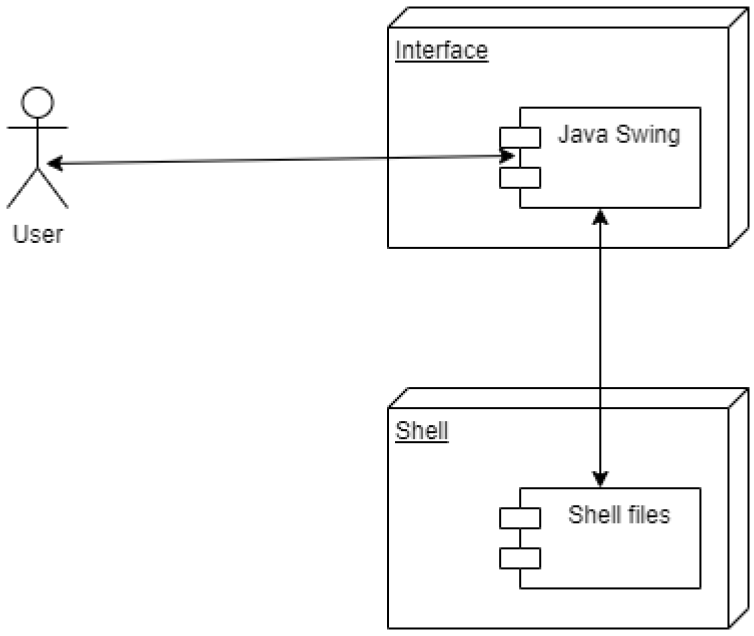
Лабораторна робота #3

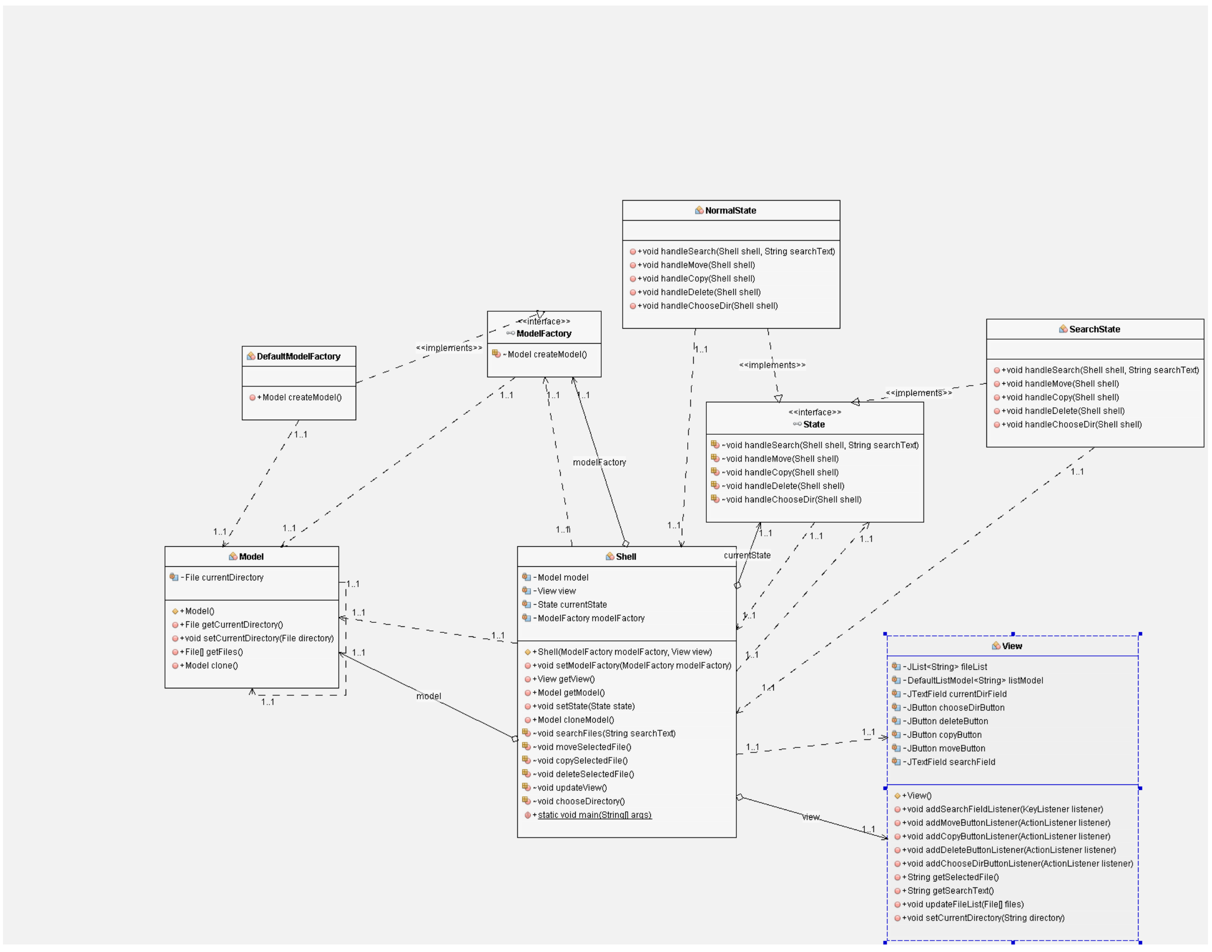
Тема: “#18. Shell (total commander)”

Студента групи ІА-14

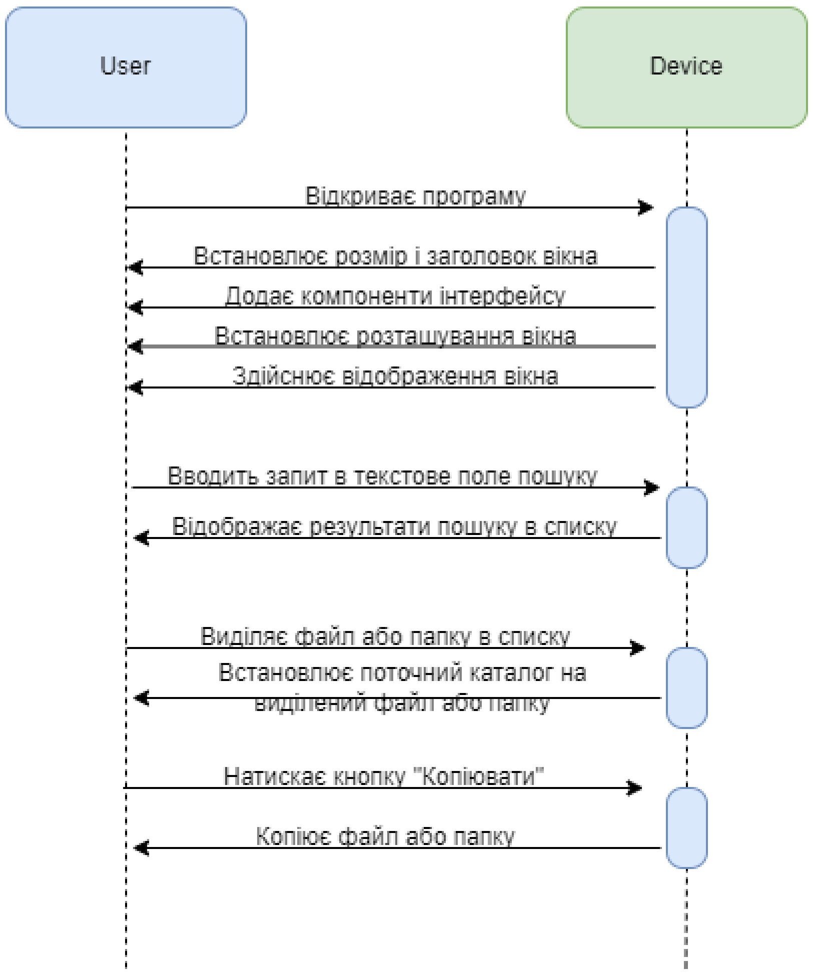
Розметова Т. Р.



Interface
Java Swing
User
Shell
Shell files



NormalState
+void handleSearch(Shell shell, String searchText)
+void handleMove(Shell shell)
+void handleCopy(Shell shell)
+void handleDelete(Shell shell)
+void handleChooseDir(Shell shell)
<<interface>>
ModelFactory
~Model createModel()
DefaultModelFactory
+Model createModel()
<<implements>>
SearchState
+void handleSearch(Shell shell, String searchText)
+void handleMove(Shell shell)
+void handleCopy(Shell shell)
+void handleDelete(Shell shell)
+void handleChooseDir(Shell shell)
<<implements>>
<<implements>>
<<interface>>
State
~void handleSearch(Shell shell, String searchText)
~void handleMove(Shell shell)
~void handleCopy(Shell shell)
~void handleDelete(Shell shell)
~void handleChooseDir(Shell shell)
modelFactory
currentState
Model
-File currentDirectory
+Model()
+File getCurrentDirectory()
+void setCurrentDirectory(File directory)
+File[] getFiles()
+Model clone()
Shell
-Model model
-View view
-State currentState
-ModelFactory modelFactory
+Shell(ModelFactory modelFactory, View view)
+void setModelFactory(ModelFactory modelFactory)
+View getView()
+Model getModel()
+void setState(State state)
+Model cloneModel()
~void searchFiles(String searchText)
~void moveSelectedFile()
~void copySelectedFile()
~void deleteSelectedFile()
~void updateView()
~void chooseDirectory()
+static void main(String[] args)
model
view
View
-JList<String> fileList
-DefaultListModel<String> listModel
-JTextField currentDirField
-JButton chooseDirButton
-JButton deleteButton
-JButton copyButton
-JButton moveButton
-JTextField searchField
+View()
+void addSearchFieldListener(KeyListener listener)
+void addMoveButtonListener(ActionListener listener)
+void addCopyButtonListener(ActionListener listener)
+void addDeleteButtonListener(ActionListener listener)
+void addChooseDirButtonListener(ActionListener listener)
+String getSelectedFile()
+String getSearchText()
+void updateFileList(File[] files)
+void setCurrentDirectory(String directory)
1..1


User
Device
Відкриває програму
Встановлює розмір і заголовок вікна
Додає компоненти інтерфейсу
Встановлює розташування вікна
Здійснює відображення вікна
Вводить запит в текстове поле пошуку
Відображає результати пошуку в списку
Виділяє файл або папку в списку
Встановлює поточний каталог на виділений файл або папку
Натискає кнопку "Копіювати"
Копіює файл або папку

Вихідні коди

```
package com.tyims.shell;



import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.io.File;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.StandardCopyOption;

import java.util.Arrays;

import java.util.stream.Collectors;

import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

import javax.swing.SwingUtilities;



public class Shell {

private Model model;

private View view;



public Shell(Model model, View view) {

this.model = model;

this.view = view;
```

```
view.addChooseDirButtonListener(new ChooseDirButtonListener());
view.addCopyButtonListener(new CopyButtonListener());
view.addDeleteButtonListener(new DeleteButtonListener());
view.addMoveButtonListener(new MoveButtonListener());
view.addSearchFieldListener(new SearchFieldListener());
updateView();
}
private class SearchFieldListener implements KeyListener {
@Override
public void keyTyped(KeyEvent e) {
// Не використовується
}

@Override
public void keyPressed(KeyEvent e) {
// Не використовується
}

@Override
public void keyReleased(KeyEvent e) {
searchFiles(view.getSearchText());
}
}
private class MoveButtonListener implements ActionListener {
@Override
```

```
public void actionPerformed(ActionEvent e) {
moveSelectedFile();
}
}
private class CopyButtonListener implements ActionListener {
@Override
public void actionPerformed(ActionEvent e) {
copySelectedFile();
}
}


private class DeleteButtonListener implements ActionListener {
@Override
public void actionPerformed(ActionEvent e) {
deleteSelectedFile();
}
}
private class ChooseDirButtonListener implements ActionListener {
@Override
public void actionPerformed(ActionEvent e) {
chooseDirectory();
}
}
private void searchFiles(String searchText) {
File[] files = model.getFiles();


if (!searchText.isEmpty()) {
```

```
files = Arrays.stream(files)
.filter(file -> file.getName().toLowerCase().contains(searchText.toLowerCase()))
.collect(Collectors.toList())
.toArray(new File[0]);
}

view.updateFileList(files);
}
private void moveSelectedFile() {
String selectedFileName = view.getSelectedFile();
if (selectedFileName != null) {
File sourceFile = new File(model.getCurrentDirectory(), selectedFileName);

JFileChooser fileChooser = new JFileChooser();
fileChooser.setDialogTitle("Select Destination Directory");
fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);

int result = fileChooser.showOpenDialog(view);
if (result == JFileChooser.APPROVE_OPTION) {
File destinationDir = fileChooser.getSelectedFile();
File destinationFile = new File(destinationDir, selectedFileName);

try {
Path sourcePath = sourceFile.toPath();
Path destinationPath = destinationFile.toPath();
Files.move(sourcePath, destinationPath, StandardCopyOption.REPLACE_EXISTING);
```

```
updateView();
} catch (IOException ex) {
JOptionPane.showMessageDialog(view, "Failed to move the file.", "Error", JOptionPane.ERROR_MESSAGE);
}
}
}
}
private void copySelectedFile() {
String selectedFileName = view.getSelectedFile();
if (selectedFileName != null) {
File sourceFile = new File(model.getCurrentDirectory(), selectedFileName);

JFileChooser fileChooser = new JFileChooser();
fileChooser.setDialogTitle("Select Destination Directory");
fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);

int result = fileChooser.showOpenDialog(view);
if (result == JFileChooser.APPROVE_OPTION) {
File destinationDir = fileChooser.getSelectedFile();
File destinationFile = new File(destinationDir, selectedFileName);

try {
Path sourcePath = sourceFile.toPath();
Path destinationPath = destinationFile.toPath();
Files.copy(sourcePath, destinationPath, StandardCopyOption.REPLACE_EXISTING);
updateView();
```

```
} catch (IOException ex) {

JOptionPane.showMessageDialog(view, "Failed to copy the file.", "Error", JOptionPane.ERROR_MESSAGE);

}

}

}

}

private void deleteSelectedFile() {

String selectedFileName = view.getSelectedFile();

if (selectedFileName != null) {

File fileToDelete = new File(model.getCurrentDirectory(), selectedFileName);

if (fileToDelete.exists()) {

int result = JOptionPane.showConfirmDialog(

view,

"Ви точно хочете це видалити?",

"Видалено",

JOptionPane.YES_NO_OPTION

);



if (result == JOptionPane.YES_OPTION) {

boolean deleted = fileToDelete.delete();

if (deleted) {

updateView();

} else {

JOptionPane.showMessageDialog(view, "Failed to delete the file.", "Error",
JOptionPane.ERROR_MESSAGE);

}

}

}
```

```
}
}
private void updateView() {
File[] files = model.getFiles();
view.updateFileList(files);
view.setCurrentDirectory(model.getCurrentDirectory().getAbsolutePath());
}

private void chooseDirectory() {
JFileChooser fileChooser = new JFileChooser();
fileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);

int result = fileChooser.showOpenDialog(view);
if (result == JFileChooser.APPROVE_OPTION) {
File selectedDir = fileChooser.getSelectedFile();
model.setCurrentDirectory(selectedDir);
updateView();
}
}

public static void main(String[] args) {
SwingUtilities.invokeLater(new Runnable() {
@Override
public void run() {
Model model = new Model();
View view = new View();
new Shell(model, view);
}
});
}
}


package com.tyims.shell;


import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyListener;
import java.io.File;


public class View extends JFrame{
private JList<String> fileList;
private DefaultListModel<String> listModel;
private JTextField currentDirField;
private JButton chooseDirButton;
private JButton deleteButton;
private JButton copyButton;
private JButton moveButton;
private JTextField searchField;


public View() {
```

```
super("Shell (Total Commander) TR");

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

setSize(600, 400);

setLocationRelativeTo(null);



listModel = new DefaultListModel<>();

fileList = new JList<>(listModel);

JScrollPane scrollPane = new JScrollPane(fileList);



currentDirField = new JTextField();

currentDirField.setEditable(false);



chooseDirButton = new JButton("Обрати папку");



JPanel topPanel = new JPanel(new BorderLayout());

topPanel.add(currentDirField, BorderLayout.CENTER);

topPanel.add(chooseDirButton, BorderLayout.EAST);



add(topPanel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

searchField = new JTextField();

searchField.setPreferredSize(new Dimension(150, 25));

searchField.setToolTipText("Search");

JPanel searchPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));

searchPanel.add(new JLabel("Search: "));
searchPanel.add(searchField);

setVisible(true);

deleteButton = new JButton("Видалити");

copyButton = new JButton("Копіювати");

moveButton = new JButton("Перенести");

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));

buttonPanel.add(deleteButton);

buttonPanel.add(copyButton);

buttonPanel.add(moveButton);

add(searchPanel, BorderLayout.EAST);

add(buttonPanel, BorderLayout.SOUTH);

}

public void addSearchFieldListener(KeyListener listener) {

searchField.addKeyListener(listener);

}



public void addMoveButtonListener(ActionListener listener) {

moveButton.addActionListener(listener);

}

public void addCopyButtonListener(ActionListener listener) {

copyButton.addActionListener(listener);

}



public void addDeleteButtonListener(ActionListener listener) {

deleteButton.addActionListener(listener);

}

public void addChooseDirButtonListener(ActionListener listener) {
```

```
chooseDirButton.addActionListener(listener);

}



public String getSelectedFile() {

return fileList.getSelectedValue();

}

public String getSearchText() {

return searchField.getText();

}



public void updateFileList(File[] files) {

listModel.clear();

for (File file : files) {

listModel.addElement(file.getName());

}

}



public void setCurrentDirectory(String directory) {

currentDirField.setText(directory);

}

}



package com.tyims.shell;



import java.io.File;
```

```
public class Model {

private File currentDirectory;



public Model() {

setCurrentDirectory(new File("."));

}



public File getCurrentDirectory() {

return currentDirectory;

}



public void setCurrentDirectory(File directory) {

this.currentDirectory = directory;

}



public File[] getFiles() {

return currentDirectory.listFiles();

}

}
```