Міністерство освіти і науки України

Черкаський державний технологічний університет

Кафедра програмного забезпечення автоматизованих систем

Звіт

З лабораторної роботи №7

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:** Колекції. RTTI.

**Мета:** Зрозуміти призначення колекцій та навчитись їх використовувати. Зрозуміти призначення RTTI та навчитись його використовувати.

**Постановка завдання:** В даній роботі будуть використовуватись два класи, розроблені в попередніх лабораторних роботах.

Розробити програму, яка:

1. Створює колекцію об’єктів (JPanel та двох власних класів попередніх лабораторних робіт), причому об’єкти вносяться в колекцію у випадковій послідовності (використати генератор випадкових чисел).
2. Відображає колекцію на екрані з можливістю вибору поточного елементу.
3. Відображає тип поточного об’єкту, беручи інформацію з об’єкту типу Class для даного об’єкту.
4. Забезпечує можливість виконання самостійно розроблених методів поточного об’єкту.

**Індивідуальне завдання:** Пристрій для писання та Олівець.

**Хід роботи:**

**Лістинг програми:**

**Main**

package com.company;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.util.\*;

import java.util.List;

public class Main {

static WritingTool objWritingTool;

static JPanel object\_0;

static Pen object\_1;

static Pencil object\_2;

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();

JLabel labelClass = new JLabel("Class:");

List listClass = new ArrayList(3);

JComboBox comboBoxClass = buildComboBoxClass(tabbedPane, labelClass, listClass);

JButton btnRandom = buildBtnRandom(comboBoxClass, listClass);

functionalPanel.add(labelClass);

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(3,1));

return functionalPanel;

}

private static JPanel initMainPanel() {

JPanel mainPanel = new JPanel();

mainPanel.setLayout(new FlowLayout());

return mainPanel;

}

private static JComboBox buildComboBoxClass(final JTabbedPane tabbedPane, final JLabel labelClass, final List listClass) {

JComboBox comboBoxClass = new JComboBox();

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

ActionListener actionListenerLabel = new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

boolean isObject = false;

labelClass.setText(null);

labelClass.setText("Class: " + (String) comboBoxClass.getSelectedItem().toString());

if (tabbedPane.indexOfTab("Class object") == 1)

tabbedPane.remove(1);

WritingTool oldWritingTool = null;

WritingTool newWritingTool = null;

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

case "javax.swing.JPanel" :

break;

case "com.company.Pen":

oldWritingTool = object\_1;

object\_1 = new Pen();

newWritingTool = object\_1;

break;

case "com.company.Pencil":

oldWritingTool = object\_2;

object\_2 = new Pencil();

newWritingTool = object\_2;

break;

default:

throw new IllegalStateException("Incorrect type");

//break;

}

if (oldWritingTool != null) {

int index = listClass.indexOf(oldWritingTool);

listClass.remove(oldWritingTool);

listClass.add(index, newWritingTool);

objWritingTool = newWritingTool;

objWritingTool.changeImage();

isObject = true;

}

JPanel panelTab = new JPanel();

if (isObject) {

panelTab = objWritingTool.buildMainPanel();

}

else {

JButton btnExit = new JButton("Exit");

btnExit.setPreferredSize(new Dimension(300,30));

btnExit.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

tabbedPane.remove(1);

}

});

panelTab.add(btnExit);

}

tabbedPane.addTab("Class object", panelTab);

}

};

comboBoxClass.addActionListener(actionListenerLabel);

return comboBoxClass;

}

private static JButton buildBtnRandom(final JComboBox comboBoxClass, final List listClass) {

JButton btnRandom = new JButton("Create and show collection");

btnRandom.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

List<String> listRandom = new ArrayList(3);

Random random = new Random();

String randomNumber;

int indexCounter = 0; // Index\*

btnRandom.setEnabled(false);

for (int i = 0; i < 3; i++) {

listRandom.add(i, Integer.toString(i));

}

while (listRandom.size()>0)

{

randomNumber = Integer.toString(random.nextInt(3));

for (int i = 0; i < listRandom.size(); i++) {

if (listRandom.get(i).equals(randomNumber)) {

listRandom.remove(randomNumber);

switch (randomNumber) {

case "0" :

object\_0 = new JPanel();

listClass.add(indexCounter, object\_0);

break;

case "1" :

object\_1 = new Pen();

listClass.add(indexCounter, object\_1);

break;

case "2" :

object\_2 = new Pencil();

listClass.add(indexCounter, object\_2);

break;

}

indexCounter++;

}

}

}

for (int i = 0; i<3; i++) {

comboBoxClass.addItem(listClass.get(i).getClass().getName());

}

}

});

return btnRandom;

}

}

**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; }

}

**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

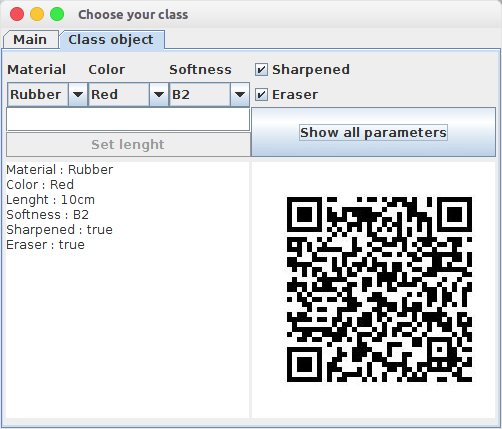
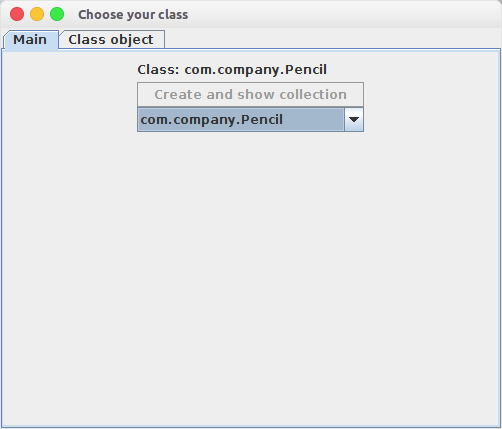
e.printStackTrace();

}

}

}

**Результат роботи програми:**



**Висновок:** У ході виконання лабораторної роботи, я зрозумів призначення колекцій та навчився їх використовувати. Зрозумів призначення RTTI та навчився його використовувати.