**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*

*The contemplation of celestial things will make a man both speak and think more sublimely and magnificently when he descends to human affairs.*

*— Cicero*

**Star Catalogs and Coordinate Systems**

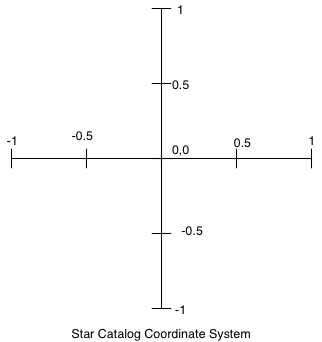
Astronomers collect lots of data about stars and there are many catalogs that identify the locations of stars. In this assignment, you will use data from a **star catalog** to create a picture that plots the locations of stars.

Since a real data set often has some incorrect data and the occasional field missing, a cleaned-up catalog has been prepared for your use in this assignment. The provided file **"stars.txt"** contains one line for each star that is represented in the catalog. The meaning of each field (column) is described below.

* The first three fields are the X, Y and Z coordinates for the star. We will ignore the Z-coordinate and use only the X and Y coordinates. Each axis in the coordinate system goes from -1 to +1, and the center point is <0, 0>. (See the figures below.)
* The fourth field is the **Henry Draper** number, which is simply a unique identifier for the star.
* The fifth field is the **magnitude** (or brightness) of the star.
* The sixth field is the **Harvard Revised** number, another 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. The fields that you will need for this assignment include the X and Y coordinates, the magnitude, the Henry Draper number, and the name (or names) of each star.

The StdDraw library is used to produce the visuals in this project and has a coordinate system with pixel position <0, 0> in the bottom-left corner, and the maximum X and Y values are the height and width of the picture in pixels (all pictures will be square). See below for a comparison of the two coordinate systems.

 A close up of a logo

Description automatically generated

StdDraw coordinate system

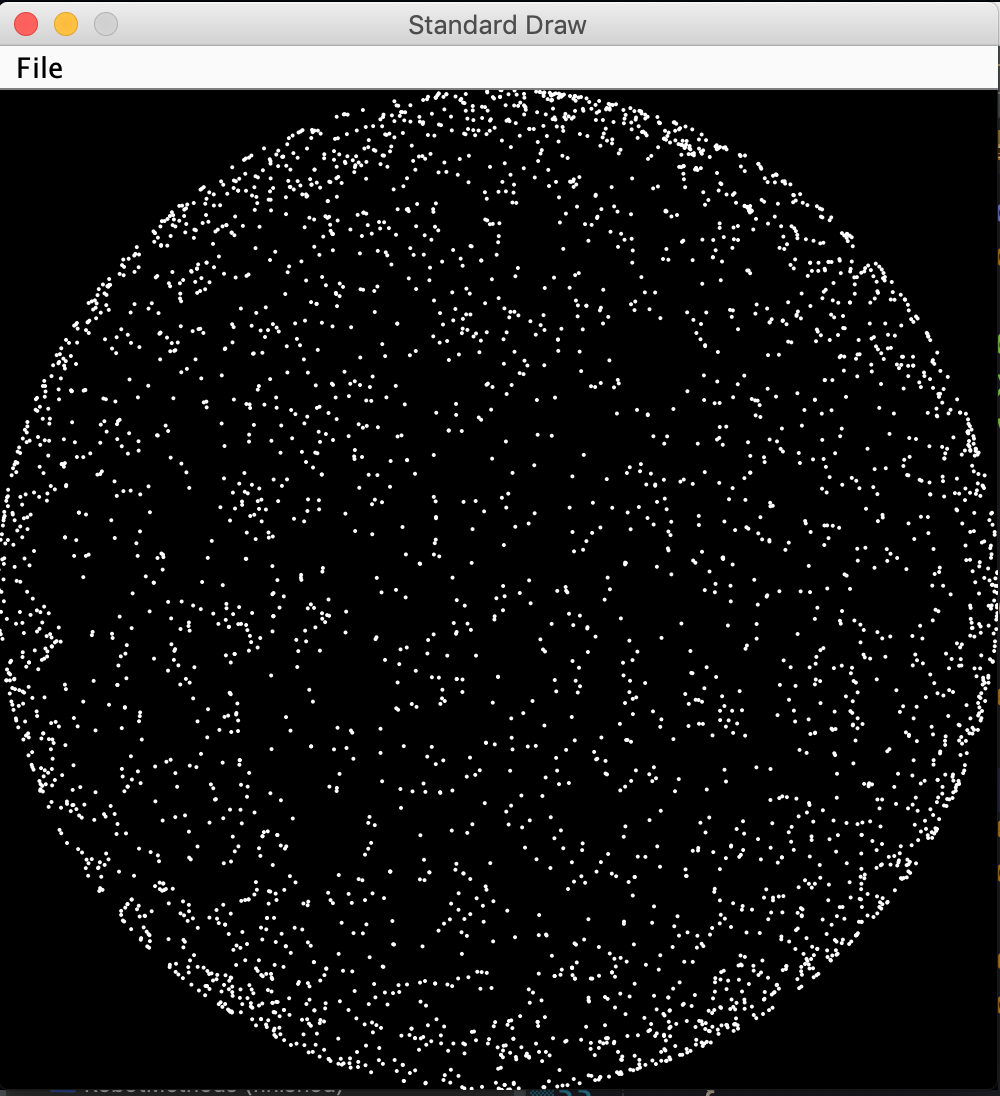
**Part 1: Creating the Data Types and Drawing Stars**

1. Create a class **Star.java**. This type is an abstraction of a single star and should have fields for the required properties (listed previously).
2. Create a class **StarChart.java**. This class will load the star information from file and plot the results.
   1. StarChart's constructor should accept a String fileName representing the catalog file containing the star data. This method should load the stars into an ArrayList<Star>. You can test this method by printing some of the contents of the list and comparing with a neighbor or the input file. Feel free to glare accusingly at either if it doesn't match your output.
      1. **Reminder:** as stated previously, a star can have no name or some number of semi-colon separated names. Your code must handle either case. Scanner's nextLine method will return the remainder of the line (everything up to the next line break).
3. To create a picture of the stars, we need to translate from the coordinate system used in the star catalog (picture above left) to the computer graphics coordinate system (above right).

Create a static method coordsToPixel(origX, origY, size). Given the X and Y coordinates of a star and the size in pixels of the picture, return the X and Y location\* of the star in terms of pixels on screen. **Hint:** choose a couple locations and convert them to StdDraw's coordinate system, then scale based on the supplied screen size.

* 1. \*Create a tiny data type **Point.java** to bundle the on-screen X and Y coordinates into an object. The coordsToPixel method should return a Point object.
  2. Test this method thoroughly before proceeding. For example, if the size of the picture is 1000 by 1000, then the <0, 0> point in the star catalog is at pixel <500, 500>; <0.5, 0.5> from the catalog is at pixel <750, 750>. You should use coordsToPixel as a helper function in the remainder of the assignment.

1. Create a class **Runner.java** with a main method. Add the code below to main to dimension the drawing window, where SIZE is a static final int variable in Runner (a class constant).

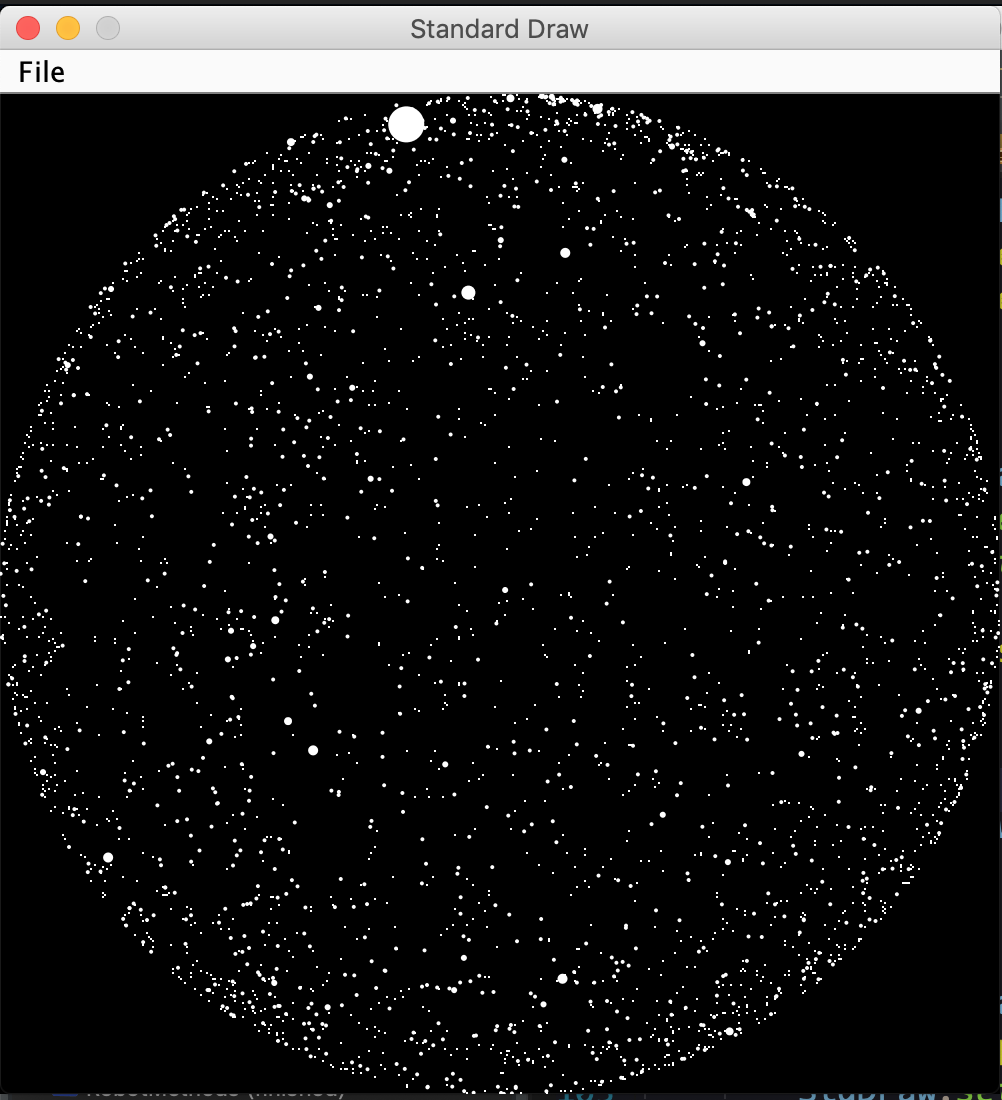


StdDraw.setCanvasSize(SIZE, SIZE);

StdDraw.setXscale(0, SIZE);

StdDraw.setYscale(0, SIZE);

1. In StarChart, write a method void drawStars() that will draw the stars to screen.
   1. Fill the drawing canvas with black.
   2. Set the current pen color to white.
   3. Loop the stars list and draw a filled circle with a 2-pixel radius at the proper on-screen location.
   4. Test the drawStars method from Runner; you should something like the above.
   5. Brighter stars have smaller magnitude values. Add the code to draw a star based on how bright it should be. To calculate the radius of the star, use the following formula:

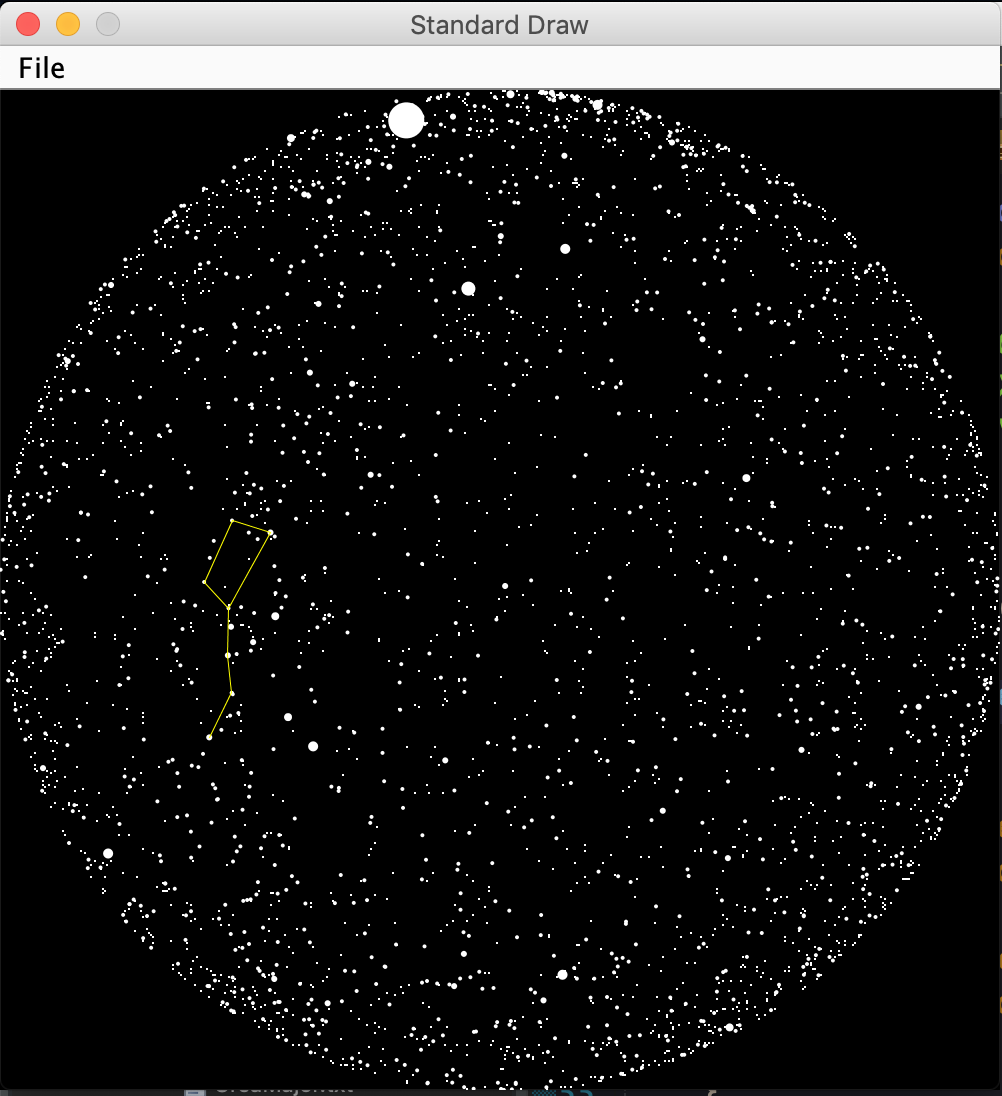


size = round(10.0 / (magnitude + 2))

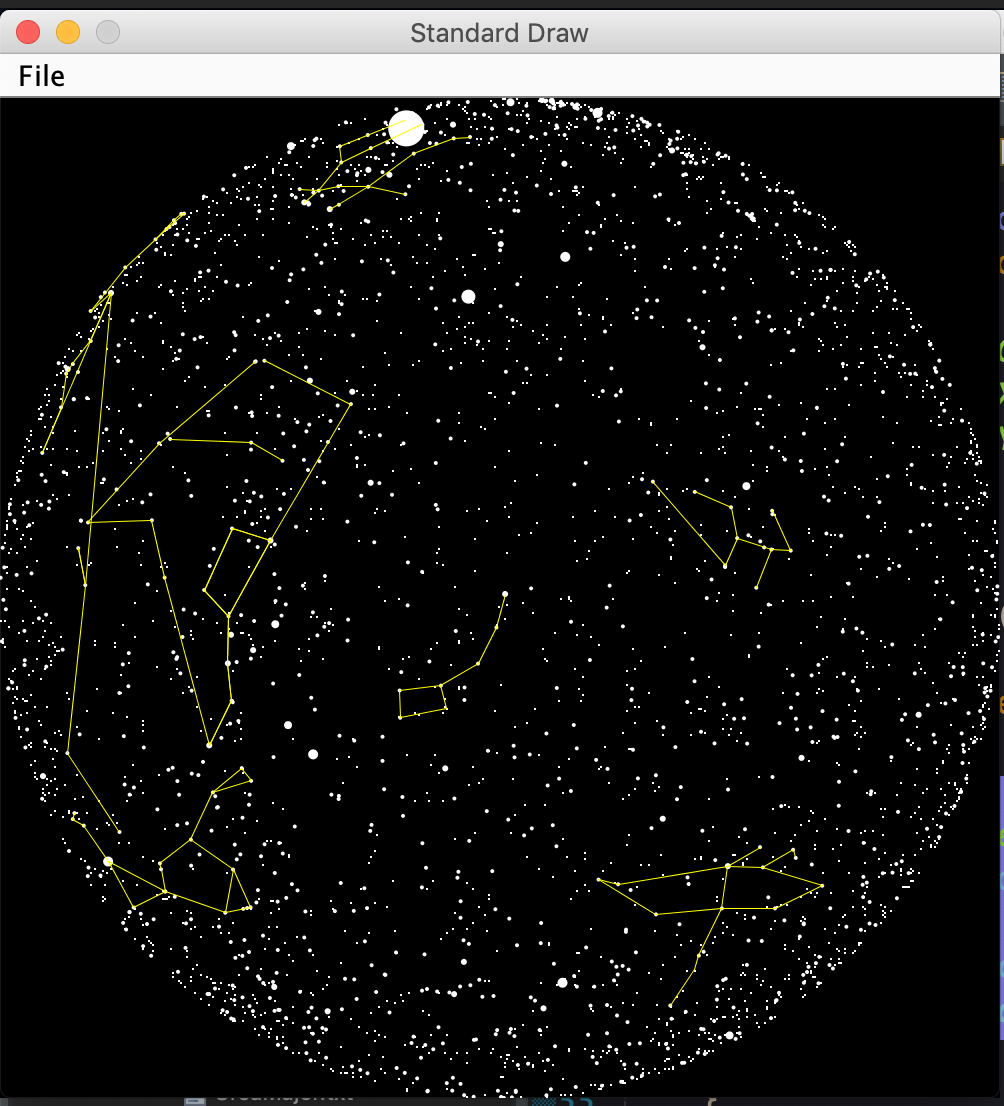
* 1. Re-run your code with this change; you should see stars of varying brightnesses as shown to the right.

**Part 2: Drawing constellations**

1. You are provided several text files that describe constellations. Each file contains two comma-separated star names per line of the file. Each pair of names represent one line to be drawn for the constellation.
2. Write a method void drawConstellation(String fileName). This method reads the constellation data from the supplied file and draws the constellation to screen on top of the star chart.
   1. Look at the format of a constellation file; stars are listed by name, as is the custom. However, names can be ambiguous, so stars are generally cataloged by their unique ID number; we need a way of mapping a star's name to its ID (and location). This is *exactly* what HashMaps are for!
   2. Create two HashMap object instance variables. The first will map a star's Draper number (its unique integer ID) to its location (a Point). The second will map a star's name (or names) to its Draper number. You will hopefully see the reason for this approach soon. Don't forget to initialize the HashMap objects in the constructor!
      1. In an appropriate location, populate the HashMaps with information. **Reminder:** a HashMap can have any number of duplicate values, only keys must be unique.
      2. Test that your HashMaps are working before proceeding; it may help to test with a smaller input file (or a subset of the given input file).
   3. Loop the input file using the HashMaps to link a star's name to its ID and then to its location. Draw a yellow line from each source star to each destination star.
   4. Run this method with "BigDipper.txt"; you should see the image below.



* 1. **(Optional)** Google a way to go through all the constellation files in the project folder (without hard-coding [manually inputting each file name]). Note that you only want to run the drawConstellation method with the constellation files.



Adapted from the **Star Charts and Constellations** project

*http://nifty.stanford.edu/2009/reid-starmap/starmap.html*