**2.2. Label**

Label is a part of JavaFX package. Label is used to display a short text or an image, it is a non-editable text control. It is useful for displaying text that is required to fit within a specific space, and thus may need to use an ellipsis or truncation to size the string to fit. Labels also are useful in that they can have mnemonics which, if used, will send focus to the Control listed as the target of the label For property. Label can only a display of text or image and it cannot get focus.

Constructor for the Label class are:

1. Label(): Creates an empty label
2. Label(String t): Creates Label with given text.
3. Label(String t, Node g): Creates a Label with the given text and graphic.

**2.3. Button**

A simple button control can contain text and/or a graphic. A button control has three different modes

* **Normal:** A normal push button.
* **Default:** The default button is rendered differently to make it apparent to users that it should be the default choice should they be unclear as to what should be selected. The behavior of the default button differs depending on the platform in which it is presented:
* **Cancel:** A Cancel Button is the button that receives a keyboard VK\_ESC press, if no other node in the scene consumes it.

When a button is pressed and released a [ActionEvent](https://openjfx.io/javadoc/12/javafx.base/javafx/event/ActionEvent.html) is sent. The application can perform some action based on this event by implementing an [EventHandler](https://openjfx.io/javadoc/12/javafx.base/javafx/event/EventHandler.html) to process the [ActionEvent](https://openjfx.io/javadoc/12/javafx.base/javafx/event/ActionEvent.html).

**2.4. TextField**

The TextField class implements a UI control that accepts and displays text input. It provides capabilities to receive text input from a user. Along with another text input control, PasswordField, this class extends the TextInput class, a super class for all the text controls available through the JavaFX API.

**2.5. TableView**

The application can use TableView, TableColumn, and TableCell classes from JavaFX API to represent data in a tabular form. The data in a table is filled by implementing the data model and by applying a cell factory.

The table classes can sort data in columns and to resize columns when necessary.

**2.6. TableColumn**

When creating a TableColumn instance, perhaps the two most important properties to set are the column text (what to show in the column header area), and the column cell value factory (which is used to populate individual cells in the column).

**2.7. ObservableList**

Uses of ObservableList in javafx.collections Abstract class that serves as a base class for ObservableList implementations.

Creates and returns a typesafe wrapper on top of provided observable list. Creates a new empty observable list that is backed by an arraylist.

**2.8. Stage**

A JavaFX Stage, javafx.stage.Stage, represents a window in a JavaFX desktop application. Inside a JavaFX Stage you can insert a JavaFX Scene which represents the content displayed inside a window - inside a Stage.

When a JavaFX application starts up, it creates a root Stage object which is passed to the start(Stage primaryStage) method of the root class of your JavaFX application. This Stage object represents the primary window of your JavaFX application.

The application can create new Stage objects later in your application's life time, in case your application needs to open more windows.

**2.9. Scene**

The JavaFX Scene class is the container for all content in a scene graph. The background of the scene is filled as specified by the fill property.

The application must specify the root Node for the scene graph by setting the root property. If a Group is used as the root, the contents of the scene graph will be clipped by the scene's width and height and changes to the scene's size (if user resizes the stage) will not alter the layout of the scene graph. If a resizable node (layout Region or Control is set as the root, then the root's size will track the scene's size, causing the contents to be relayed out as necessary.

The scene's size may be initialized by the application during construction. If no size is specified, the scene will automatically compute its initial size based on the preferred size of its content. If only one dimension is specified, the other dimension is computed using the specified dimension, respecting content bias of a root.

**2.10. Summary**

In this chapter, the presentation of javaFx and its contents.In addition, the concepts of logices used in this project are explained in detail. Project design and implementation results are discussed in later chapter.