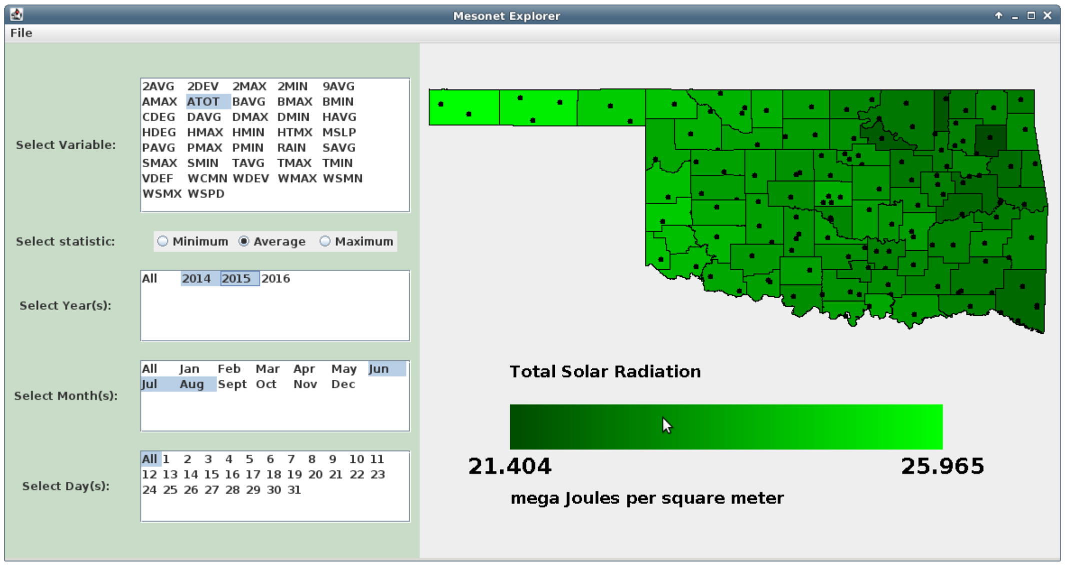
Select Day(s): [✓ All] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Average Humidity

87.39 99.92

percent

Note: the county painted in the non-color bar color has no valid data for the selected days and variableId.


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 [✓ 2014] [✓ 2015] 2016

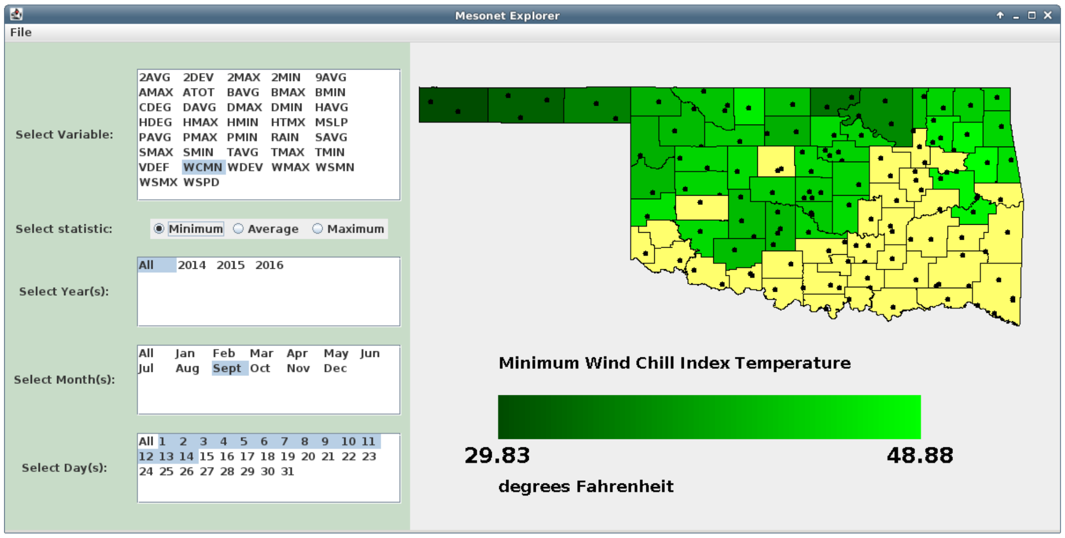
Select Month(s): All Jan Feb Mar Apr May [✓ Jun] [✓ Jul] [✓ Aug] Sept Oct Nov Dec

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

Total Solar Radiation

21.404 25.965

mega Joules per square meter


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] 2014 2015 2016

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

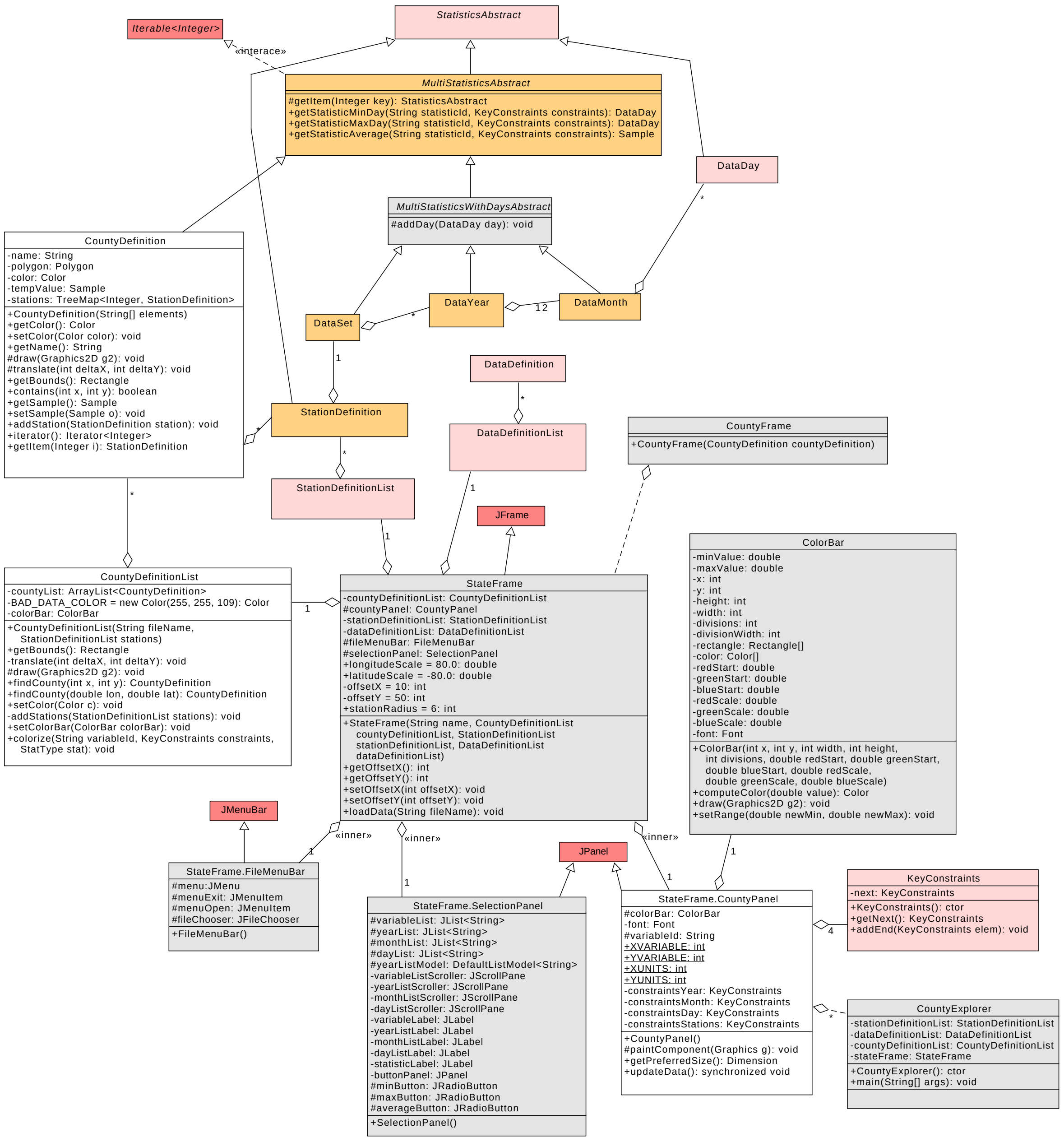
Minimum Wind Chill Index Temperature

29.83 48.88

degrees Fahrenheit

UML Design

Below is the UML diagram that emphasizes the classes that have changed since project 4.



Iterable<Integer>
«interace»
StatisticsAbstract
MultiStatisticsAbstract
#getItem(Integer key): StatisticsAbstract
+getStatisticMinDay(String statisticId, KeyConstraints constraints): DataDay
+getStatisticMaxDay(String statisticId, KeyConstraints constraints): DataDay
+getStatisticAverage(String statisticId, KeyConstraints constraints): Sample
DataDay
MultiStatisticsWithDaysAbstract
#addDay(DataDay day): void
CountyDefinition
-name: String
-polygon: Polygon
-color: Color
-tempValue: Sample
-stations: TreeMap<Integer, StationDefinition>
+CountyDefinition(String[] elements)
+getColor(): Color
+setColor(Color color): void
+getName(): String
#draw(Graphics2D g2): void
#translate(int deltaX, int deltaY): void
+getBounds(): Rectangle
+contains(int x, int y): boolean
+getSample(): Sample
+setSample(Sample o): void
+addStation(StationDefinition station): void
+iterator(): Iterator<Integer>
+getItem(Integer i): StationDefinition
DataSet
DataYear
DataMonth
12
*
1
DataDefinition
StationDefinition
DataDefinitionList
CountyFrame
+CountyFrame(CountyDefinition countyDefinition)
StationDefinitionList
JFrame
CountyDefinitionList
-countyList: ArrayList<CountyDefinition>
-BAD_DATA_COLOR = new Color(255, 255, 109): Color
-colorBar: ColorBar
+CountyDefinitionList(String fileName, StationDefinitionList stations)
+getBounds(): Rectangle
-translate(int deltaX, int deltaY): void
#draw(Graphics2D g2): void
+findCounty(int x, int y): CountyDefinition
+findCounty(double lon, double lat): CountyDefinition
+setColor(Color c): void
-addStations(StationDefinitionList stations): void
+setColorBar(ColorBar colorBar): void
+colorize(String variableId, KeyConstraints constraints, StatType stat): void
StateFrame
-countyDefinitionList: CountyDefinitionList
#countyPanel: CountyPanel
-stationDefinitionList: StationDefinitionList
-dataDefinitionList: DataDefinitionList
#fileMenuBar: FileMenuBar
#selectionPanel: SelectionPanel
+longitudeScale = 80.0: double
+latitudeScale = -80.0: double
-offsetX = 10: int
-offsetY = 50: int
+stationRadius = 6: int
+StateFrame(String name, CountyDefinitionList countyDefinitionList, StationDefinitionList stationDefinitionList, DataDefinitionList dataDefinitionList)
+getOffsetX(): int
+getOffsetY(): int
+setOffsetX(int offsetX): void
+setOffsetY(int offsetY): void
+loadData(String fileName): void
ColorBar
-minValue: double
-maxValue: double
-x: int
-y: int
-height: int
-width: int
-divisions: int
-divisionWidth: int
-rectangle: Rectangle[]
-color: Color[]
-redStart: double
-greenStart: double
-blueStart: double
-redScale: double
-greenScale: double
-blueScale: double
-font: Font
+ColorBar(int x, int y, int width, int height, int divisions, double redStart, double greenStart, double blueStart, double redScale, double greenScale, double blueScale)
+computeColor(double value): Color
+draw(Graphics2D g2): void
+setRange(double newMin, double newMax): void
JMenuBar
«inner»
«inner»
«inner»
JPanel
StateFrame.FileMenuBar
#menu:JMenu
#menuExit: JMenuItem
#menuOpen: JMenuItem
#fileChooser: JFileChooser
+FileMenuBar()
StateFrame.SelectionPanel
#variableList: JList<String>
#yearList: JList<String>
#monthList: JList<String>
#dayList: JList<String>
#yearListModel: DefaultListModel<String>
-variableListScroller: JScrollPane
-yearListScroller: JScrollPane
-monthListScroller: JScrollPane
-dayListScroller: JScrollPane
-variableLabel: JLabel
-yearListLabel: JLabel
-monthListLabel: JLabel
-dayListLabel: JLabel
-statisticLabel: JLabel
-buttonPanel: JPanel
#minButton: JRadioButton
#maxButton: JRadioButton
#averageButton: JRadioButton
+SelectionPanel()
StateFrame.CountyPanel
#colorBar: ColorBar
-font: Font
#variableId: String
+XVARIABLE: int
+YVARIABLE: int
+XUNITS: int
+YUNITS: int
-constraintsYear: KeyConstraints
-constraintsMonth: KeyConstraints
-constraintsDay: KeyConstraints
-constraintsStations: KeyConstraints
+CountyPanel()
#paintComponent(Graphics g): void
+getPreferredSize(): Dimension
+updateData(): synchronized void
KeyConstraints
-next: KeyConstraints
+KeyConstraints(): ctor
+getNext(): KeyConstraints
+addEnd(KeyConstraints elem): void
4
CountyExplorer
-stationDefinitionList: StationDefinitionList
-dataDefinitionList: DataDefinitionList
-countyDefinitionList: CountyDefinitionList
-stateFrame: StateFrame
+CountyExplorer(): ctor
+main(String[] args): void

The colors denote classes that: we provide (*gray*), you bring forward from project 4 with no change (*pink*), you bring forward from project 4 with minor changes (*orange*), we provide skeletons of and you must fill some of the details (*white*), and the Java API provides (*red*).

Class Design Outline

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

- **MultiStatisticsAbstract** in project 4 is being split into **MultiStatisticsAbstract** and **MultiStatisticsWithDaysAbstract** classes. The details of this split are shown in the UML. Modify your classes accordingly.
- **StationDefinition**: add a method with the prototype:

  ```
      public void draw(Graphics2D g2)
  ```

  This method draws a circle at the station's longitude/latitude. Note that the scale and offset are provided by the **StateFrame** class.

  In addition, **StationDefinition** must now extend **StatisticsAbstract**.
- **CountyDefinition** describes a single county, including its shape, the stations that are located in the county 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()*, *contains()*, *getBounds()* and *addStation()* methods.
- **CountyDefinitionList** 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()*, *addStations()*, *colorize()* and *findCounty()* methods.
- The **StateFrame.CountyPanel** class is responsible for displaying the state of Oklahoma. Complete the *mouseClicked()* method. Select the parameters for the Color Bar. Complete the *paintComponent()* and *updateData()* methods.
- Revisit your implementation of **KeyConstraints**. Implement an appropriate set of unit tests.

- Provide unit tests for **CountyDefinition**.
- Provide unit tests to confirm that **StationDefinition** produces the correct statistic values given a provided set of constraints.

Notes

- In project 4, we selected a subset of years using **KeyConstraints**. Here, because we are querying a county for its statistic value, we must provide a linked list of four different constraints in the following order: stations, years, months and days.
- We are assuming a simple linear transformation between longitude/latitude and screen coordinates. We have chosen the scale already (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 *countyShapes.csv* file. Each row contains data for one county: the name of the county, followed by a sequence of longitude/latitude pairs. The number of coordinates will vary from one county to the next.
- The Java API provides a range of useful classes. We strongly suggest that you examine the APIs for the **Shape**, **Polygon** and the **Graphics2D** classes.

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 (five primary classes plus four JUnit test classes) and that all of your documented methods have the necessary documentation.

Submission Instructions

- All required components (source code and compiled documentation) are due at 1:29:00 pm on Monday, December 5th (i.e, before class begins)
- Submit your project to Web-Cat using one of the two procedures documented in the Lab 2 specification.

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 Web-Cat server.
- Any time following the submission, you may do the code review with the instructor or one of 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 Canvas). 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). Note that schedule code review time **may not** be used for help with a lab or a project
    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 deadline, you have the option of going back to make changes and resubmitting (by the deadline). If you do this, you may need to return for another code review, as determined by the grader conducting the current code review

- The code review must be completed by Friday, December 9th to receive credit for the project

References

- The Java API: `https://docs.oracle.com/javase/8/docs/api/`
- Choosing accessible colors:
  `http://www.somersault1824.com/tips-for-designing-scientific-figures-for-color-blind-readers/`

Rubric

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

Correctness/Testing: 45 points

The Web-Cat server will grade this automatically upon submission. Your code will be compiled against our set of tests. These unit tests will not be visible to you, but the Web-Cat server will inform you as to how many tests your code passed/failed. This grade component is proportional to the fraction of tests that your code passes (so 22.5 points means that your code passed half of the tests).

Style/Coding: 20 points

The Web-Cat server will grade this automatically upon submission. Every violation of the *Program Formatting* standard described in Lab 1 will result in a subtraction of a small number of points (usually two points). Looking at your submission report on the Web-Cat server, you will be able to see a notation for each violation that describes the nature of the problem and the number of subtracted points.

Design/Readability: 35 points

This element will be assessed by a grader during the code review. Any *errors* in your program will be noted in the code stored on the Web-Cat server, and two points will be deducted for each. Possible errors include:

- Non-descriptive or inappropriate project- or method-level documentation
- Missing or inappropriate inline documentation
- Inappropriate choice of variable or method names
- Inefficient implementation of an algorithm
- Incorrect implementation of an algorithm
- Incomplete coverage of your Unit Tests. We expect that your unit tests will test all lines of your code

If you do not submit compiled Javadoc for your project, 5 points will be deducted from this part of your score.

Note that the grader may also give *warnings* or other feedback. Although no points will be deducted, the issues should be addressed in future submissions(where points may be deducted).

Bonus: up to 5 points

You will earn one bonus point for every twelve hours that your assignment is submitted early.

Penalties: up to 100 points

You will lose five points for every twelve hours that your assignment is submitted late (up to 48 hours). Submissions will not be accepted more than 48 hours after the deadline.