Міністерство освіти і науки України

Черкаський державний технологічний університет

Кафедра програмного забезпечення автоматизованих систем

Звіт

З лабораторної роботи №5

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:** Створення ієрархії класів. Спадкування. Перевизначення методів.

**Мета:** Зрозуміти принцип спадкування. Оволодіти технікою побудови ієрархії класів. Навчитись перевизначати методи.

**Постановка завдання:** Створити ієрархію з двох класів, що позначають певний товар, відповідно до варіанту завдання. Перший клас повинен спадкуватися від JPanel, другий – від першого. В обох класах повинен бути перевизначений метод paint, причому в першому намальоване зображення може бути довільним абстрактним. В кожному з класів повинно бути не менше трьох власних полів та по кілька власних методів. В кожному з класів повинно бути по два конструктора.

Написати на Java програму, яка створює вікно, наповнене елементами інтерфейсу, що забезпечують взаємодію користувача з об'єктами розроблених класів. У вікні повинні бути випадаючий список зі списком конструкторів двох класів та кнопка, при натисненні якої створюється об'єкт вибраним конструктором та виводиться на екран. В один момент часу на екрані може бути лише один об'єкт. Якщо створюється новий об'єкт, то попередній повинен автоматично вилучатись. Також у вікні повинні відображатись елементи введення (текстовими полями, прапорцями, кнопками тощо), через які можна змінити значення полів поточного об'єкту, таким чином змінюючи його стан. Зміна стану повинна відображатись на зображенні об'єкту. У вікні повинен бути об'єкт класу JTextArea (текстова область) та кнопка, при натисненні на яку значення полів об'єкту повинні виводитись в текстову область.

**Індивідуальне завдання:** Предмет меблів та стілець.

**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 FornitureItem fi;

static boolean isFrame = false;

public static void main(String[] args) {

String[] classString = {

"Chair",

"Table"

};

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

JLabel labelMaterial = new JLabel("Material");

JLabel labelForm = new JLabel("Form");

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

JButton btn2 = new JButton("Set legs number");

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 "Chair":

fi = new Chair();

varFrame = "Chair";

fi.toString();

fi.loadImage();

fi.repaint();

break;

case "Table":

fi = new Table();

varFrame = "Table";

fi.toString();

fi.loadImage();

fi.repaint();

break;

default:

throw new IllegalStateException("Incorrect furniture 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());

}

});

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

}

});

btn2.setEnabled(false);

};

}

**Chair**

package com.company;

import java.awt.\*;

public class Chair extends FornitureItem {

@Override

public String[] ik() {

String[] itemsKind = {

"Textile",

"Leather",

"None"

};

return itemsKind;

}

@Override

public String varName1() {

String pole = "Back";

return pole;

};

@Override

public String varName2() {

String pole = "Coating";

return pole;

};

@Override

public String labelText(){

String pole = "Coating";

return pole;

};

@Override

public String toString() {

String str;

String str1 = super.toString();

String str2 =

"\n" +

"Seat Form : " +

this.getForm() +

"\n" +

"Coating Kind : " +

this.getSubMaterial() +

"\n" +

"Back : " +

Boolean.toString(this.getChecker1()) +

"\n" +

"Coating : " +

Boolean.toString(this.getChecker2()) +

"\n";

str = str1 + str2;

QRCodeGenerator makeQR = new QRCodeGenerator();

makeQR.codeGen(str);

loadImage();

repaint();

return str1 + str2;

};

@Override

public void paint(Graphics g) {

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

}

}

**Table**

package com.company;

import java.awt.\*;

public class Table extends FornitureItem {

@Override

public String[] ik() {

String[] itemsKind = {

"Glossiness",

"Mate",

"None"

};

return itemsKind;

}

@Override

public String varName1() {

String pole = "Drawer";

return pole;

};

@Override

public String varName2() {

String pole = "Painting";

return pole;

};

@Override

public String labelText(){

String pole = "Painting";

return pole;

};

@Override

public String toString(){

String str;

String str1 = super.toString();

String str2 =

"\n" +

"Table Form : " +

this.getForm() +

"\n" +

"Painting Kind : " +

this.getSubMaterial() +

"\n" +

"Drawer : " +

Boolean.toString(this.getChecker1()) +

"\n" +

"Painting : " +

Boolean.toString(this.getChecker2()) +

"\n";

str = str1 + str2;

QRCodeGenerator makeQR = new QRCodeGenerator();

makeQR.codeGen(str);

loadImage();

repaint();

return str;

};

@Override

public void paint(Graphics g) {

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

}

}

**FornitureItem**

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 FornitureItem extends JPanel{

private String material;

private int legsNumber;

public String form;

public String subMaterial;

public boolean checker1;

public boolean checker2;

public JFrame mainFrame;

public FornitureItem() {

material = "Wood";

legsNumber = 4;

form = "Square";

checker1 = false;

checker2 = false;

loadImage();

};

String[] itemsMaterial = {

"Wood",

"Metal",

"Plastic"

};

String[] itemsForm = {

"Square",

"Rectangle",

"Circle"

};

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 loadImage() {

try {

image = ImageIO.read(new File("BM.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 labelForm, 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 comboBoxForm = new JComboBox(this.itemsForm);

JComboBox comboBoxSubMaterial = new JComboBox(ik());

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(labelForm);

comboBoxPanel.add(labelExt);

comboBoxPanel.add(comboBoxMaterial);

comboBoxPanel.add(comboBoxForm);

comboBoxPanel.add(comboBoxSubMaterial);

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

checkBoxPanel.add(checkBox1);

checkBoxPanel.add(checkBox2);

ActionListener actionListenerMaterial = new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

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

}

};

comboBoxMaterial.addActionListener(actionListenerMaterial);

ActionListener actionListenerForm = new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

setForm(comboBoxForm.getSelectedItem().toString());

}

};

comboBoxForm.addActionListener(actionListenerForm);

ActionListener actionListenerTC = new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

setSubMaterial(comboBoxSubMaterial.getSelectedItem().toString());

}

};

comboBoxSubMaterial.addActionListener(actionListenerTC);

checkBox1.addItemListener(new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

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

}

});

comboBoxSubMaterial.setEnabled(false);

setSubMaterial("None");

checkBox2.addItemListener(new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

if(e.getStateChange() == ItemEvent.SELECTED) {

comboBoxSubMaterial.setEnabled(true);

setSubMaterial(comboBoxSubMaterial.getSelectedItem().toString());

setChecker2(true);

} else {

comboBoxSubMaterial.setEnabled(false);

setSubMaterial("None");

setChecker2(false);

};

}

});

};

public void destructForm() {

mainFrame.dispose();

mainFrame.setVisible(false);

};

public String toString(){

return

"Material : " +

this.getMaterial() +

"\n" +

"Legs Number : " +

Integer.toString(this.getLegsNumber());

};

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 setForm(String form){

this.form = form;

}

public String getForm() {

return form;

}

public void setSubMaterial(String subMaterial){

this.subMaterial = subMaterial;

}

public String getSubMaterial() {

return subMaterial;

}

public void setMaterial(String material){

this.material = material;

}

public String getMaterial() {

return material;

}

public void setLegsNumber(int legsNumber) { this.legsNumber = legsNumber; }

public int getLegsNumber() { return legsNumber; }

}

**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) {

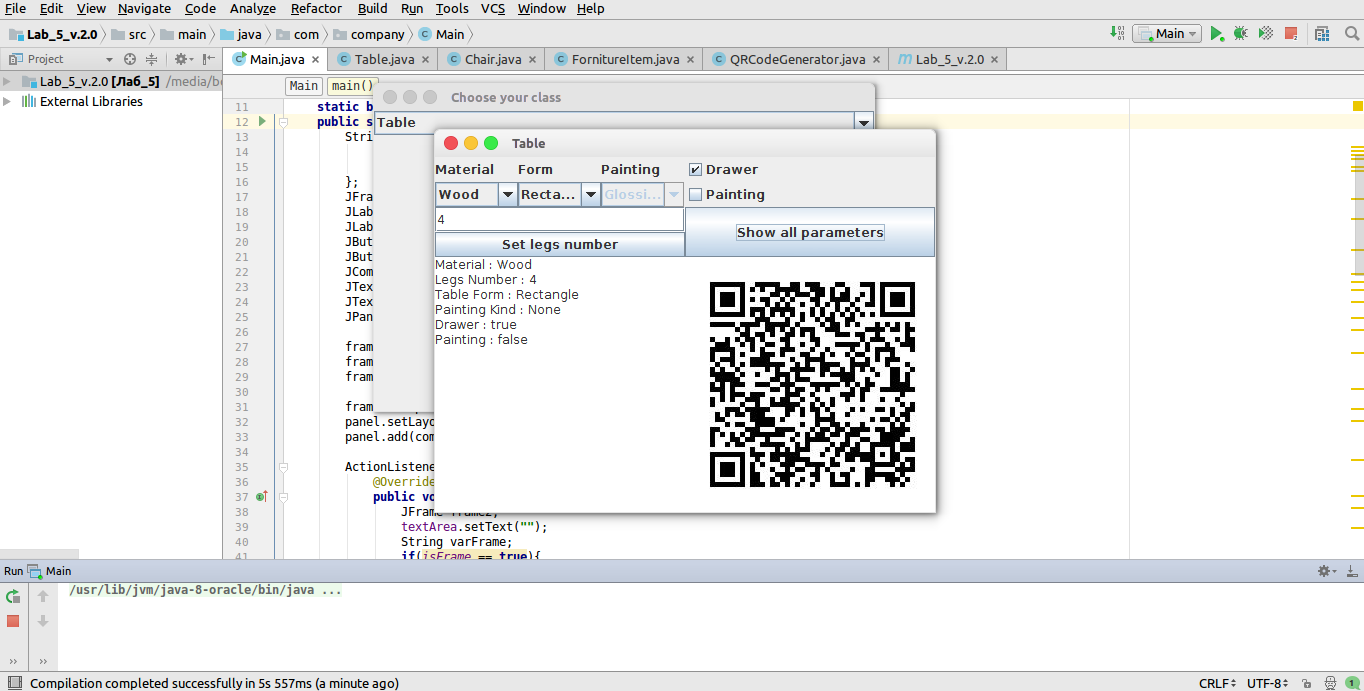
// TODO Auto-generated catch block

e.printStackTrace();

}

}

}



**Результат роботи програми:**

**Висновок:** У ході виконання лабораторної роботи, я зрозумів принцип спадкування. Оволодіd технікою побудови ієрархії класів. Навчився перевизначати методи.