Міністерство освіти і науки України

Черкаський державний технологічний університет

Кафедра програмного забезпечення автоматизованих систем

Звіт

З лабораторної роботи №6

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:** Поліморфізм.

**Мета:** Зрозуміти принцип поліморфізму. Навчитись використовувати поліморфізм при створенні об'єктно-орієнтованих програм.

**Постановка завдання:** Взяти програму, розроблену в попередній лабораторній роботі ("Створення ієрархії класів. Спадкування. Перевизначення методів"). До двох розроблених класів додати по два методи: перший приймає групу параметрів, взятих з елементів введення інтерфейсу користувача (наприклад, текстових полів) та змінює значення всіх полів класу; другий повертає рядок зі значеннями всіх полів класу (методи повинні бути перевизначені в класі-нащадку, кількість параметрів повинна бути розумною – не більше кількох).

Додати до вікна інтерфейсу програми дві кнопки; при натисненні першої кнопки повинен виконуватись перший метод поточного створеного об’єкту, при цьому береться інформація з текстових полів та інших відповідних елементів інтерфейсу та записується в поля об’єкту; при натисненні другої кнопки – виконується другий метод і повернене значення записується у текстову область. Зробити два варіанти третього класу (запуску програми та інтерфейсу користувача): перший – щоб було два вказівники на класи відповідно до варіанту; другий – так, щоб був лише вказівник на базовий клас з двох розроблених відповідно до варіанту і при натисненні кнопки створення об’єкту об’єкт кожного з двох класів присвоювався саме цьому вказівникові. Порівняти отриманий результат.

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

**Хід роботи:**

**Лістинг програми:**

**Main**

package com.company;

import javax.swing.\*;

import javax.swing.event.DocumentEvent;

import javax.swing.event.DocumentListener;

import java.awt.\*;

import java.awt.event.\*;

public class Main {

static WritingTool fi;

static boolean isFrame = false;

public static void main(String[] args) {

String[] classString = {

"Pen",

"Pencil"

};

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

JLabel labelMaterial = new JLabel("Material");

JLabel labelForm = new JLabel("Color");

JButton btn1 = new JButton("Show all parameters");

JButton btn2 = new JButton("Set lenght");

JComboBox comboBoxClass = new JComboBox(classString);

JTextArea textArea = new JTextArea();

JTextField textField = new JTextField();

JPanel panel = new JPanel();

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

frame.setSize(500, 300);

frame.setVisible(true);

frame.add(panel, BorderLayout.NORTH);

panel.setLayout(new GridLayout(1,1));

panel.add(comboBoxClass);

ActionListener actionListenerComboBoxClass = new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JFrame frame2;

textArea.setText("");

String varFrame;

if(isFrame == true){

fi.destructForm();

};

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

case "Pen":

fi = new Pen();

varFrame = "Pen";

fi.toString();

fi.loadImage();

fi.repaint();

break;

case "Pencil":

fi = new Pencil();

varFrame = "Pencil";

fi.toString();

fi.loadImage();

fi.repaint();

break;

default:

throw new IllegalStateException("Incorrect writing tool type");

//break;

}

isFrame = true;

frame2 = new JFrame(varFrame);

fi.buildForm(frame2, labelMaterial, labelForm, btn1, btn2, textArea, textField);

}

};

comboBoxClass.addActionListener(actionListenerComboBoxClass);

btn1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

textArea.setText(fi.toString());

fi.strToQR(fi.toString());

// btn2.set

}

});

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

@Override

public void insertUpdate(DocumentEvent e) {

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

btn2.setEnabled(false);

}

else{

btn2.setEnabled(true);

}

}

@Override

public void removeUpdate(DocumentEvent e) {

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

btn2.setEnabled(false);

}

else{

btn2.setEnabled(true);

}

}

@Override

public void changedUpdate(DocumentEvent e) {}

});

btn2.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e)

{

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

}

});

btn2.setEnabled(false);

};

}

**Pen**

package com.company;

import java.awt.\*;

public class Pen extends WritingTool {

public Pen() {

option = "Medium";

};

@Override

public String[] ik() {

String[] itemsKind = {

"Low",

"Medium",

"High"

};

return itemsKind;

}

@Override

public void getAllParam() {

color = this.getColor();

option = this.getOption();

material = this.getMaterial();

checker1 = this.getChecker1();

checker2 = this.getChecker2();

lenght = this.getLenght();

};

@Override

public String varName1() {

String pole = "Ink";

return pole;

};

@Override

public String varName2() {

String pole = "On";

return pole;

};

@Override

public String labelText(){

String pole = "Thickness";

return pole;

};

@Override

public String toString() {

String str;

String str1 = super.toString();

String str2 =

"\n" +

"Thickness : " +

option +

"\n" +

"Ink : " +

checker1 +

"\n" +

"On : " +

checker2 +

"\n";

str = str1 + str2;

return str;

};

}

**Pencil**

package com.company;

import java.awt.\*;

public class Pencil extends WritingTool {

public Pencil() {

option = "HB";

}

@Override

public void getAllParam() {

color = this.getColor();

option = this.getOption();

material = this.getMaterial();

checker1 = this.getChecker1();

checker2 = this.getChecker2();

lenght = this.getLenght();

};

@Override

public String[] ik() {

String[] itemsKind = {

"B3",

"B2",

"B1",

"HB",

"H1",

"H2",

"H3"

};

return itemsKind;

}

@Override

public String varName1() {

String pole = "Sharpened";

return pole;

};

@Override

public String varName2() {

String pole = "Eraser";

return pole;

};

@Override

public String labelText(){

String pole = "Softness";

return pole;

};

@Override

public String toString(){

String str;

String str1 = super.toString();

String str2 =

"\n" +

"Softness : " +

option +

"\n" +

"Sharpened : " +

checker1 +

"\n" +

"Eraser : " +

checker2 +

"\n";

str = str1 + str2;

return str;

};

@Override

public void paint(Graphics g) {

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

}

}

**WritingTool**

package com.company;

import javax.imageio.ImageIO;

import javax.swing.\*;

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 String material;

public int lenght;

public String color;

public String option;

public boolean checker1;

public boolean checker2;

public JFrame mainFrame;

public boolean imgLoaded;

public WritingTool() {

material = "Wood";

lenght = 10;

color = "Black";

checker1 = false;

checker2 = false;

loadImage();

imgLoaded = false;

};

public void getAllParam() {

color = this.getColor();

option = this.getOption();

material = this.getMaterial();

checker1 = this.getChecker1();

checker2 = this.getChecker2();

lenght = this.getLenght();

};

String[] itemsMaterial = {

"Wood",

"Rubber",

"Plastic"

};

String[] itemsColor = {

"Black",

"Blue",

"Red",

"Green",

"Violet"

};

public String[] ik() {

String[] itemsKind = {

"None"

};

return itemsKind;

}

public String varName1(){

String pole = "";

return pole;

};

public String varName2(){

String pole = "";

return pole;

};

public String labelText(){

String pole = "";

return pole;

};

public Image image;

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 buildForm(JFrame frame, JLabel labelMaterial, JLabel labelColor, JButton btn1, JButton btn2, JTextArea textArea, JTextField textField) {

mainFrame = frame;

JPanel panel = new JPanel();

JPanel comboBoxPanel = new JPanel();

JPanel checkBoxPanel = new JPanel();

JPanel textFieldPanel = new JPanel();

JPanel imgPanel = new JPanel();

JCheckBox checkBox1 = new JCheckBox(varName1());

JCheckBox checkBox2 = new JCheckBox(varName2());

JLabel labelExt = new JLabel(labelText());

JComboBox comboBoxMaterial = new JComboBox(this.itemsMaterial);

JComboBox comboBoxColor = new JComboBox(this.itemsColor);

JComboBox comboBoxOption = new JComboBox(ik());

JButton btnGetAllParam = new JButton("GetAllParam");

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

mainFrame.setSize(500, 355);

mainFrame.setVisible(true);

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

imgPanel.add(textArea);

imgPanel.add(this);

mainFrame.add(panel, BorderLayout.NORTH);

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

panel.add(comboBoxPanel);

panel.add(checkBoxPanel);

panel.add(textFieldPanel);

panel.add(btn1);

mainFrame.add(imgPanel, BorderLayout.CENTER);

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

textFieldPanel.add(textField);

textFieldPanel.add(btn2);

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

comboBoxPanel.add(labelMaterial);

comboBoxPanel.add(labelColor);

comboBoxPanel.add(labelExt);

comboBoxPanel.add(comboBoxMaterial);

comboBoxPanel.add(comboBoxColor);

comboBoxPanel.add(comboBoxOption);

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

checkBoxPanel.add(checkBox1);

checkBoxPanel.add(checkBox2);

checkBoxPanel.add(btnGetAllParam);

btnGetAllParam.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

getAllParam();

}

});

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 actionListenerTC = new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

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

}

};

comboBoxOption.addActionListener(actionListenerTC);

comboBoxOption.setSelectedItem(option);

checkBox1.addItemListener(new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

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

}

});

checkBox2.addItemListener(new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

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

}

});

};

public void destructForm() {

mainFrame.dispose();

mainFrame.setVisible(false);

};

public String toString(){

return

"Material : " +

material +

"\n" +

"Color : " +

color +

"\n" +

"Lenght : " +

lenght +

"cm";

};

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

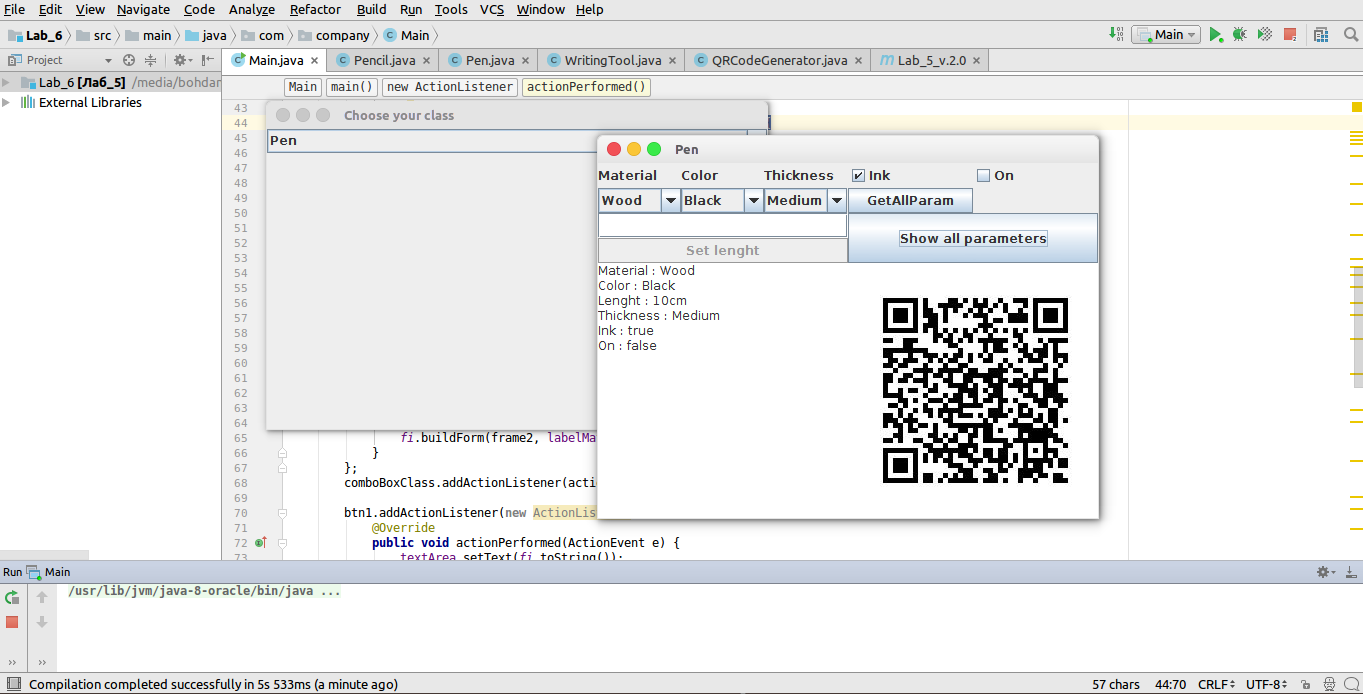
e.printStackTrace();

}

}

}

**Результат роботи програми:**



**Висновок:** У ході виконання лабораторної роботи, я зрозумів принцип поліморфізму. Навчився використовувати поліморфізм при створенні об'єктно-орієнтованих програм.