CSCI 242
Programming Project 5
Star Charts and Constellations

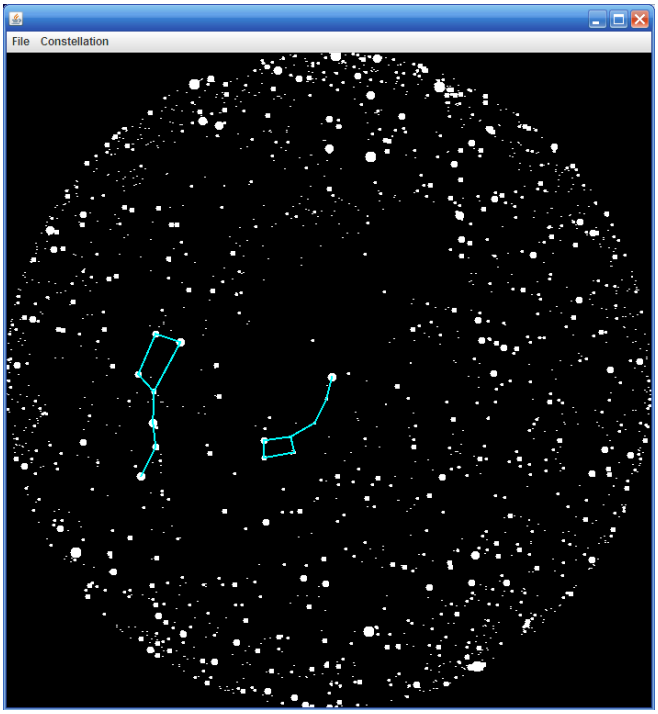
We had the sky up there, and we used to lay on our backs and look up at them, and discuss whether they was made or just happened.
— Mark Twain

This project is similar to one used at the University of Toronto.

Points: 50

Due Date: Wednesday, November 30
This is a difficult project. Start early and work throughout the allotted time.

The goal of this project is to create a star chart like the one below, that displays the night sky and lets the user choose constellations to display.



Along the way you will gain experience with a number of aspects of programming that we have talked about in class, including:

- Inheritance,
- Reading text files,
- Working with `Collections`,
- Graphical user interfaces, and
- Graphics.

Data Files for this Project

This project depends on a number of data files. They are all found in /home/student/classes/Cs242/StarrySky. If you are working in the lab, please do not make copies of the files, but hardcode the filenames to the course directory above. If you are working on your home PC, make copies, but again change the pathname to point to the lab copy before you submit your program. **You will lose points on this project if your submitted program is not reading the copy in the course directory.**

The files are:

stars.txt

This file contains information about stars, one line per star. Astronomers collect lots of data about stars and there are many catalogs that identify the locations of stars. Most of the fields are separated by whitespace, but the final field is semicolon delimited.

The first three fields are the x, y and z coordinates for the star. We will only use the x and y coordinates when plotting the star. Each axis in the coordinate system goes from -1 to +1, and the centre point is (0,0). Stars with negative z values are in the Southern hemisphere and should not be displayed.

The fourth field is the Henry Draper number, which is a unique integer identifier for the star.

The fifth field is a double representing the magnitude (or brightness) of the star.

The sixth field is Harvard Revised number, another integer identifier.

The seventh field exists only for a small number of stars and is a semicolon-separated list of names for a star. A star may have several names.

Two unique identifiers appear in the data because the star data has been collected from different sources, and the catalogs have several different ways to uniquely identify stars.

Constellation files - BigDipper.txt, Bootes.txt, Cassiopeia.txt, Cygnus.txt, Gemini.txt, Hydra.txt, UrsaMajor.txt, UrsaMinor.txt

Each of these files contains a description of a constellation. Each line contains the names of two stars, separated by a comma. Note that there may be spaces within the names of the stars.

Detailed Directions

This project consists of developing two major classes and a number of helper classes. Because this project is longer and more difficult than most, we are giving you more explicit directions on how to develop each.

StarPanel

The `StarPanel` class will be where the stars and constellations are displayed. It should inherit from `JPanel`.

To create the `StarPanel` class:

In the Projects window, right click on the Source package and choose `New – JPanel` form.

In the Design view, right click on the form and choose Properties.

Set the following properties:

```
minimumSize      [32767, 32767]
```

`preferredSize [32767, 32767]`

This will let the star panel grow and shrink with its containing window.

Star Panel Data

Switch to the Source view.

Declare the following private variables.

A list to hold all the stars, e.g. `List<Star> stars;`

A map to hold the named stars. The key is the name, the value is the star, e.g. `Map<String, Star> namedStars;`

A map to hold the constellations. The key is the constellation name. The value is a list of pairs of strings, each being a star name e.g. `Map<String, Constellation> constellations;`

A list of visible constellations, e.g. `List<Constellation> visibleConstellations;`

Your instructor strongly suggests using generics for all of these data structures. You are, of course, free to declare any other variables you think might be useful.

The Star class

One of the features of Java that is particularly useful with GUIs is inner classes.

Inside the `StarPanel` class declare a private class named `Star`.

Place private data members in the class containing all the information needed for a star (see the file description, above). Hint: The names of the stars can be stored in an array of Strings.

`readStars()`

Back inside of the `StarPanel` class (but outside of the `Star` class), write a method that iteratively reads a star from `stars.txt`, creates a `Star` object containing the data, and adds the object to the list of stars.

This sounds easy, but actually can be a bit tricky. Reading in the first six fields of each star is straightforward. After reading in the Harvard Revised number, however, you will have zero or more names of the star, delimited by semicolons. Perhaps the easiest way to read the last field is to read it all at once using `nextLine()`. Trim the resulting `String` to remove leading and trailing whitespace, then split the `String` using the pattern: `";\\s*"`. This creates an array of `String`s.

Add each star with a name should be added to `namedStars`, as well.

Test and debug this method before moving on. Your instructor has supplied a small file named `stars.short.txt`. It contains three stars, one with no name, one with one name and one with multiple names. Make certain you are reading the data for each correctly.

Modify `StarPanel`'s constructor so that the star file is read when the object is created.

`@Override`
`public void paintComponent(Graphics g)`

Override `paintComponent()` for `StarPanel` to draw the stars.

Each star is to appear as a circle, the brighter the star, the larger the circle. We need to do three calculations before we can draw the star. Write a private method for each:

`xToPixel()` and **`yToPixel()`**
The coordinate system used for pixels in a picture has position (0,0) in the upper left corner of the picture, and the maximum x and y values are the height and width of the picture in pixels. These methods should covert from the (x,y) coordinate system used by the star objects to the appropriate pixel coordinates. For example:
(-1.0, 1.0) in star coordinates becomes (0,0) in pixel coordinates.
(0.0, 0.0) in star coordinates becomes (0.5*width, 0.5*height) in star coordinates.
(0.5, 0.5) in star coordinates becomes (0.75*width, 0.25*height) in star coordinates.
It is up to you to work out the math. See your instructor if you get stuck. Note that you can get the current width and height of the `StarPanel` by calling `getWidth()` and `getHeight()`.

`findRadius()`
The radius of the star depends on its magnitude. By trial and error, your instructor found that

```
radius = max(1, 6.0-magitude)
```

gives fairly nicely sized stars.

`paintComponent()` requires you to iterate across your list of stars. If the star's z value is greater than or equal to zero, you should render it by calling `g.fillOval()`. `fillOval()` takes four parameters, the first two are the upper left corner of the oval. The next two are the width and height for the star. Your line of code will look something like:

```
g.fillOval(pixelX-radius, pixelY-radius, 2*radius, 2*radius);
```

Note that we subtract the radius from the pixel values to move it from the center of the star to the upper left. We also double the radius to get the diameter.

If the star's z value is negative, it appears in the Southern hemisphere and should not be rendered.

StarryFrame Class
Add a frame class to your project by right clicking on the source package icon in the Projects window and choosing
New – JFrame form
Name the class `StarFrame`.

In the Design view, right click on the Frame and choose Properties – Set Layout – Border Layout. This lets the StarPanel fill the entire frame.

Drag a `StarPanel` to the `StarFrame`.

Right click on the frame's editor window and choose Run – File.
A window should appear that displays the stars. Note that your code, as written, will display the stars as black circles on a white background.

Test and debug your program until this part of it works. As with reading the stars file, you will find it useful to test with a file that contains just a few stars.

Constellation
Declare a class named `Constellation`. It should contain a list of pairs of star names.

As discussed above, there are several text files that describe constellations. Each file contains two star names per line of the file, separated by a comma. Each pair of star names represents one line to be drawn for the constellation. The goal of this section is to draw lines between the stars for any constellation.

`private void readConstellation(String constellationName)`
Add the `readConstellation()` method to the `StarPanel` class. `readConstellation` should take the name of the constellation as a parameter, add `".txt"` to the name and open the appropriate file.

An easy way to read a constellation file is to read an entire line, trim off the whitespace, and then split the remaining `String` using the pattern, `",\\s+"`. This gives us an array of two `String`s, which can be added to the list of star pairs for this constellation.

After reading an entire constellation, add the `Constellation` object to the map of constellations.

Test this method thoroughly before proceeding to try to draw the constellations.

`private void readAllConstellations()`
Add a method to the StarPanel class that reads in all the Constellation files. It may be easiest to hardcode the constellations names into an array, and then loop through the array, calling `readConstellation()` for each. Obviously, each constellation should be added to the `constellations` map.

Modify `StarPanel`'s constructor so that all the constellations are read when the panel is created.

Modifying `paintComponent()`
Return to your `paintComponent()` method and modify it as follows:
After you have completed drawing the stars add a loop that walks through the list of currently visible constellations. For each constellation in the list, take each star pair, look up each star in the map of named stars, convert the x and y values to pixels and draw a line between them. The actual line of code to draw the line will look something like:

```
g.drawLine(star1PixelX, star1PixelY, star2PixelX, star2PixelY);
```

It is probably worthwhile to use one or more private helper methods for this.

Again, test this method before proceeding. The best way to test it, is to add one constellation, say UrsaMinor (the little dipper), to the visible constellations and run the `StarFrame` class. Lines should be drawn showing the little dipper.

`public void toggleConstellation(String constellationName)`
Add a public method to the `StarPanel` class named `toggleConstellation()`. If the constellation is not in the list of visible constellations, add it to the list. Otherwise, remove it from the list. Hint: remember that Java's `List`s have a `contains()` method.

Again, test this method before proceeding.

Adding a Menu to the StarFrame Class
Return to the Design view of the `StarFrame` class.
Add a Swing Menu Bar to the frame.
Right click on the Edit menu and change the text to Constellation.
Add the following Menu Items.
Add an Exit menu item to the File menu. Code its ActionEvent to call `System.exit(0);`
Add a menu item for each Constellation to the Constellation menu. Each ActionEvent should call `toggleConstellation()` with the appropriate name. Thus, selecting UrsaMinor the first time will make the constellation visible. Selecting it a second time will make it invisible.

Test all the menu options until you are certain they work.

Getting help from your Instructor
As always, your instructor is happy to help you when you are stuck. There are seven places in this write up where he suggests you test your code. If you come to him for help, he will ask you where you are stuck, and then ask you to demonstrate that you have tested all the methods up to that point.

Grading Considerations
Your program must compile and draw stars to receive a passing grade. Drawing constellations and having a working menu system are required to receive full credit. Partially completed programs should provide tests to show what portions are working.

Good software engineering is expected. Use NetBeans **Source-Format** to make certain your code is appropriately indented. Write as many comments as you thing appropriate, then double the number.

Extras for Experts
No extra credit for this, just the fun of learning.
Modify the `paintComponent()` method so that you are drawing white stars on a black background. Further modify it so that constellations are drawn in a different color.