**PROJECT REPORT**

**Class Description:**

There are in all 13 classes that are used in this project. These classes are divided into 3 packages viz. Base, Geom and ProjScr.

The Base PKG contains the 4 classes that are provided by the course/Lab instructor.

Below is the description of the other packages that were either already created during the Lab assignments or were created specifically for this project.

ProjSrc PKG: This is the main package of the project and contains 2 classes.

MainCall – This is the main executable class of the project. It is divided into multiple modules based on the functionality. All the dedicated methods are called from the Main().

City – This class is similar to the one developed with same name during Lab assignments but here certain modifications had been done to it, so that the creation of a city entity is done here. It is accessed from the MainCall class.

Geom PKG: This package hosts all the classes to create and maintain the different shapes used in the project. Each of the below classes have the getter() and setter() for their attributes.

Shape – It is a base class that holds all the general attributes and associated methods of the shape entity. This class is extended by the following shape-specific classes.

Circle – Class to handle the circle shape objects

Label – Class to handle the label objects

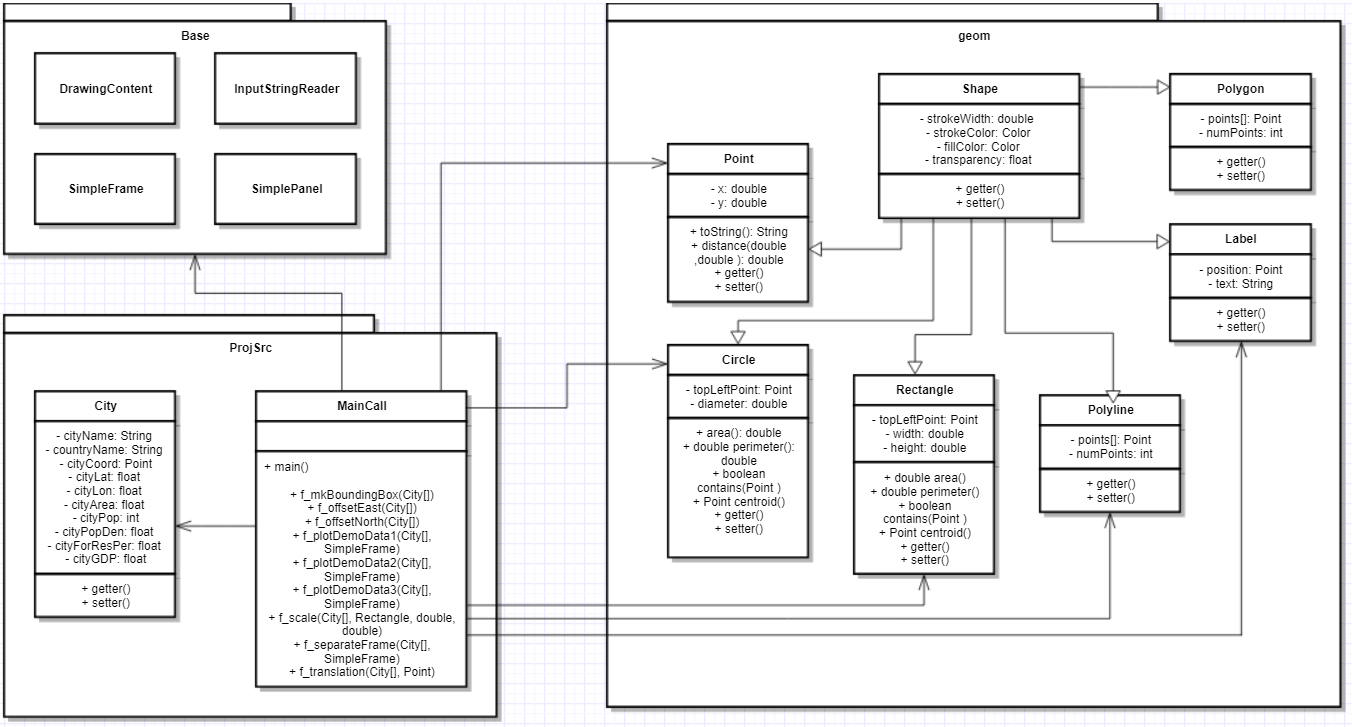
Point – Class to handle the point shape objects

Polygon – Class to handle the polygon shape objects

Polyline – Class to handle the polyline shape objects

Rectangle – Class to handle the rectangle shape objects

**Class Diagram:**



**Steps:**

The standard flow of project is as below,

1. The application accepts an input file (in .csv format) that contains the list of records of the capital cities in Europe along with their other details like area, population, GDP, etc.
2. This input file is then parsed and then each record is stored in a separate City class object.
3. In the next step, the module to create bounding box is called. This method works on the array of city objects and accesses their coordinates to identify the points on the extremes.
4. After that, a pair of modules is called to create the offset. These 2 functions shift the coordinates of the cities towards east and north such that the east-most city will have the X-coordinate as 0 and also the south-most city will have the Y-coordinate as 0. The coordinates of all the other cities will change accordingly
5. Another advantage of doing this is that it eliminates the negative coordinates. Further it becomes easier when the scaling and translation is done.
6. Till this point whatever manipulations that were done to city coordinates that was to the actual coordinates. Now we do their scaling so that they would be compatible with the frame on which the cities would be plotted.
7. As given in the project guidelines, first a generic scaling is done based on the comparison between height and width of bounding box. The scaling factor obtained here is applied to both X and Y-coordinates alike. The results of this scaling are printed and verified.
8. And then the next scaling is done, where a separate scaling factor is calculated for the vertical and horizontal directions. These factors are then multiplied so that we get the frame based coordinates for the cities.
9. The next step is to flip the coordinates vertically such that cities with lower values of Y-coordinate/latitude are on the bottom side of the frame. This is done in the same module as translation.
10. After this we are finished with manipulations of coordinates and they are in the form that is needed to plot them on a frame on screen.
11. From the main(), a module is called that separates the frame area in two sections, one for the city coordinates and the other part for plotting the demographic data of the cities.
12. Three methods are called to plot the demographic data for the cities
    1. Circular buffer around the city coordinates, whose radius is proportional to the area of city and the color saturation depends on the population density.
    2. GDP of each of the 10 cities is plotted in top right side. The horizontal bar chart indicates the GDP (in billion Euro) for each city.
    3. The second bar graph to the right indicates the total population value of cities and also shows the percent of foreign population both, in figures and graphically.
13. The city coordinates are plotted on the left of the frame. They also have buffer around them as explained in 12.a.

**Summary of challenging aspect of Project:**

1. The most challenging thing about this project is that it appears to very simple and straight forward but as you start developing, one needs to take into consideration a lot of aspects and parameters.
2. It forces you to visualize the actual coordinate system and the virtual system of the frame.
3. More attention needs to be given while doing the scaling and translation of the coordinates.
4. It was challenging as well to think on which demographic data to be plotted graphically and how it can be done.

**Other details:**

1. Argument to the main function needs to be given as .csv file name.
2. Some alteration had been done with the frame limits and its buffer size, so that all the cities are plotted well inside the frame and their scaling is restored correctly.
3. For filling the buffer circle based on density, had to use a threshold value instead of normalizing the density value for all the cities. The density of Paris city was around 4 times more than the 2nd most dense city in the list, because of which the normalized values don’t appear informative as all buffer other than Paris look pale and similar to each other. So the density values greater than a threshold will saturation as one.