All the charts and created by using **Factory Pattern**

chartFactory.ts :

It crates the Objects for the all the classes that plots various charts like line chart, bar chart, stacked bar chart, real-time line chart, heat map

```
export class ChartsFactory {

  // Return Object for SimpleMovingAverage1hr class
  createSimpleMovingAverage1hr() {
    return new SimpleMovingAverage1hr();
  }

  // Return Object for SimpleMovingAverage24hr class
  createSimpleMovingAverage24hr() {
    return new SimpleMovingAverage24hr();
  }

  createSimpleMovingAverage7days() {
    return new SimpleMovingAverage7days();
  }

  // Return Object for RealTimeLine class
  createRealTimeDataLine() {
    return new RealTimeLine();
  }

  createStackedBarChart(){
    return new StackedBarChart()
  }

  createLineChart(){
    return new LineChart();
  }

  createBarChart(){
    return new BarChart();
  }

  createHeatMap(){
    return new HeatMap();
  }
}
```

charts.ts

It is the interface implemented by all the classes to draw the chart.

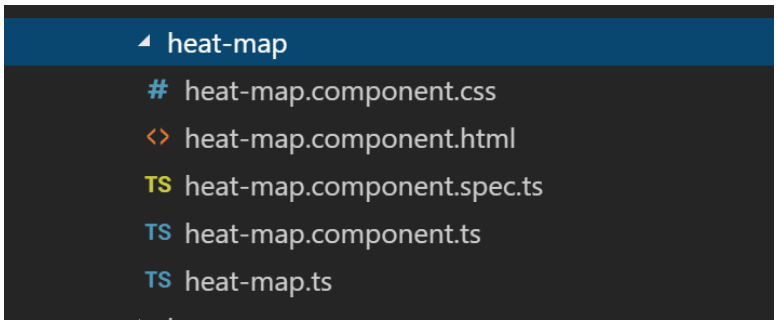
```
export interface Charts {
    drawChart(data,x,y,svg);
}
```

Components :

The Angular Components that are created to draw different charts are:

- heat-map
- stacked-bar-chart
- real-time-line-chart
- bar-chart-places-reviews
- line-chart-places-reviews

heat-map



heat-map.component.css : It contains all the additional styles for the component.

heat-map.component.html : It contains all the HTML template for the heat map.

heat-map.component.ts : It is creates the object for the factory class that returns the class that contains the method to plot the heat map.

Sample Code:

```
    //Creating object for the factory class
```

```
    const chartsFactory = new ChartsFactory();

    //Return object for HeatMap Class that plot Heat Map
    this.heatMap = chartsFactory.createHeatMap();
```

call to the drawChart() method plots the heat map on the user interface.

```
  this.heatMap.drawChart(this.heatmap,this.heatMapData,this.map);
```

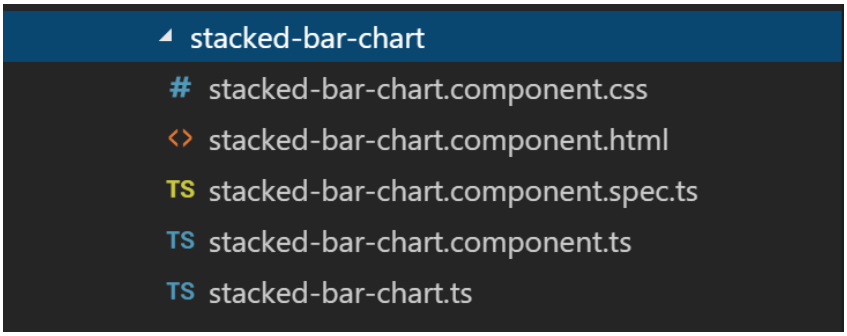
heat-map.ts: it implements the interface that contains drawChart() method that plots the heatmap on the user interface

```
export class HeatMap implements Charts{
  heatmap;
  map;

 //Plot Heat Map
  drawChart(heatmap,data,map,g){
    this.heatmap = new google.maps.visualization.HeatmapLayer({
      data: data
    });
    this.heatmap.setMap(this.map);
  }

}
```

stacked-bar-chart



stacked-bar-chart.component.css : It contains all the additional styles for the component.

stacked-bar-chart.component.html : It contains all the HTML template for the stacked bar chart.

stacked-bar-chart.component.ts : It is creates the object for the factory class that returns the class that contains the method to draw the stacked bar chart.

Sample Code:

```
  // Creates the object of the factory class
    const chartsFactory = new ChartsFactory();

    //factoty class return the object of the stacked bar chart class
    this.stackedBarChart = chartsFactory.createStackedBarChart();
```

call to the drawChart() method plots the stacked bar chart on the user interface.

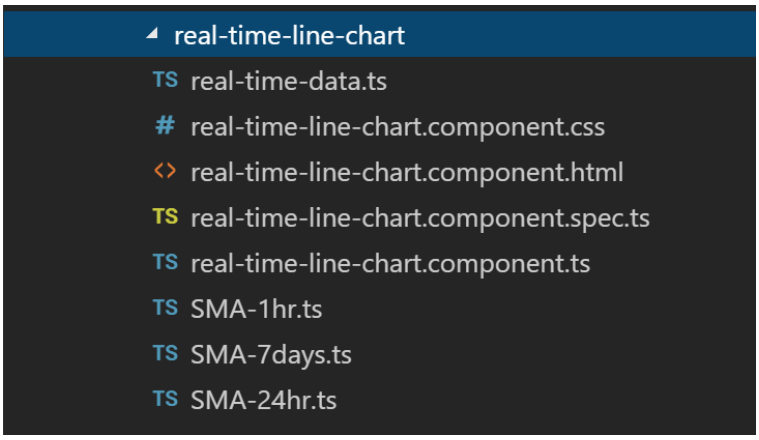
```
this.stackedBarChart.drawChart(this.stations, null, null, null);
```

stacked-bar-chart.ts: it implements the interface that contains drawChart() method that plots the stacked bar chart for the divvy station available docks and available bikes on the user interface

Sample Code:

```
export class StackedBarChart implements Charts{
 drawChart(data,x,y,g) {
 //All the code that plots stacked bar chart goes here.
 }
}
```

real-time-line-chart



real-time-line-chart.component.css : It contains all the additional styles for the component.

real-time-line-chart.component.html : It contains all the HTML template for the line chat.

real-time-line-chart.component.ts: This contains the typescript for implementation of line chart. It creates the object for factory class ChartFactory to draw the lines for real time data based on the user selection and simple moving average for 1hr,24 hrs and 7 days.

Sample Code:

```
   //Creating Object to Factory class
    const chartsFactory = new ChartsFactory();


    // Calling createSimpleMovingAverage1hr,createSimpleMovingAverage24hr in
impleMovingAverage class
    this.simpleMovingAverage1hr = chartsFactory.createSimpleMovingAverage1hr();
    this.simpleMovingAverage24hr = chartsFactory.createSimpleMovingAverage24hr();
    this.simpleMovingAverage7days =
chartsFactory.createSimpleMovingAverage7days();
    this.realTimeDataLine = chartsFactory.createRealTimeDataLine();
```

call to the drawChart() method plots the real-time line chart on the user interface.

```
   if (this.dataType === "SMA") {
        if(this.selected === "24 hr"){
        this.simpleMovingAverage24hr.drawChart(this.stations, this.x, this.y,
this.svg);
        }else if(this.selected === "168 hr"){
```

```
         this.simpleMovingAverage7days.drawChart(this.stations, this.x, this.y,
this.svg);
        }else{
          this.simpleMovingAverage1hr.drawChart(this.stations, this.x, this.y,
this.svg);
        }
      }else{
        this.realTimeDataLine.drawChart(this.stations, this.x, this.y, this.svg);
      }
    });
```

real-time-data.ts:

It implements the interface Charts and contains code to draw line for real time data based on the user selection for 1hr,24 hrs and 7days.

Sample Code:

```
export class RealTimeLine implements Charts{
  line: d3Shape.Line<[number, number]>;

    //public drawLine(stations,x,y,svg) {
      drawChart(data,x,y,svg){
        this.line = d3Shape
          .line()
          .x((d: any) => x(new Date(d.lastCommunicationTime.replace(/-/g, '/'))))
          .y((d: any) => y(d.availableDocks));
        svg
          .append("path")
          .datum(data)
          .attr("class", "line")
          .attr("id", "line")
          .attr("d", this.line);
      }
}
```

SMA-1hr.ts:

It implements the Charts interface and contains code to draw line for Simple Moving Average for 1hr.

```
export class SimpleMovingAverage1hr implements Charts{
```

```
        drawChart(stations,x,y,svg){
//the code to draw SMA for 1 hr goes here
    }
}
```

SMA-24hr.ts:

It implements the Charts interface and contains code to draw line for Simple Moving Average for 24hrs.

```
export class SimpleMovingAverage24hr implements Charts{

        drawChart(stations,x,y,svg){
//the code to draw SMA for 24 hrs goes here
    }
}
```

SMA-7days.ts:

It implements the Charts interface and contains code to draw line for Simple Moving Average for 7days.

```
export class SimpleMovingAverage7days implements Charts{

        drawChart(stations,x,y,svg){
//the code to draw SMA for 7days goes here
    }
}
```

bar-chart-places-reviews

bar-chart-places-reviews.component.css : It contains all the additional styles for the component.

bar-chart-places-reviews.component.html : It contains all the HTML template for the bar chart.

bar-chart-places-reviews.component.ts : It is creates the object for the factory class that returns the class that contains the method to draw the bar chart.

Sample Code:

```
  //Creates object for the factory class
    const chartsFactory = new ChartsFactory();

    //Factory class returns object for the bar chart
    this.barChart = chartsFactory.createBarChart();
```

call to the drawChart() method plots the bar chart on the user interface.

```
        this.barChart.drawChart(this.places,this.x,this.y,this.g);
```

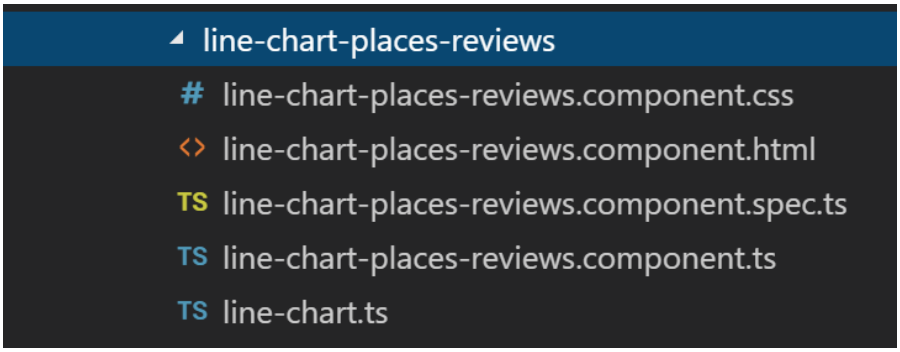
bar-chart.ts: it implements the interface that contains drawChart() method that plots the bar chart for place and review count on the user interface

```
export class BarChart implements Charts{
        drawChart(data,x,y,g) {

//All the code that draw the bar chart goes here.

      }
}
```

line-chart-places-reviews



line-chart-places-reviews.component.css : It contains all the additional styles for the component.

line-chart-places-reviews.component.html : It contains all the HTML template for the line chart.

line-chart-places-reviews.component.ts : It is creates the object for the factory class that returns the class that contains the method to draw the line chart.

Sample Code:

```
  //Creates object for factory class
    const chartsFactory = new ChartsFactory();

    //factory class returns object for the line chart
    this.lineChart = chartsFactory.createLineChart();
```

call to the drawChart() method plots the line chart on the user interface.

```
this.lineChart.drawChart(this.places,this.x,this.y,this.svg);
```

line-chart.ts : it implements the interface that contains drawChart() method that plots the line chart for place and review count on the user interface

```
export class LineChart implements Charts{
        drawChart(data,x,y,g) {

//All the code that draw the bar chart goes here.

      }
}
```