Одесский национальный политехнический университет

Кафедра интеллектуальных систем и сетей

Лабораторная работа №10

По предмету «Системное программирование»

На тему «**Коллекции**»

Выполнили

Студенты группы АЕ-145

Михайлов О.М.

Зайцева М.О.

Проверил

Олещук О.В.

Одессы 2017

**Цель лабораторной работы**

Создать класс, представляющий собой элемент коллекции.

Обеспечить добавление элементов в коллекцию, удаление из коллекции, вывод всей коллекции на экран.

Варианты заданий

Вариант 1

Разработать программу для работы со списком студентов.

Тип коллекции: List.

Реализовать интерфейс: IComparable.

Обеспечить сортировку студентов по алфавиту.

**Ход выполнения работы:**

**package customList;**

**public interface** CustomList **extends** Iterable {

**int** size();

**boolean** isEmpty();

**boolean** contains(String element);

**void** add(Student student);

**void** add(**int** index, Student element);

**void** remove(Student element);

Student remove(**int** index);

**void** clear();

Student get(**int** index);

**int** indexOf(Student element);

**void** sortList();

}

**package customList;**

**public class** Student {

String name;

**int age**;

**public** Student(String name, **int** age) {

**this**.name = name;

**this**.**age** = age;

}

@Override

**public** String toString() {

**return "Student{"** +

**"name='"** + name + **'\''** +

**", age="** + **age** +

**'}'**;

}

**public** String getName() {

**return** name;

}

}

**package customList;**

**public class** MainCustom {

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

CustomList list = **new** CustomArrayList();

Student st1 = **new** Student(**"Oleg"**, 10);

Student st2 = **new** Student(**"Anton"**, 12);

Student st3 = **new** Student(**"Leha"**, 141);

Student st4 = **new** Student(**"Masha"**, 14);

list.add(st1);

list.add(st2);

list.add(st3);

list.add(st4);

System.***out***.println(list.toString());

list.sortList();

}

}

**package** customList;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.Comparator;

**import** java.util.Iterator;

**import** java.util.NoSuchElementException;

**public class** CustomArrayList **implements** CustomList {

**private static final int *INITIAL\_CAPACITY*** = 9;

**private** Student[] **array**;

**private int size** = 0;

**public** CustomArrayList() {

**this**(***INITIAL\_CAPACITY***);

**this**.**students** = **new** ArrayList<>();

}

**private** ArrayList<Student> **students**;

**public** CustomArrayList(**int** capacity) {

**array** = **new** Student[capacity];

}

**public int** size() {

**return size**;

}

**public boolean** isEmpty() {

**return size** == 0;

}

**public boolean** contains(String element) {

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

**if** (**array**[i].equals(element)) {

**return true**;

}

}

**return false**;

}

**public void** add(Student element) {

resizeArray();

**array**[**size**] = element;

**students**.add(element);

**size**++;

}

**public void** sortList() {

Collections.*sort*(**students**, **new** Comparator<Student>() {

@Override

**public int** compare(Student s1, Student s2) {

**return** s1.getName().compareTo(s2.getName());

}

});

System.***out***.println(**students**.toString());

}

**public void** add(**int** index, Student element) {

resizeArray();

**if** (index == **size** + 1) {

**size**++;

}

**array**[index - 1] = element;

**students**.add(element);

}

**private void** resizeArray() {

Student[] arrayTmp;

**if** (**size** == **array**.**length**) {

arrayTmp = **new** Student[**size** \* 2];

System.*arraycopy*(**array**, 0, arrayTmp, 0, **size**);

**array** = arrayTmp;

}

**if** (**size** == **array**.**length** / 4) {

arrayTmp = **new** Student[**array**.**length** / 2];

System.*arraycopy*(**array**, 0, arrayTmp, 0, **size**);

**array** = arrayTmp;

}

}

**public void** remove(Student element) {

**int** index = indexOf(element);

**for** (**int** i = index + 1; i < **size** - 1; i++) {

**array**[i] = **array**[i + 1];

**students**.remove(element);

}

**size**--;

}

**public** Student remove(**int** index) {

**int** indexDelete = index;

**if** (index == **size**) {

**array**[index - 1] = **null**;

}

**for** (**int** i = index - 1; i < **size** - 1; i++) {

**array**[i] = **array**[indexDelete];

**students**.remove(indexDelete);

indexDelete++;

}

**size**--;

**array**[**size**] = **null**;

resizeArray();

**return array**[index];

}

**public void** clear() {

**size** = 0;

resizeArray();

**students**.clear();

}

**public** Student get(**int** index) {

**if** (index <= 0) {

**throw new** NullPointerException(**"An array starts at 1"**);

}

**return array**[index - 1];

}

**public int** indexOf(Student element) {

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

**if** (**array**[i].equals(element)) {

**return** i;

}

}

**throw new** NoSuchElementException(

**"No such element int this list"**);

}

**public** Iterator iterator() {

**return new** ListIterator();

}

**private class** ListIterator **implements** Iterator {

**private int current**;

ListIterator() {

**current** = 0;

}

**public boolean** hasNext() {

**return current** < **size**;

}

**public** Object next() {

**if** (**array**.**length** == 0) {

**throw new** NoSuchElementException();

}

Student element = **array**[**current**];

**current**++;

**return** element;

}

}

@Override

**public** String toString() {

**return** Arrays.*toString*(**array**);

}

*///////////////////////////////*

**public class** UpSort **implements** Comparator<Student> {

**private** Student[] **students**;

**public** UpSort(Student[] students) {

**this**.**students** = students;

}

@Override

**public int** compare(Student s1, Student s2) {

**return** s1.getName().compareTo(s2.getName());

}

}

}

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.