Міністерство освіти і науки України

Черкаський державний технологічний університет

Кафедра програмного забезпечення автоматизованих систем

Звіт

З лабораторної роботи №8

З предмету «Об’єктно-орієнтоване програмування»

Перевірив

Викладач кафедри

Програмного забезпечення автоматизованих систем

Крайовий В. М.

Виконав

Студент І курсу

Група ПЗ-154

Кравченко Артем Олегович

Виконав

Студент ІІ курсу

Група ПЗ-154

Мороз Б.В.

Черкаси 2016

**Тема:** Композиція.

**Мета:** Зрозуміти принципи та призначення композиції та навчитись використовувати її при побудові класів.

**Постановка завдання:** В даній роботі будуть використовуватись результати попередніх робіт.

1. Взяти один клас з попередніх робіт та розробити другий клас відповідно до варіанту. При розробці другого класу використати принцип композиції, зробивший його полем об'єкт або масив чи колекцію об’єктів першого класу.

2. Розробити третій клас, в якому реалізувати інтерфейс роботи з об’єктом другого класу. В ньому реалізувати функції:

- відображення даного об’єкту; (вивести масив)\*

- додавання до нього об’єкту першого класу; (додати елемент в масив)\*

- проведення дій над об’єктом першого класу як частини другого класу. (відняти елемент з масиву)\*

**Індивідуальне завдання:** Олівець та пенал.

**Хід роботи:**

**Лістинг програми:**

**Main**

package com.company;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class Main {

static WritingTool objWritingTool;

static Pencil object\_0;

static Pen object\_1;

public static void main(String[] args) {

JFrame mainFrame = buildMainFrame();

mainFrame.setVisible(true);

}

private static JFrame buildMainFrame() {

JFrame mainFrame = initMainFrame();

JTabbedPane tabbedPane = buildTabbledPane();

mainFrame.add(tabbedPane);

return mainFrame;

}

private static JTabbedPane buildTabbledPane() {

JTabbedPane tabbedPane = new JTabbedPane();

JPanel mainPanel = buildMainPanel(tabbedPane);

tabbedPane.add("Main", mainPanel);

return tabbedPane;

}

private static JPanel buildMainPanel(JTabbedPane tabbedPane) {

JPanel mainPanel = initMainPanel();

JPanel functionalPanel = buildFunctionalPanel(tabbedPane);

mainPanel.add(functionalPanel);

return mainPanel;

}

private static JPanel buildFunctionalPanel(JTabbedPane tabbedPane) {

JPanel functionalPanel = initFunctionalPanel();

PencilCase pencilCase = new PencilCase();

JComboBox comboBoxClass = initComboBoxClass();

JComboBox comboBoxCase = initComboBoxCase();

JButton btnTake = buildBtnTake(tabbedPane, comboBoxCase, pencilCase);

JButton btnRemoveItem = buildBtnRemoveItem(comboBoxCase, pencilCase, btnTake, tabbedPane);

JButton btnRandom = buildBtnAddItem(comboBoxClass, comboBoxCase, btnRemoveItem, btnTake);

functionalPanel.add(btnRemoveItem);

functionalPanel.add(btnTake);

functionalPanel.add(comboBoxCase);

functionalPanel.add(btnRandom);

functionalPanel.add(comboBoxClass);

return functionalPanel;

}

private static JFrame initMainFrame() {

JFrame mainFrame = new JFrame("Choose your class");

mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

mainFrame.setSize(500, 400);

return mainFrame;

}

private static JPanel initFunctionalPanel() {

JPanel functionalPanel = new JPanel();

functionalPanel.setLayout(new GridLayout(5,1));

return functionalPanel;

}

private static JPanel initMainPanel() {

JPanel mainPanel = new JPanel();

mainPanel.setLayout(new FlowLayout());

return mainPanel;

}

private static JComboBox initComboBoxClass() {

String[] classItems = {

"Pencil",

"Pen"

};

JComboBox comboBoxClass = new JComboBox(classItems);

comboBoxClass.setPreferredSize(new Dimension(215,25));

return comboBoxClass;

}

private static JComboBox initComboBoxCase() {

JComboBox comboBoxClass = new JComboBox();

comboBoxClass.setPreferredSize(new Dimension(215,25));

return comboBoxClass;

}

public static void deleteTab(JTabbedPane tabbedPane) {

if (tabbedPane.indexOfTab("com.company.Pencil") == 1 || tabbedPane.indexOfTab("com.company.Pen") == 1)

tabbedPane.remove(1);

}

private static JButton buildBtnRemoveItem(JComboBox comboBoxCase, PencilCase pencilCase, JButton btnTake, JTabbedPane tabbedPane) {

JButton btnRemoveItem = new JButton("Remove current item");

btnRemoveItem.setEnabled(false);

btnRemoveItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

pencilCase.removeElement(comboBoxCase.getSelectedIndex());

comboBoxCase.removeItemAt(comboBoxCase.getSelectedIndex());

EnableBtn(comboBoxCase, btnRemoveItem);

EnableBtn(comboBoxCase, btnTake);

deleteTab(tabbedPane);

}

});

return btnRemoveItem;

}

private static JButton buildBtnTake(JTabbedPane tabbedPane, JComboBox comboBoxCase, PencilCase pencilCase) {

JButton btnTake = new JButton("Take");

btnTake.setEnabled(false);

btnTake.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

switch (comboBoxCase.getSelectedItem().toString()) {

case "com.company.Pencil" :

objWritingTool = (Pencil) pencilCase.getObject(comboBoxCase.getSelectedIndex());

break;

case "com.company.Pen" :

objWritingTool = (Pen) pencilCase.getObject(comboBoxCase.getSelectedIndex());

break;

}

try {

objWritingTool = objWritingTool.getClass().newInstance();

} catch (InstantiationException e1) {

e1.printStackTrace();

} catch (IllegalAccessException e1) {

e1.printStackTrace();

}

deleteTab(tabbedPane);

JPanel panelTab;

panelTab = objWritingTool.buildMainPanel();

objWritingTool.changeImage();

tabbedPane.add(comboBoxCase.getSelectedItem().toString(), panelTab);

}

});

return btnTake;

}

private static JButton buildBtnAddItem(final JComboBox comboBoxClass, final JComboBox comboBoxCase, JButton btnRemoveItem, JButton btnTake) {

JButton btnAddItem = new JButton("Add item");

PencilCase pencilCase = new PencilCase();

btnAddItem.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

switch (comboBoxClass.getSelectedItem().toString()) {

case "Pencil" :

objWritingTool = new Pencil();

break;

case "Pen" :

objWritingTool = new Pen();

break;

default:

throw new IllegalStateException("Incorrect type");

}

pencilCase.addElement(objWritingTool);

comboBoxCase.removeAllItems();

for (int index=0; index< pencilCase.listclass.size(); index++) {

comboBoxCase.addItem(pencilCase.listclass.get(index).getClass().getName());

}

comboBoxCase.setSelectedIndex(0);

EnableBtn(comboBoxCase, btnRemoveItem);

EnableBtn(comboBoxCase, btnTake);

}

});

return btnAddItem;

}

private static void EnableBtn(JComboBox comboBoxCase, JButton button) {

if (comboBoxCase.getItemCount() > 0)

button.setEnabled(true);

else

button.setEnabled(false);

}

}

**Pen**

package com.company;

public class Pen extends WritingTool {

public Pen() {

option = "Medium";

};

@Override

public String toStr() {

String stringResult;

String stringWritingTool = super.toStr();

String stringPen =

"\n" +

"Thickness : " +

getOption().toString() +

"\n" +

"Ink : " +

Boolean.toString(getChecker1()) +

"\n" +

"On : " +

Boolean.toString(getChecker2()) +

"\n";

stringResult = stringWritingTool + stringPen;

return stringResult;

};

@Override

public String[] itemsKind() {

String[] items = {

"Low",

"Medium",

"High"

};

return items;

}

@Override

public String nameCheckBox\_1() {

String pole = "Ink";

return pole;

};

@Override

public String nameCheckBox\_2() {

String pole = "On";

return pole;

};

@Override

public String labelText(){

String pole = "Thickness";

return pole;

};

}

**Pencil**

package com.company;

public class Pencil extends WritingTool {

public Pencil() {

option = "HB";

}

@Override

public String toStr(){

String stringResult;

String stringWritingTool = super.toStr();

String stringPencil =

"\n" +

"Softness : " +

getOption().toString() +

"\n" +

"Sharpened : " +

Boolean.toString(getChecker1()) +

"\n" +

"Eraser : " +

Boolean.toString(getChecker2()) +

"\n";

stringResult = stringWritingTool + stringPencil;

return stringResult;

};

@Override

public String[] itemsKind() {

String[] items = {

"B3",

"B2",

"B1",

"HB",

"H1",

"H2",

"H3"

};

return items;

}

@Override

public String nameCheckBox\_1() {

String pole = "Sharpened";

return pole;

};

@Override

public String nameCheckBox\_2() {

String pole = "Eraser";

return pole;

};

@Override

public String labelText(){

String pole = "Softness";

return pole;

};

}

**WritingTool**

package com.company;

import javax.imageio.ImageIO;

import javax.swing.\*;

import javax.swing.event.DocumentEvent;

import javax.swing.event.DocumentListener;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.ItemEvent;

import java.awt.event.ItemListener;

import java.awt.Graphics;

import java.awt.Image;

import java.io.File;

import java.io.IOException;

public class WritingTool extends JPanel{

public int lenght;

public String material;

public String color;

public String option;

public boolean checker1;

public boolean checker2;

public boolean imgLoaded;

public Image image;

public WritingTool() {

material = "Wood";

lenght = 10;

color = "Black";

checker1 = false;

checker2 = false;

loadImage();

imgLoaded = false;

};

public String toStr(){

return

"Material : " +

getMaterial() +

"\n" +

"Color : " +

getColor() +

"\n" +

"Lenght : " +

getLenght() +

"cm";

};

String[] itemsMaterial = {

"Wood",

"Rubber",

"Plastic"

};

String[] itemsColor = {

"Black",

"Blue",

"Red",

"Green",

"Violet"

};

public String[] itemsKind() {

String[] items = {

"None"

};

return items;

}

public String nameCheckBox\_1(){

String pole = "";

return pole;

};

public String nameCheckBox\_2(){

String pole = "";

return pole;

};

public String labelText(){

String pole = "";

return pole;

};

public void strToQR(String str)

{

QRCodeGenerator makeQR = new QRCodeGenerator();

makeQR.codeGen(str);

loadImage();

repaint();

}

public void loadImage() {

if (imgLoaded == false) {

try {

image = ImageIO.read(new File("Clear.jpg"));

} catch (IOException e) {

e.printStackTrace();

}

imgLoaded = true;

}

else {

try {

image = ImageIO.read(new File("Image.jpg"));

} catch (IOException e) {

e.printStackTrace();

}

}

}

public void paint(Graphics g) {

g.drawImage(image, 0, 0, 256, 256, null);

}

public void changeImage() {

loadImage();

repaint();

}

public JPanel buildMainPanel() {

JPanel mainPanel = initMainPanel();

JPanel panel = buildPanel();

JPanel imgPanel = buildImgPanel();

mainPanel.add(panel);

mainPanel.add(imgPanel);

return mainPanel;

};

private JPanel initMainPanel() {

JPanel mainPanel = new JPanel();

mainPanel.setLayout(new FlowLayout());

return mainPanel;

}

private JPanel initComboBoxPanel() {

JPanel comboBoxPanel = new JPanel();

comboBoxPanel.setLayout(new GridLayout(2,3));

return comboBoxPanel;

}

private JPanel initPanel() {

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(2,2));

panel.setPreferredSize(new Dimension(490,100));

return panel;

}

private JPanel initCheckBoxPanel() {

JPanel checkBoxPanel = new JPanel();

checkBoxPanel.setLayout(new GridLayout(2,1));

return checkBoxPanel;

}

private JPanel initImgPanel() {

JPanel imgPanel = new JPanel();

imgPanel.setLayout(new GridLayout(1, 2, 3, 3));

imgPanel.setPreferredSize(new Dimension(490,256));

return imgPanel;

}

private JPanel initTextFieldPanel() {

JPanel textFieldPanel = new JPanel();

textFieldPanel.setLayout(new GridLayout(2,1));

return textFieldPanel;

}

private JPanel buildComboBoxPanel() {

JPanel comboBoxPanel = initComboBoxPanel();

JLabel labelExt = new JLabel(labelText());

JLabel labelMaterial = new JLabel("Material");

JLabel labelColor = new JLabel("Color");

JComboBox comboBoxMaterial = new JComboBox(this.itemsMaterial);

JComboBox comboBoxColor = new JComboBox(this.itemsColor);

JComboBox comboBoxOption = new JComboBox(itemsKind());

comboBoxPanel.add(labelMaterial);

comboBoxPanel.add(labelColor);

comboBoxPanel.add(labelExt);

comboBoxPanel.add(comboBoxMaterial);

comboBoxPanel.add(comboBoxColor);

comboBoxPanel.add(comboBoxOption);

ActionListener actionListenerMaterial = new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

setMaterial(comboBoxMaterial.getSelectedItem().toString());

}

};

comboBoxMaterial.addActionListener(actionListenerMaterial);

ActionListener actionListenerColor = new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

setColor(comboBoxColor.getSelectedItem().toString());

}

};

comboBoxColor.addActionListener(actionListenerColor);

ActionListener actionListenerOption = new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

setOption(comboBoxOption.getSelectedItem().toString());

}

};

comboBoxOption.addActionListener(actionListenerOption);

comboBoxOption.setSelectedItem(option);

return comboBoxPanel;

}

private JPanel buildCheckBoxPanel() {

JPanel checkBoxPanel = initCheckBoxPanel();

JCheckBox checkBox\_1 = new JCheckBox(nameCheckBox\_1());

JCheckBox checkBox\_2 = new JCheckBox(nameCheckBox\_2());

checkBoxPanel.add(checkBox\_1);

checkBoxPanel.add(checkBox\_2);

checkBox\_1.addItemListener(new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

setChecker1(e.getStateChange() == ItemEvent.SELECTED);

}

});

checkBox\_2.addItemListener(new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

setChecker2(e.getStateChange() == ItemEvent.SELECTED);

}

});

return checkBoxPanel;

}

private JButton buildBtnSetLenght(JTextField textField) {

JButton btnSetLenght = new JButton("Set lenght");

btnSetLenght.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e)

{

setLenght(Integer.parseInt(textField.getText()));

}

});

btnSetLenght.setEnabled(false);

return btnSetLenght;

}

private JTextField buildTextField() {

JTextField textField = new JTextField();

JButton btnSetLenght = buildBtnSetLenght(textField);

textField.getDocument().addDocumentListener(new DocumentListener() {

@Override

public void insertUpdate(DocumentEvent e) {

if (textField.getText().equals("") || textField.getText().equals(null)){

btnSetLenght.setEnabled(false);

}

else{

btnSetLenght.setEnabled(true);

}

}

@Override

public void removeUpdate(DocumentEvent e) {

if (textField.getText().equals("") || textField.getText().equals(null)){

btnSetLenght.setEnabled(false);

}

else{

btnSetLenght.setEnabled(true);

}

}

@Override

public void changedUpdate(DocumentEvent e) {}

});

return textField;

}

private JPanel buildTextFieldPanel() {

JPanel textFieldPanel = initTextFieldPanel();

JTextField textField = buildTextField();

JButton btnSetLenght = buildBtnSetLenght(textField);

textFieldPanel.add(textField);

textFieldPanel.add(btnSetLenght);

return textFieldPanel;

}

JTextArea textArea = new JTextArea();

private JButton buildBtnShowAllParameters() {

JButton btnShowAllParameters = new JButton("Show all parameters");

btnShowAllParameters.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

textArea.setText(toStr());

strToQR(toStr());

}

});

return btnShowAllParameters;

}

private JPanel buildPanel() {

JPanel panel = initPanel();

JPanel comboBoxPanel = buildComboBoxPanel();

JPanel checkBoxPanel = buildCheckBoxPanel();

JPanel textFieldPanel = buildTextFieldPanel();

JButton btnShowAllParameters = buildBtnShowAllParameters();

panel.add(comboBoxPanel);

panel.add(checkBoxPanel);

panel.add(textFieldPanel);

panel.add(btnShowAllParameters);

return panel;

}

private JPanel buildImgPanel() {

JPanel imgPanel = initImgPanel();

imgPanel.add(textArea);

imgPanel.add(this);

return imgPanel;

}

public void setChecker1(boolean checker1){

this.checker1 = checker1;

}

public boolean getChecker1(){

return checker1;

}

public void setChecker2(boolean checker2){

this.checker2 = checker2;

}

public boolean getChecker2(){

return checker2;

}

public void setColor(String color){

this.color = color;

}

public String getColor() {

return color;

}

public void setOption(String option){

this.option = option;

}

public String getOption() {

return option;

}

public void setMaterial(String material){

this.material = material;

}

public String getMaterial() {

return material;

}

public void setLenght(int lenght) { this.lenght = lenght; }

public int getLenght() { return lenght; }

}

**PencilCase**

package com.company;

import java.util.ArrayList;

import java.util.List;

public class PencilCase {

public static List listclass = new ArrayList();

public static Object getObject(int index) {

Object object = listclass.get(index);

try {

object.getClass().newInstance();

} catch (InstantiationException e) {

e.printStackTrace();

} catch (IllegalAccessException e) {

e.printStackTrace();

}

return object;

}

public static void removeElement(int index) {

listclass.remove(index);

}

public static void addElement(Object var) {

listclass.add(var);

}

}

**QRCodeGenerator**

package com.company;

import com.google.zxing.BarcodeFormat;

import com.google.zxing.WriterException;

import com.google.zxing.common.BitMatrix;

import com.google.zxing.qrcode.QRCodeWriter;

import javax.imageio.ImageIO;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

public class QRCodeGenerator {

public static void codeGen(String text) {

QRCodeWriter writer = new QRCodeWriter();

int width = 256, height = 256;

BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB); // create an empty image

int white = 255 << 16 | 255 << 8 | 255;

int black = 0;

try {

BitMatrix bitMatrix = writer.encode(text.toString(), BarcodeFormat.QR\_CODE, width, height);

for (int i = 0; i < width; i++) {

for (int j = 0; j < height; j++) {

image.setRGB(i, j, bitMatrix.get(i, j) ? black : white); // set pixel one by one

}

}

try {

ImageIO.write(image, "jpg", new File("BM.jpg")); // save QR image to disk

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

} catch (WriterException e) {

// TODO Auto-generated catch block

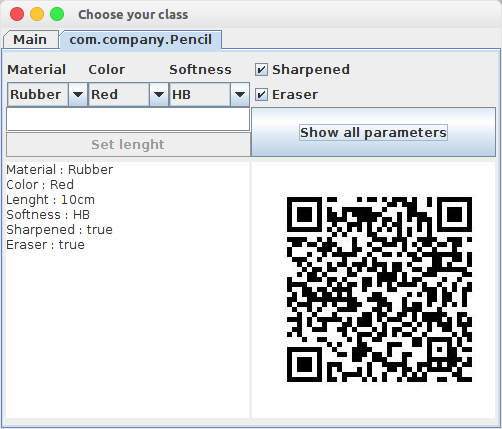
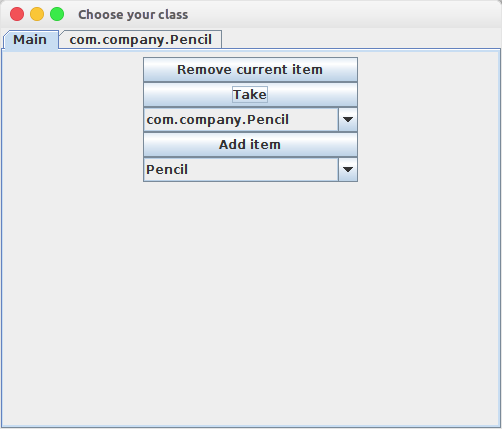
e.printStackTrace();

}

}

}

**Результат роботи програми:**



**Висновок:** У ході виконання лабораторної роботи, я зрозумів принципи та призначення композиції та навчився використовувати її при побудові класів.