Star catalogue

Write a C++ application which helps astronomers catalogue stars as follows:

- The information about astronomers is in a text file. Each astronomer has **a name** (string) and **the constellation** he/she studies. This file is manually created and it is read when the application starts.
- Another file contains information about the catalogued stars. Each **Star** has **a name**, **a constellation** it belongs to, **the coordinates: RA (Right Ascension)** and **Dec (declination)** – given as integers and **a diameter**. These are read when the application starts.

1. When the application is launched, a new window is created for each astronomer, having as title the astronomer's name. Each window contains a **QTableView** that displays data for all the stars (name, coordinates, diameter), read from the input file. **[1p]**

2. A checkbox will allow the astronomer to see only the stars in the constellation they are studying (when the check box is checked, only the stars in that constellation are shown, when it is unchecked, all stars are shown) **[1p]**

3. Any astronomer can add a new star, by inputting its name, coordinates and diameter. The star's constellation will automatically be the constellation studied by the astronomer. This operation fails if the name is empty, there is another star with the same name, or the diameter is <= 0. **[1p]**

4. An astronomer can search for stars by name. The results will be shown in a list. The list modifies as the astronomer types the name. **[1p]**

5. When a star is added, all the astronomers will see the modified catalogue of stars. **The score for this functionality is given only if you are using the Qt Model/View architecture. [1.5p]**

6. Each astronomer will have a "*View*" button, that allows them to see the star selected in the **QTableView,** within its constellation. A new window will show all stars in the current constellation. Each star will be represented according to its coordinates, using a circle proportional to its diameter. The selected star will be red, while all the others will be black. **[1.5p]**

7. When the application is exited, all stars are saved to the file, sorted by constellation. **[1p]**

default 1p

Observations

1. Specify and test the function which adds a star **[1p]**
2. Use a layered architecture. If you do not use a layered architecture, you will receive 50% of each functionality.

Non-functional requirements

1. Use STL to represent you data structures.
2. Use layouts for automatic resizing and repositioning of your widgets.
3. Use the exception mechanism.

You are allowed to use Qt Designer.