API

```java
/**
 * Interface for the display plugin to be registered and used in {@link Framework}.
 *
 * @author KathyJin, StellaZhang, YanningMao
 */
public interface DisplayPlugin {
        /**
         * Checks whether the data can be displayed in this visual format.
         *
         * @param data, the data to be displayed
         * @return availability, whether the data can be displayed in this visualization format
         */
        public boolean isAvailable(DataSet data);

        /**
         * Creates a JPanel that displays the data visualization
         *
         * @param data, the data to be displayed
         * @return panel, the panel painted with the data visualization
         */
        public JPanel display(DataSet data);

        /**
        * Called when the plugin is first registered with the framework,
        * giving the plugin a chance to perform any initial set-up
        * before the game has begun (if necessary).
        *
        * @param framework, The {@link Framework} instance with which the plugin is registered.
        */
        public void onRegister(Framework framework);
}
```

```java
/**
 * Interface for the data plugin to be registered and used in {@link Framework}.
 *
 * @author KathyJin, StellaZhang, YanningMao
 */
public interface DataPlugin {
        /**
         * Extracts raw data from the path fileName, process raw data, converts it into
         * the DataSet format, and return processed data.
         *
         * @param path, the source path (path/fileName/URL) to extract raw data
         * @return dataSet, the data after processing
         */
         public DataSet process(String path);

         /**
         * Called when the plugin is first registered with the framework,
         * giving the plugin a chance to perform any initial set-up
         * before the game has begun (if necessary).
         *
         * @param framework, The {@link Framework} instance with which the plugin is registered.
         */
         public void onRegister(Framework framework);
}
```

```java
/**
 * The interface for the data visualization framework.
 * Customized with DataPlugin and DisplayPlugin, and provides interactive data
 * visualization functionality.
 *
 * @author KathyJin, StellaZhang, YanningMao
 */

public interface Framework {
        /**
         * Gets the path input by the user for data extraction.
         *
         * @return path, the source path to get data
         */
        String getPath();

        /**
         * Loads the data plugin to be used for source data extraction.
         *
         * @param dataPlugin, the data plugin selected for data extraction.
         */
        void loadPlugin(DataPlugin dataPlugin);

        /**
         * Loads the display plugin to be used for data visualization
         * (i.e. sets current display plugin to displayPlugin)
         *
         * @param displayPlugin, the display plugin selected for data visualization.
         */
        void loadPlugin(DisplayPlugin displayPlugin);
```

```java
        /**
         * Register all available data plugins provided by user to be used by the framework.
         *
         * @param dataPlugin, a data plugin provided by the user.
         */
        void registerPlugin(DataPlugin dataPlugin);

        /**
         * Register all available display plugins provided by user to be used by the framework.
         *
         * @param displayPlugin, a display plugin provided by the user.
         */
        void registerPlugin(DisplayPlugin displayPlugin);

        /**
         * Sort the selected columns (fields) of data using the comparator.
         * Columns that appear first have higher priorities in the sorting.
         *
         * @param cols, ordered indices of the selected columns to be sorted
         * @param cmp, a comparator that specifies the order of sorting.
         */
        void sort(List<Integer> cols, Comparator cmp);

        /**
         * Filters selected column (filed) of data using the filter.
         *
         * @param col, index of the selected column
         * @param flt, a filter to be used for filtering
         */
        void filter(int col, Filter flt);

}
```

```java
/**
 * An observer interface that listens for changes of state form the framework.
 * The {@link FrameworkImpl} calls these methods to notify the {@link FrameworkGui}
 * when it should update its display.
 *
 * @author KathyJin, StellaZhang, YanningMao
 */
public interface StateChangeListener {
        /**
         * Called when a new {@link DataPlugin} is registered with the framework
         *
         * @param plugin The Plug-in that has just been registered.
         */
        void onDataPluginRegistered(DataPlugin plugin);

        /**
         * Called when a new {@link DisplayPlugin} is registered with the framework
         *
         * @param plugin The Plug-in that has just been registered.
         */
        void onDisplayPluginRegistered(DisplayPlugin plugin);

        /**
         * Called when a file is already read from the given URL/local path.
         */
        void onFileReady();

        /** Called when data is extracted from the file*/
        void onDataExtracted();

        /** Called when a function type is selected*/
        void onFunctionSelected();

        /** Called when a visualization is selected*/
        void onChartSelected();
}
```

```java
/**
 * The framework GUI implementation. This class is responsible for displaying
 * the framework GUI to the screen, and for forwarding events to
 * {@link FrameworkImpl} when GUI-related events are detected.
 *
 * @author KathyJin, StellaZhang, YanningMao
 */
public class FrameworkGui implements StateChangeListener {

    public FrameworkGui(FrameImpl fc) {}

    @Override
    public void onDataPluginRegistered(DataPlugin plugin) {}

    @Override
    public void onDisplayPluginRegistered(DisplayPlugin plugin) {}

    @Override
    public void onFileReady() {}

    @Override
    public void onDataExtracted() {}

    @Override
    public void onFunctionSelected() {}

    @Override
    public void onChartSelected() {}
}
```

```java
/**
 * The internal representation of data extracted.
 * Porvides methods that can get a row(record), a column(filed) from the data set.
 *
 * @author KathyJin, StellaZhang, YanningMao
 */
public interface DataSet {
        /**
         * Retrieve data from row i.
         *
         * @param i, the row index of the record that needed to be retrieved
         * @return record, all the data in row i
         */
        List<? extends Object> getRow(int i);

        /**
         * Retrieve data from column i.
         *
         * @param j, the column index of the column that needed to be retrieved
         * @return data, all the data in column i
         */
        List<? extends Object> getCol(int j);

        /**
         * Retrieve data block from row i,column j.
         *
         * @param i, row index
         * @param j, column index
         * @return dataBlock, the data at row i and col j
         */
        String getCell(int i, int j);
```

```java
        /**
         * Return column of the data set.
         *
         * @return title, title of the data set
         */
        String getTitle();

        /**
         * Return name of the indicated row.
         *
         * @param i, row index
         * @return title, name of the indicated row
         */
        String getRowTitle(int i);

        /**
         * Return name of the indicated column.
         *
         * @param j, column index
         * @return title, name of the indicated column
         */
        String getColTitle(int j);
}
```