Model

**package** final01.model;  
  
**import** final01.view.observers.AddingObserver;  
**import** final01.view.observers.DeletingObserver;  
**import** final01.view.observers.FIleListObserver;  
**import** final01.view.observers.SearchObserver;  
**import** java.nio.file.Path;  
  
  
*//implement strategy pattern***public interface** ModelInterface {  
  
 **void** addFile(String path);  
 **void** deleteFile(String path);  
 **void** searchFile(String path);  
 **void** getFileList();  
  
 Path getRootPath();  
 **void** setRootPath(String source);  
  
  
  
 **void** registerObserver(AddingObserver o);  
 **void** removeObserver(AddingObserver o);  
  
 **void** registerObserver(DeletingObserver o);  
 **void** removeObserver(DeletingObserver o);  
  
 **void** registerObserver(SearchObserver o);  
 **void** removeObserver(SearchObserver o);  
  
 **void** registerObserver(FIleListObserver o);  
 **void** removeObserver(FIleListObserver o);  
  
  
}

**package** final01.model;  
  
**import** final01.view.observers.AddingObserver;  
**import** final01.view.observers.DeletingObserver;  
**import** final01.view.observers.FIleListObserver;  
**import** final01.view.observers.SearchObserver;  
  
**import** java.io.File;  
**import** java.io.IOException;  
**import** java.nio.file.Files;  
**import** java.nio.file.Path;  
**import** java.nio.file.Paths;  
**import** java.util.ArrayList;  
**import** java.util.Arrays;  
**import** java.util.Collections;  
**import** java.util.List;  
  
*//The model doesn't know anything about controller and view  
//but the model informs observers about changes in its state***public class** Model **implements** ModelInterface{  
  
 **private** ArrayList<AddingObserver> **addingObservers**=**new** ArrayList<>();  
 **private** ArrayList<DeletingObserver> **deletingObservers**=**new** ArrayList<>();  
 **private** ArrayList<SearchObserver> **searchObservers**=**new** ArrayList<>();  
 **private** ArrayList<FIleListObserver> **fIleListObservers**=**new** ArrayList<>();  
  
 **private** Path **rootPath**;  
  
 **public** Model(Path rootPath) {  
 **this**.**rootPath** = rootPath;  
 }  
  
 @Override  
 **public** Path getRootPath() {  
 **return rootPath**;  
 }  
  
  
 *//++Main logic begin* **public void** setRootPath(String rootPath) {  
 Path path = **null**;  
  
 **try** {  
 path = Paths.*get*(rootPath);  
 setRootPath(path);  
 } **catch** (Exception e) {  
 System.***out***.println(**"Fail. Specified path is not exist"**);  
 **return**;  
 }  
 }  
  
  
 **public void** setRootPath(Path rootPath) {  
  
 **if** (!Files.*exists*(rootPath)) {  
 System.***out***.println(**"Fail. Specified path is not exist."**);  
 **return**;  
 }  
  
 **if** (!Files.*isDirectory*(rootPath)) {  
 System.***out***.println(**"Fail. Specified path is not to Directory path."**);  
 **return**;  
 }  
  
 **this**.**rootPath** = rootPath;  
 System.***out***.println(**"New working directory is "**+rootPath+**"\n"**);  
 }  
  
  
  
 @Override  
 **public void** addFile(String stringPath) {  
  
 Path file=Paths.*get*(stringPath);  
 **try** {  
 Files.*createFile*(**rootPath**.resolve(file));  
 notifyAddingObservers(stringPath, **true**);  
 } **catch** (IOException e) {  
 notifyAddingObservers(stringPath, **false**);  
 }  
  
 }  
  
 @Override  
 **public void** deleteFile(String stringPath) {  
  
 Path file=Paths.*get*(stringPath);  
 **try** {  
 Files.*delete*(**rootPath**.resolve(file));  
 notifyDeletingObservers(stringPath, **true**);  
 } **catch** (IOException e) {  
 notifyDeletingObservers(stringPath, **false**);  
 }  
 }  
  
 @Override  
 **public void** searchFile(String stringPath) {  
  
 Path file=Paths.*get*(stringPath);  
 Boolean result=Files.*exists*(**rootPath**.resolve(file));  
 notifySearchObservers(stringPath, result);  
 }  
  
  
  
 @Override  
 **public void** getFileList(){  
  
 File dir = **new** File(**rootPath**.toString());  
 File[] arrFiles = dir.listFiles();  
 List<File> lst = Arrays.*asList*(arrFiles);  
 Collections.*sort*(lst);  
 notifyFIleListObservers(lst);  
  
 }  
  
 *//--Main logic end  
  
  
 //Work with observers;* **private void** notifyAddingObservers(String fileName, Boolean isAdded){  
 **for** (AddingObserver addingObserver : **addingObservers**) {  
 addingObserver.updateAddingState(fileName, isAdded);  
 }  
 }  
  
 **private void** notifyDeletingObservers(String fileName, Boolean isDeleted){  
 **for** (DeletingObserver deletingObserver : **deletingObservers**) {  
 deletingObserver.updateDeletingState(fileName, isDeleted);  
 }  
 }  
  
 **private void** notifyFIleListObservers(List<File> filesList){  
 **for** (FIleListObserver fIleListObserver : **fIleListObservers**) {  
 fIleListObserver.updateFileList(filesList);  
 }  
 }  
  
 **private void** notifySearchObservers(String fileName, Boolean isFound){  
 **for** (SearchObserver searchObserver : **searchObservers**) {  
 searchObserver.updateSearchingState(fileName, isFound);  
 }  
 }  
  
 @Override  
 **public void** registerObserver(AddingObserver o) {  
 **addingObservers**.add(o);  
 }  
  
 @Override  
 **public void** removeObserver(AddingObserver o) {  
 **addingObservers**.remove(o);  
 }  
  
 @Override  
 **public void** registerObserver(DeletingObserver o) {  
 **deletingObservers**.add(o);  
 }  
  
 @Override  
 **public void** removeObserver(DeletingObserver o) {  
 **deletingObservers**.remove(o);  
 }  
  
 @Override  
 **public void** registerObserver(SearchObserver o) {  
 **searchObservers**.add(o);  
 }  
  
 @Override  
 **public void** removeObserver(SearchObserver o) {  
 **searchObservers**.remove(o);  
 }  
  
 @Override  
 **public void** registerObserver(FIleListObserver o) {  
 **fIleListObservers**.add(o);  
 }  
  
 @Override  
 **public void** removeObserver(FIleListObserver o) {  
 **fIleListObservers**.remove(o);  
 }  
 }

Controller

**package** final01.controller;  
  
*//implement strategy pattern***public interface** ControllerInterface {  
  
 **void** addFile();  
  
 **void** deleteFile();  
  
 **void** searchFile();  
  
 **void** getFileList();  
  
 **void** setRootDirectory();  
  
 **void** exit();  
  
}

**package** final01.controller;  
  
**import** final01.util.StringHelper;  
**import** final01.model.Model;  
**import** final01.model.ModelInterface;  
  
  
  
**public class** Controller **implements** ControllerInterface {  
  
 *//strategy pattern* **private** ModelInterface **model**;  
  
  
 **public** Controller(ModelInterface model) {  
 **this**.**model** = model;  
 }  
  
  
 **private** String readFileName(){  
 System.***out***.println(**"Enter file name"**);  
 **return** StringHelper.*readString*();  
 }  
  
  
 *//The controller receives additional information and delegates the execution of the model methods* @Override  
 **public void** addFile() {  
 **model**.addFile(readFileName());  
 }  
  
 @Override  
 **public void** deleteFile() {  
 **model**.deleteFile(readFileName());  
 }  
  
 @Override  
 **public void** searchFile() {  
 **model**.searchFile(readFileName());  
 }  
  
 @Override  
 **public void** getFileList() {  
 **model**.getFileList();  
 }  
  
 @Override  
 **public void** setRootDirectory() {  
 System.***out***.println(**"Enter path to main directory"**);  
 String source= StringHelper.*readString*();  
 **model**.setRootPath(source);  
 }  
  
 @Override  
 **public void** exit() {  
 System.*exit*(0);  
 }  
  
  
  
  
}

View

**package** final01.view.observers;  
  
**public interface**AddingObserver {  
  
 **void** updateAddingState(String fileName, **boolean** isAdded);  
  
}

**package** final01.view.observers;  
  
**public interface** DeletingObserver {  
 **void** updateDeletingState(String fileName, **boolean** isDeleted);  
}

**package** final01.view.observers;  
  
**import** java.io.File;  
**import** java.util.List;  
  
**public interface** FIleListObserver {  
 **void** updateFileList(List<File> fileList);  
}

**package** final01.view.observers;  
  
**public interface** SearchObserver {  
  
 **void** updateSearchingState(String fileName, **boolean** isFounded);  
  
}

**package** final01.view;  
  
**public interface** ViewInterface {  
  
 **void** buildMenuStructure();  
 **void** showMenu();  
 **void** selectAndPerformOperation();  
}

**package** final01.view;  
  
**import** final01.util.StringHelper;  
**import** final01.controller.ControllerInterface;  
**import** final01.menu.MenuItem;  
**import** final01.model.ModelInterface;  
**import** final01.view.observers.AddingObserver;  
**import** final01.view.observers.DeletingObserver;  
**import** final01.view.observers.FIleListObserver;  
**import** final01.view.observers.SearchObserver;  
  
**import** java.io.File;  
**import** java.util.List;  
  
*//implement observer pattern***public class** View **implements** AddingObserver, DeletingObserver, FIleListObserver, SearchObserver,ViewInterface {  
  
 *//root of menu tree* **private** MenuItem **rootMenuItem**;  
  
 *//current composite in menu tree* **private** MenuItem **currentComposite**;  
  
 *//implement strategy pattern* **private** ControllerInterface **controller**;  
 **private** ModelInterface **model**;  
  
  
  
  
 **public** View(ControllerInterface controller, ModelInterface model) {  
 **this**.**controller** = controller;  
 **this**.**model** = model;  
 **rootMenuItem**=**new** MenuItem(**"root"**,**null**);  
 **currentComposite**=**rootMenuItem**;  
  
 model.registerObserver((AddingObserver) **this**);  
 model.registerObserver((DeletingObserver) **this**);  
 model.registerObserver((SearchObserver) **this**);  
 model.registerObserver((FIleListObserver) **this**);  
  
 }  
  
 **public** MenuItem getCurrentComposite() {  
 **return currentComposite**;  
 }  
  
 **public void** setCurrentComposite(MenuItem currentComposite) {  
 **this**.**currentComposite** = currentComposite;  
 }  
  
 @Override  
 **public void** showMenu(){  
 System.***out***.println(**"Enter number of item:"**);  
 **for** (**int** i = 0; i <**currentComposite**.**menuItems**.size() ; i++) {  
 System.***out***.println(i+1+**". "**+**currentComposite**.**menuItems**.get(i).getCommandName());  
 }  
 }  
  
 **public void** menuLevelUp(){  
 setCurrentComposite(getCurrentComposite().getParentMenuItem());  
 showMenu();  
 }  
  
 @Override  
 **public void** selectAndPerformOperation(){  
 **try** {  
 **int** numberOfCommand=Integer.*parseInt*(StringHelper.*readString*());  
  
 *//illegal number* **if** (numberOfCommand<0||numberOfCommand>**currentComposite**.**menuItems**.size()){  
 **throw new** Exception();  
 }  
  
 MenuItem selectedMenuItem=**currentComposite**.**menuItems**.get(numberOfCommand-1);  
 **if** (selectedMenuItem.isComposite()){  
 *//This is composite menu item* **currentComposite**=selectedMenuItem;  
 }  
 *//implement composite pattern* selectedMenuItem.execute();  
 selectAndPerformOperation();  
  
 *//other errors* } **catch** (Exception e) {  
 System.***out***.println(**"Invalid item selected. Try again."**);  
 selectAndPerformOperation();  
 }  
  
 }  
  
 *//implement command pattern  
 //all menu items included command interface* @Override  
 **public void** buildMenuStructure(){  
  
 *//Adding root menu* **rootMenuItem**.**menuItems**.clear();  
 **rootMenuItem**.addMenuNode(**new** MenuItem(**"Change working directory"**,**controller**::setRootDirectory));  
 **rootMenuItem**.addMenuNode(**new** MenuItem(**"Show file list in working directory"**,**controller**::getFileList));  
  
 MenuItem businesOperation=**new** MenuItem(**"Business operation"**,**this**::showMenu);  
 **rootMenuItem**.addMenuNode(businesOperation);  
 **rootMenuItem**.addMenuNode(**new** MenuItem(**"Exit program"**,**controller**::exit));  
  
 *//Adding business operation menu* businesOperation.addMenuNode(**new** MenuItem(**"Add file"**, **controller**::addFile));  
 businesOperation.addMenuNode(**new** MenuItem(**"Delete file"**, **controller**::deleteFile));  
 businesOperation.addMenuNode(**new** MenuItem(**"Search file"**, **controller**::searchFile));  
 businesOperation.addMenuNode(**new** MenuItem(**"Return to previous menu"**, **this**::menuLevelUp));  
  
  
 }  
  
 *//implements observation* @Override  
 **public void** updateAddingState(String fileName, **boolean** isAdded) {  
 **if** (isAdded)  
 System.***out***.println(**"File "**+fileName+**" has been successfully added"**);  
 **else** System.***out***.println(**"Sorry. File "**+fileName+**" has not been added"**);  
 }  
  
 @Override  
 **public void** updateDeletingState(String fileName, **boolean** isDeleted) {  
  
 **if** (isDeleted)  
 System.***out***.println(**"File "**+fileName+**" has been successfully deleted"**);  
 **else** System.***out***.println(**"Sorry. File "**+fileName+**" has not been deleted"**);  
 }  
  
 @Override  
 **public void** updateFileList(List<File> fileList) {  
  
 System.***out***.println(**"Printing file list in this directory:"** + **model**.getRootPath()+**" in ascending order"**);  
  
 **for** (File file : fileList) {  
 System.***out***.println(file);  
 }  
  
 System.***out***.println(**"---------------------------------"**);  
  
 }  
  
 @Override  
 **public void** updateSearchingState(String fileName, **boolean** isFounded) {  
 **if** (isFounded)  
 System.***out***.println(**"File "**+fileName+**" has been successfully founded"**);  
 **else** System.***out***.println(**"File "**+fileName+**" has not been founded"**);  
 }  
  
  
}

Menu

**package** final01.menu;  
  
*//implement command pattern***public interface** CommandInterface {  
 **void** execute();  
}

**package** final01.menu;  
  
**import** java.util.ArrayList;  
  
*//implement composite pattern***public class** MenuItem **implements** CommandInterface {  
  
 **private** MenuItem **parentMenuItem**;  
 **private** String **commandName**;  
  
 **public** String getCommandName() {  
 **return commandName**;  
 }  
  
 **public** CommandInterface **command**;  
 **public** ArrayList<MenuItem> **menuItems**;  
  
  
  
 **public** MenuItem(String commandName, CommandInterface command) {  
 **this**.**commandName**=commandName;  
 **this**.**command**=command;  
 **menuItems**=**new** ArrayList<>();  
 }  
  
  
 **public boolean** isComposite(){  
 **return this**.**menuItems**.size()>0;  
 }  
  
 **public void** addMenuNode(MenuItem menuItem){  
 **menuItems**.add(menuItem);  
 menuItem.setParentMenuItem(**this**);  
 }  
  
  
 @Override  
 **public void** execute() {  
 **command**.execute();  
 }  
  
 **public void** setParentMenuItem(MenuItem parentMenuItem) {  
 **this**.**parentMenuItem** = parentMenuItem;  
 }  
  
 **public** MenuItem getParentMenuItem() {  
 **return parentMenuItem**;  
 }  
}

Util

**package** final01.util;  
  
**import** java.io.BufferedReader;  
**import** java.io.IOException;  
**import** java.io.InputStreamReader;  
  
  
*//one point to get access to input stream.***public class** StringHelper {  
  
 **private static** BufferedReader *bufferedReader*=**new** BufferedReader(**new** InputStreamReader(System.***in***));  
  
 **public static** String readString(){  
 String result;  
 **while** (**true**){  
 **try** {  
 result=*bufferedReader*.readLine();  
 **return** result;  
 } **catch** (IOException e) {  
 System.***out***.println(**"Error with reading string."**);  
 }  
 }  
 }  
  
  
}

Solution

**package** final01;  
  
**import** final01.controller.Controller;  
**import** final01.model.Model;  
**import** final01.view.View;  
  
**import** java.nio.file.Paths;  
  
**public class** Solution {  
  
 **public static void** main(String[] args) {  
 Model model=**new** Model(Paths.*get*(**"D:\\Test folder"**));  
 Controller controller=**new** Controller(model);  
 View view=**new** View(controller, model);  
  
 *//start* view.buildMenuStructure();  
 view.showMenu();  
 view.selectAndPerformOperation();  
 }  
  
}