File Explorer
Documentation

Muhammad Azmain Adel
ID : 1405075

April 25, 2017

For this assignment on design patterns, the task was to make a simple File Explorer. It has two different styles of viewing files and also has a tree like hierarchy.

Description of Classes

- **Main**
  This class controls the stage for the explorer UI. We can change the title and stage size from this class.
  Functions in this class :
  1. start()
  2. main()
- **Controller**
  This class controls the *.fxml* document for our UI. It controls the buttons, views and textfields used in this project.
  Functions in this class :
  1. initialize()
  2. switchToTiles() : For the Button for switching to Tiles View.
  3. switchToTable() : For the Button for switching to Table View.
  4. goBack() : For the 'Back' Button.
  5. expandTree() : This method expands the tree when clicked and shows it in Tiles/Table view.
  6. makeTilesView()
  7. makeTableView()
- **TreeViewItem**
  This class is for the items in the Tree View. It is extended from the *TreeItem* class.
  Functions in this class :

1. Constructors
2. buildChildren()
3. getChildren()
4. isLeaf()
5. toString() : For returning the absolute path.

- **TilesViewItem**
  This class is for the items in the Tiles View. It is extended from the *VBox* class.
  Functions in this class :

  1. Constructors
  2. toString() : For returning the absolute path.

- **TableViewItem**
  This class is for the cells in the Table View. It is extended from the *File* class.
  Functions in this class :

  1. Constructors
  2. Getters & Setters for the 4 variables,
     (a) fileIcon (ImageView)
     (b) fileName (String)
     (c) fileSize (long)
     (d) dateModified (String)
  3. getPath() : For returning the absolute path.

- **FileIcon**
  This class contains the icon for each file. It takes the File and uses a method to return it's icon as ImageView item.
  Functions in this class :

  1. getImageView() : This function gets the icon from the system and converts it to an ImageView object, so that we can use it in the UI.

Design Patterns

The design patterns that were used in this project are :

Composite Pattern

- This design pattern was used to create the Tree View. Directories which has sub-directories are in this view. Every view can contain new directories, which leads to another directory objects.

- The parent-child relation was maintained properly. The whole hierarchy was shown.

Adapter Pattern

- TreeViewItem (Adapter) adapts TreeItem (Adaptee) and here TreeView is the Target.
- TilesViewItem (Adapter) adapts VBox (Adaptee) and the Target is TilePane.
- TableViewItem (Adapter) adapts File (Adaptee) and the Target is TableView.

Singleton Pattern

- The classes TilesViewItems and TableViewItems follow Singleton pattern.
- When makeTableView() or makeTilesView() methods are first called, only one instance of these classes are created according to the flag *viewType*. But every other time that one instance is updated, no other instance is created.

Factory Pattern

- The buttons *Details View* and *Tiles View* follow the Factory pattern.
- Among the two classes TreeViewItem and TableViewItem, the instance that is shown in the UI is determined by the buttons.