ASSIGNMENT ON DESIGN PATTERNS

MAY 2, 2017

MD. HASIN ABRAR
SECTION: A2 ROLL: 1405048

CLASSES:

FileExplorer: This is the class that launches the initial GUI. It takes the user to the default TableView interface of the File Explorer.

Methods within:

A. static void main (String [] args)
B. void start (Stage stage)

FileValue: This class contains all the attributes of each object that is shown in the TableView, TilesView and TreeView GUI. This includes file path, file name, file size, file icon and last modified date.

Methods within:

A. String toString() : Used to show the file name in the TreeView GUI.
B. void setFileSize(long fileSize)
C. void setFileIcon(ImageView fileIcon)
D. void setCreateDate(String createDate)
E. long getFileSize()
F. ImageView getFileIcon()
G. String getCreateDate()
H. void setFilePath(Path filePath)
I. void setFileName(String fileName)
J. Path getFilePath()
K. String getFileName()

ItemList: This class contains an ArrayList that contain the files of the current directory and one String attribute that holds the absolute path of the current directory.

Methods within:

A. static void fillUpList(Path dir)

FileIconSource: This class manages the file icon according to the file's extension.

Methods within:

A. static ImageView getIcon(String imageName)
B. static String getFileExt(String fname)
C. static javax.swing.Icon getJSwingIconFromFileSystem(File file)
D. static Image getFileIcon(String fname)
E. static Image jswingIconToImage(javax.swing.Icon jswingIcon)

TableViewResource: This class contains one TableView and four TableColumn attributes. It takes references from the FXML file and populates it with the directory data.

Methods within:

A. void setCellFactory()
B. void populateTableView()
C. void onDoubleClickActivateTable(TextField dirField)

TilesViewResource: It is used for populating the TilesView of the corresponding FXML file. It contains an ArrayList to populate the TilesView. Again, it has ScrollPane, TilePane, ImageView, and VBox as its attributes with which it takes references from the FXML file.

Methods within:

A. void populateTileView(TextField dirField) : This method takes the ScrollPane, puts TilePane over it, makes new VBox and adds a Label and an ImageView as its child. It also sets up the listener of the VBox.

TreeResource: This class creates the whole TreeView and passes the reference to any FXML file that requests. There can be only one instance of this class.

Methods within:

A. static TreeResource getResourceInstance ()
B. TreeView<FileValue> getTreeViewInstance()
C. void mainTree() : It sets up the root node and uses *createTree(FileValue dir, TreeItem<FileValue> parent)* method to create the rest of the tree.
D. TreeItem<FileValue> createTree(FileValue dir, TreeItem<FileValue> parent)
E. FileValue getFileValue(File file) : a method just to return a semi-dummy object of FileValue type.
F. void activateClickingTree(TableViewResource tableResource, TextField dirField)
G. void activateClickingTree(TilesViewResource tileResource, TextField dirField)

TableViewFXMLController: This is the controller class of the *tableViewFXML.fxml* file. It uses the *TreeResource* and *TableViewResource* classes to populate and control the GUI in the FXML files.

Methods within:

A. void toogleButtonAction(ActionEvent event)
B. void updateDirectory() : updates the directory TextField.
C. void initialize (URL url, ResourceBundle rb)

TilesViewFXMLController: Works similarly like *TableViewFXMLController*. Uses *TreeResource* and *TilesViewResource* classes to resolve its usage.

Methods within:

A. void toogleButtonAction(ActionEvent event)
B. void updateDirectory()
C. void initialize (URL url, ResourceBundle rb)

DESIGN PATTERN:

Composite Pattern:

A. TreeView: In *TreeResource* class, all the items of the *TreeView* (JAVA API) attribute that is the main tree are of the same *FileValue* type, indicating it a composite pattern.
B. TableView: In *TableViewResource* class, all the items of the *TableView* (JAVA API) attribute are from the *ObservableList* which contained all the objects of the same *FileValue* type. This makes this JAVA API following the composite pattern.

Singleton Pattern:

A. TreeResource: There can be only one instance of this class which indicates its following of the singleton pattern.

Adapter Pattern:

A. VBox: For this application, *TilePane* can't directly contain the icon and the name of the file. In *TilesViewResource* class, *VBox* (JAVA API) is used to give *TilePane* that functionality.
B. TilePane: Again, *ScrollPane* couldn't contain the *VBox* objects directly. So, in *TilesViewResource* class, it takes *TilePane* (JAVA API) as an adapter class to show the files.