Одесский национальный политехнический университет

Кафедра интеллектуальных систем и сетей

Лабораторная работа №9

По предмету «Системное программирование»

На тему «**Основы языка C#. Интерфейсы, события.**»

Выполнили

Студенты группы АЕ-145

Михайлов О.М.

Зайцева М.О.

Проверил

Олещук О.В.

Одессы 2017

**Цель лабораторной работы**

Спроектировать интерфейсы согласно варианту задания. Для каждого интерфейса разработать три реализующих его класса. Все необходимые данные должны передаваться в класс **только посредством интерфейсных методов**. Таким же образом должны возвращаться результаты работы. Ввод-вывод данных должен осуществляться за пределами классов, реализующих интерфейсы.

Предусмотреть события, связанные с инициализацией класса, а также события, вызываемые до и после вызова основных сервисных функций.

Обеспечить корректное поведение программы в случае отсутствия реализации части функциональности интерфейса (например, в случае отсутствия в файле нужного значения).

Обеспечить выбор пользователем в процесс работы программы одной из трех реализаций интерфейса.

Варианты заданий

Вариант 1

Разработать интерфейс для перевода текста по словам. На базе данного интерфейса реализовать следующие классы:

переводчик, основанный на словах, считанных из файла;

переводчик, просто преобразующий латиницу в кириллицу или наоборот;

класс-заглушку, возвращающий исходную строку без изменения.

**Ход выполнения работы:**

**package** com.translate;

**public interface** TranslateIF {

**void** translateText(String text);

**void** transcriptionText(String text);

**void** immutableText(String text);

String getText();

String viewText();

}

**package** com.translate;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.util.HashMap;

**public class** Translator **implements** TranslateIF {

**private** String[] **buffer**;

**private** StringBuilder **result** = **new** StringBuilder();

**private** HashMap<String, String> **dictionary** = **new** HashMap<String, String>();

**public** Translator() {

**dictionary**.put(**"олег"**, **"оleg"**);

**dictionary**.put(**"маша"**, **"мasha"**);

**dictionary**.put(**"кукуруза"**, **"corn"**);

**dictionary**.put(**"картошка"**, **"potato"**);

}

**public void** translateText(String text) {

**buffer** = text.trim().split(**" "**);

**result**.delete(0, **result**.length());

**for** (**int** i = 0; i < **buffer**.**length**; i++) {

**if** (**dictionary**.containsKey(**buffer**[i])) {

**result**.append(**dictionary**.get(**buffer**[i]) + **" "**);

}

}

}

**public void** transcriptionText(String text) {

**buffer** = text.trim().split(**" "**);

**result**.delete(0, **result**.length());

**for** (**int** i = 0; i < **buffer**.**length**; i++) {

**if** (**dictionary**.containsKey(**buffer**[i])) {

**result**.append(getTransription(**buffer**[i]) + **" "**);

}

}

}

**public void** immutableText(String text) {

**result**.delete(0, **result**.length());

**result**.append(text);

}

**public** String getText() {

String tmp = **null**;

**try** {

BufferedReader bufferedReader = **new** BufferedReader(

**new** FileReader(**"/home/mikha/Документы/Universe/lab9/text.txt"**));

tmp = bufferedReader.readLine();

} **catch** (IOException e) {

e.printStackTrace();

}

**return** tmp;

}

**public** String viewText() {

**return result**.toString();

}

**private** String getTransription(String s) {

StringBuilder result = **new** StringBuilder(**""**);

**char**[] tmp = s.toCharArray();

**for** (**int** i = 0; i < tmp.**length**; i++) {

**switch** (tmp[i]) {

**case 'а'**:

result.append(**'a'**);

**break**;

**case 'б'**:

result.append(**'b'**);

**break**;

**case 'в'**:

result.append(**'v'**);

**break**;

**case 'г'**:

result.append(**'g'**);

**break**;

**case 'д'**:

result.append(**'d'**);

**break**;

**case 'е'**:

result.append(**'e'**);

**break**;

**case 'ё'**:

result.append(**"je"**);

**break**;

**case 'ж'**:

result.append(**"zh"**);

**break**;

**case 'з'**:

result.append(**'z'**);

**break**;

**case 'и'**:

result.append(**'i'**);

**break**;

**case 'й'**:

result.append(**'y'**);

**break**;

**case 'к'**:

result.append(**'k'**);

**break**;

**case 'л'**:

result.append(**'l'**);

**break**;

**case 'м'**:

result.append(**'M'**);

**break**;

**case 'н'**:

result.append(**'n'**);

**break**;

**case 'о'**:

result.append(**'o'**);

**break**;

**case 'п'**:

result.append(**'p'**);

**break**;

**case 'р'**:

result.append(**'r'**);

**break**;

**case 'с'**:

result.append(**'s'**);

**break**;

**case 'т'**:

result.append(**'t'**);

**break**;

**case 'у'**:

result.append(**'u'**);

**break**;

**case 'ф'**:

result.append(**'f'**);

**break**;

**case 'х'**:

result.append(**"kh"**);

**break**;

**case 'ц'**:

result.append(**'c'**);

**break**;

**case 'ч'**:

result.append(**"ch"**);

**break**;

**case 'ш'**:

result.append(**"sh"**);

**break**;

**case 'щ'**:

result.append(**"jsh"**);

**break**;

**case 'ъ'**:

result.append(**"hh"**);

**break**;

**case 'ы'**:

result.append(**"ih"**);

**break**;

**case 'ь'**:

result.append(**"jh"**);

**break**;

**case 'э'**:

result.append(**"eh"**);

**break**;

**case 'ю'**:

result.append(**"ju"**);

**break**;

**case 'я'**:

result.append(**"ja"**);

**break**;

}

}

**return** result.toString();

}

}

**package** com.translate;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JToolBar;

**public class** Main {

**static** JButton *button1*;

**static** JButton *button2*;

**static** JButton *button3*;

**static** String *text*;

**static** Translator *translator* = **new** Translator();

**public static void** main(String[] args) {

**final** JFrame frame = **new** JFrame(**"Second"**);

*text* = *translator*.getText();

*button1* = **new** JButton(**"File"**);

*button1*.addActionListener(**new** ActionListener() {

**public void** actionPerformed(ActionEvent actionEvent) {

*translator*.translateText(*text*);

JLabel textView = **new** JLabel();

JFrame newFrame = **new** JFrame();

textView.setText(*translator*.viewText());

newFrame.add(textView);

newFrame.setSize(200, 100);

newFrame.setVisible(**true**);

}

});

*button2* = **new** JButton(**"Transcription"**);

*button2*.addActionListener(**new** ActionListener() {

**public void** actionPerformed(ActionEvent actionEvent) {

JLabel textView2 = **new** JLabel();

JFrame newFrame = **new** JFrame();

*translator*.transcriptionText(*text*);

textView2.setText(*translator*.viewText());

newFrame.add(textView2);

newFrame.setSize(200, 100);

newFrame.setVisible(**true**);

}

});

*button3* = **new** JButton(**"Cap"**);

*button3*.addActionListener(**new** ActionListener() {

**public void** actionPerformed(ActionEvent actionEvent) {

JLabel textView3 = **new** JLabel();

JFrame newFrame = **new** JFrame();

*translator*.immutableText(*text*);

textView3.setText(*translator*.viewText());

newFrame.add(textView3);

newFrame.setSize(200, 100);

newFrame.setVisible(**true**);

}

});

frame.setSize(250, 50);

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

JToolBar toolBar = **new** JToolBar(**"Choose type"**);

toolBar.add(*button1*);

toolBar.add(*button2*);

toolBar.add(*button3*);

frame.add(toolBar);

frame.setVisible(**true**);

}

}

Вывод

В данной лабораторной работе мы разобрали основы ООП с использованием интерфейсов для реализации шаблонного скелета переводчика, который работает с определённым набором словарём в HashMap и преобразует текст транскрипцией.