Documentation and Design Pattern Usage on *"File Explorer"*

Muhtasim Ulfat Tanmoy
1405086

May 2, 2017

1 Design Pattern Identification and Selection

The given file Explorer project has following requirements:

- **TreeView** - for showing file hierarchy.
- **TabletView** - for showing file in a table.
- **TileView** - for showing file in a tiles.
- **Switching** - convenient way for transition between views.

Now, **TreeView** uses the composite design pattern.As we can create different node and each node can also have other nodes as child it matches all the specification for **Composite design pattern**.

TableView and **TileView** has list of files that has been adapted to the view in our code through **TableItem** and **TileItem** class(works as adapter).The **Tile** class further defines the structure of our TileView.The **File** class is the adaptee and **TableView** and **TilePane** is the target class ,this clearly represents the **Adapter class pattern**.

Switching between views like making the same set of of files appear in different form in front of users clearly references**Factory design pattern**..The **BaseItem** class defines the base Item for our both tile and table view.Now when **TableView** is activated the **BaseItem** will autometically get **TableItem** and for othe case it will get **TileItem** through **Tile** class.

The **Preferences** class is static therefore initiated once.Now only changing the view from that class loads the app in a certain view.It can't be initiated again and all class gets the same value on that class.

So,**TreeView class** in javafx implements the composite pattern,the Tableview class does the adapter pattern and differnt view switching implements the factory design pattern.The main function that is initialized only once during the lifetime of the app implements singleton degign pattern.

2 Design Pattern Usage

The design pattern usage on our project is described with relevent classes below:

The Main class runs only once and it sets the fxml in the app.It sets the resolution and therefore implements Singleton Pattern.

The Controller class controls the whole app with different properties.All setOnClickListener is implemented here.

The Treeview API is used here to show the hierarchy of folders to navigate to.This **Treeview** implements **composite** design pattern.

The **myTreeItem** class implements all the functionalities of a single tree node or element.It's onClick EventHandler implements what to do when an item is clicked.The folder icon is changed when a tree node having children is expanded or closed.And also a list of all available childen including folder and file is shown

Now there are two ways of showing the all the file list in the current working directory.

Tableview : The **Target class** where all the **File** classes are are adapted to **Tableview** through **TableItem** class and CellValueFactory.

The **TableItem class** takes the **File** class API and relevent file values are filtered througnh this class to our **Tableview** class.

TilePane : Here all the files are adapted to Tilepane view through **TileItem**s.Here each **TileItems** are actually a going through **Tile** class that extends vBox API class.

TileItem : class extends vBox and defines an structure for each item in our tileView.It autometically sets image for file or directory.It also implements **onClickListener** for easy transition to different directories.

The above two methods implement Adapter class

Switching View : Done through **ItemFactory** Class.This class takes the view and sets them accordingly.ToggleGroup has been used for switching views.

Preferences class can be initialized once and therefore all class can only access it's one instance.Implements **Singleton design Pattern**

3 Design Pattern Implementation

The design pattern implementation is provided in code.

Back functionality is used to to upper hiererchy.

Go: User can also go to differnt folders by typing the path to that folder.

And a meaningful function name has been used and proper documentation is given as comment beside every function and variable declaration.

Problem faced:

1.onClickListener on each item.
2.extending class functionality loss.
3.Interface concept.
4.Controller manipulation.