Министерство образования и науки Украины

Одесский Политехнический Университет

Кафедра “Информационные технологии”

Отчёт по лабораторной работе №9

На тему: «**Коллекции объектов**»

Выполнил cтудент группы:

АД-201

Стыцковский Н.Ю.

Проверили:

Рудниченко Н.Д.

Оглавление

[Цель работы: 3](#_1fob9te)

[Теоретические сведения 3](#_3znysh7)

[Задание 6](#_2et92p0)

[Листинг 6](#_tyjcwt)

[Скриншоты: 10](#_3dy6vkm)

[Вывод: 11](#_1t3h5sf)

Цель работы: Научиться создавать автономные приложения для работы с окнами и элементами меню.

# Теоретические сведения

В Java имеется несколько способов хранения объектов (или, точнее, ссылок на объекты). Самый простой вариант хранения объектов – массивы. Массив обеспечивает самый эффективный способ хранения групп объектов. Однако массив имеет фиксированный размер, а в общем случае во время написания программы разработчик может не знать точное количество объектов или необходимо более эффективные способы хранения объектов. К тому же, массивы неэффективны при большом объеме элементов. Для таких случаев, в Java и других языках реализован механизм коллекций объектов.

Коллекции – это хранилища, поддерживающие различные способы накопления и упорядочения объектов с целью обеспечения возможностей эффективного доступа к ним. Они представляют собой реализацию абстрактных типов (структур) данных, поддерживающих три основные операции:

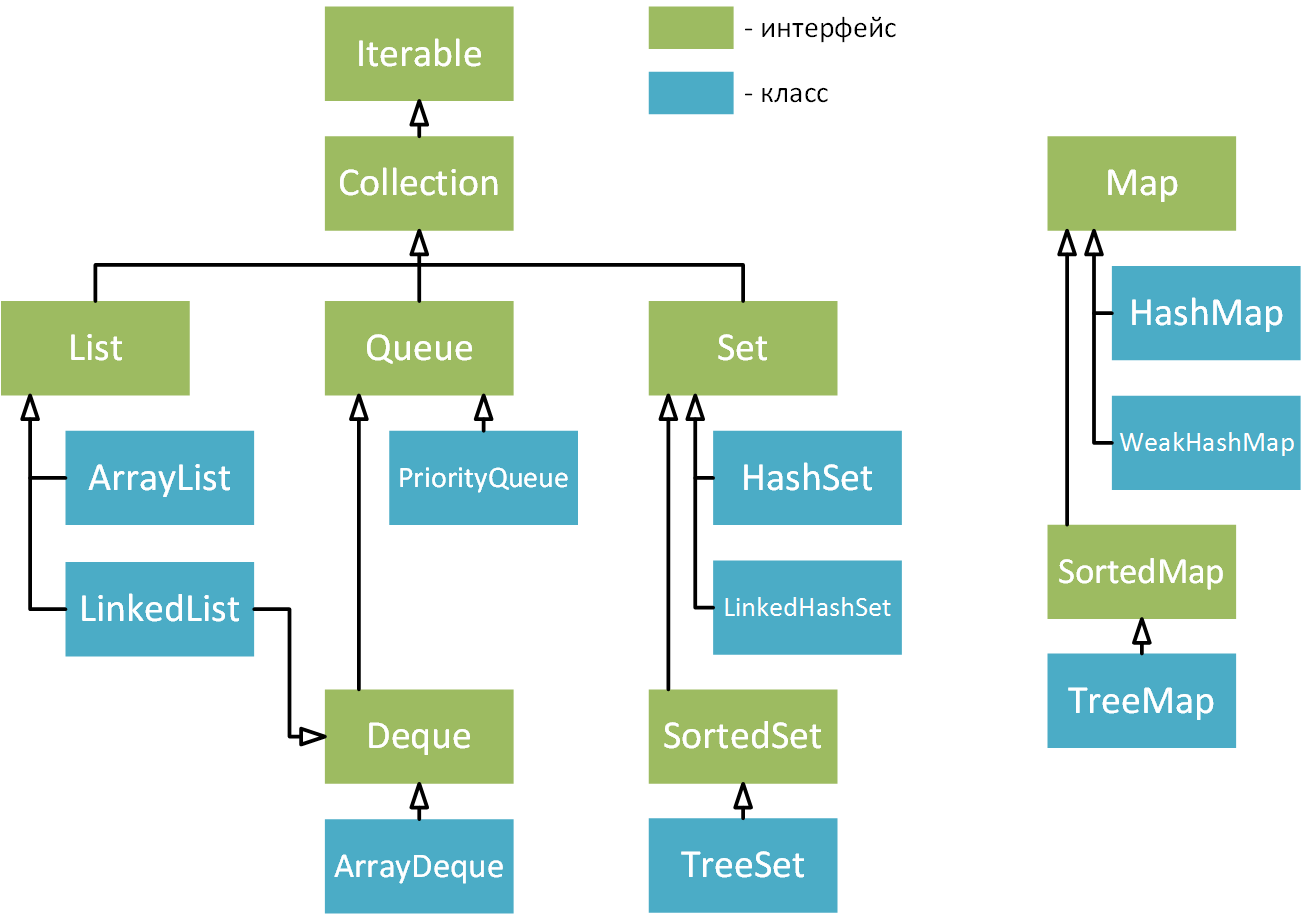
* добавление нового элемента в коллекцию;
* удаление элемента из коллекции;
* изменение элемента в коллекции.

Коллекции в Java объединены в библиотеке классов java.util и представляют собой контейнеры для хранения и манипулирования объектами.

Основные интерфейсы коллекций:

* Map<K,V> – карта отображения вида «ключ-значение»;
* Collection<E> – вершина иерархии остальных коллекций;
* List<E> – специализирует коллекции для обработки списков;
* Set<E> – специализирует коллекции для обработки множеств, содержащих уникальные элементы.

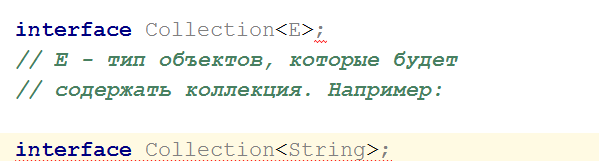
На изображении ниже приведены основные интерфейсы и классы коллекций



*Основные интерфейсы и классы коллекций в Java*

ИНТЕРФЕЙС COLLECTION

Этот интерфейс служит основанием, на котором построен весь каркас коллекций, поскольку он должен быть реализован почти всеми классами коллекций (кроме коллекций, реализующих интерфейс Map). Интерфейс Collection является обобщенным и объявляется следующим образом:



Интерфейс Collection расширяет интерфейс Iterable (interface Collection extends Iterable). Это означает, что все коллекции можно перебирать, организовав цикл for в стиле for each. В интерфейсе Collection определяются основные методы, которые должны иметь все коллекции. Некоторые основные методы перечислены в таблице ниже.

*Некоторые основные методы, определенные в интерфейсе Collection*

|  |  |
| --- | --- |
| Метод | Описание |
| add(Object o) | Добавляет указанный объект в коллекцию |
| remove(Object o) | Удаляет указанный объект из коллекции |
| clear() | Удаляет все элементы из коллекции |
| size() | Возвращает количество элементов в коллекции |
| iterator() | Возвращает объект, который используется для доступа к элементам коллекции |

Интерфейс List

Список (List) представляет собой упорядоченный набор элементов и может содержать повторяющиеся элементы. Вы можете получить доступ к любому элементу по индексу. Список представляет собой динамический массив. Список является одним из наиболее используемых типов коллекций.

Основные свойства коллекций, реализующих интерфейс List:

* список может включать одинаковые элементы;
* элементы в списке хранятся в том порядке, в котором они помещались;
* можно получить доступ к любому элементу по его порядковому номеру (индексу) внутри списка

Существуют две основные разновидности List:

1) Базовый контейнер ArrayList с превосходной скоростью произвольного доступа к элементам, но относительно медленными операциями вставки и удаления элементов в середине. Пожалуй, самая часто используемая коллекция. ArrayList инкапсулирует в себе обычный массив, длина которого автоматически увеличивается при добавлении новых элементов.

Так как ArrayList использует массив, то время доступа к элементу по индексу минимально (в отличие от LinkedList). При удалении произвольного элемента из списка, все элементы находящиеся «правее» смещаются на одну ячейку влево, при этом реальный размер массива (его емкость, capacity) не изменяется. Если при добавлении элемента, оказывается, что массив полностью заполнен, будет создан новый массив размером , в него будут помещены все элементы из старого массива плюс новый, добавляемый элемент.

2) Связанный список LinkedList с оптимальным последовательным доступом и низкозатратными операциями вставки и удаления в середине списка. Операции произвольного доступа LinkedList выполняет относительно медленно, но обладает более широкой функциональностью, чем ArrayList.

Это структура данных, состоящая из узлов, каждый из которых содержит как собственно данные, так и две ссылки («связки») на следующий и предыдущий узел списка. Доступ к произвольному элементу осуществляется за линейное время (но доступ к первому и последнему элементу списка всегда осуществляется за константное время — ссылки постоянно хранятся на первый и последний, так что добавление элемента в конец списка вовсе не значит, что придется перебирать весь список в поисках последнего элемента). В целом же, LinkedList в абсолютных величинах проигрывает ArrayList и по потребляемой памяти и по скорости выполнения операций.

В общем случае, следует использовать ArrayList. Коллекцию LinkedList имеет смысл использовать в случае, если происходит интенсивная вставка\удаление в середину списка либо необходимо гарантированное, заранее известное время добавления элемента в список.

В таблице ниже приведены некоторые методы интерфейса List, которые используются в ArrayList и LinkedList.

# Задание

1. Дан список из 100 целочисленных значений от -1000 до 1000 (создайте и сгенерируйте список). Не используя вспомогательных объектов, переставьте отрицательные элементы данного списка в конец, а положительные – в начало списка. Не используйте методы сортировки.

2. Дан случайный текст – <http://pastebin.com/JEwMJVbh>. Выделить все различные слова. Слова, отличающиеся только регистром букв, считать одинаковыми (подумайте о том – какую коллекцию лучше использовать).

3. Дан случайный текст – <http://pastebin.com/JEwMJVbh>. Выделить все различные слова. Для каждого слова подсчитать частоту его встречаемости (). Слова, отличающиеся регистром букв, считать различными (подумайте о том – какую коллекцию лучше использовать).

4. Дан случайный текст – <http://pastebin.com/JEwMJVbh>. Разбить текст на предложения, каждое предложение записать в список. Выполнить сортировку строк, используя метод **sort()** из класса **Collections**. Результат вывести на экран.

# Листинг

Main:

**package** com.company;  
**import** java.util.Scanner;  
**public class** Main  
{  
 **public static void** main(String[] args)  
 {  
 Scanner scanner = **new** Scanner(System.***in***);  
 **int** choice = scanner.nextInt();  
 **switch** (choice)  
 {  
 **case** 1:  
 Task1.FirstTask();  
 **case** 2:  
 Task2.SecondTask();  
 **case** 3:  
 Task3.ThirdTask();  
 **case** 4:  
 Task4.FourthTask();  
 }  
 }  
}

**Задание 1:**

**package** com.company;  
**import** java.util.ArrayList;**public class** Task1 {  
 **public static void** FirstTask() {  
 ArrayList<Integer> list = **new** ArrayList<Integer>();  
 **for** (**int** i = 0; i < 100; i++) {  
 **double** rand = Math.*random*() \* 2000 - 1000;  
 list.add((**int**) rand);  
 }**boolean** isSorted = **false**;  
 **int** buf;  
 **while** (!isSorted) {  
 isSorted = **true**;  
 **for** (**int** i = 0; i < list.size() - 1; i++) {  
 **if** (list.get(i) > list.get(i + 1)) {  
 isSorted = **false**;  
 buf = list.get(i);  
 list.set(i,list.get(i+1) );  
 list.set(i + 1, buf);  
 }  
 }  
 }  
 **for** (**int** i = 0; i < list.size(); i++)  
 System.***out***.println(list.get(i));  
 }  
}

**Задание 2:**

**package** com.company;  
**import** java.io.\*;  
**import** java.util.HashMap;  
**import** java.util.Map;**public class** Task2 {  
 **public static void** SecondTask() {  
 String str = **""**;  
 Map<String, Integer> map = **new** HashMap<>();  
 **try** (BufferedReader br = **new** BufferedReader(**new** FileReader(**"C:\\Users\\user\\Desktop\\ Политех20-21 \\ООП\\lab9\\src\\com\\company\\Task2.txt"**))) {  
 **while** ((str = br.readLine()) != **null**) {  
 str = str.toLowerCase();str = str.replaceAll(**"\\W"**, **" "**);  
 System.***out***.println(str);  
 **for**(String s: str.split(**" "**)) {  
 **if**(map.containsKey(s)){  
 map.put(s, map.get(s)+1);  
 }**else** {  
 map.put(s, 1);  
 }  
 }  
 }  
 } **catch** (IOException ex) {  
  
 System.***out***.println(ex.getMessage());  
 }  
 map.remove(**""**);  
 System.***out***.println(map);  
 }  
}

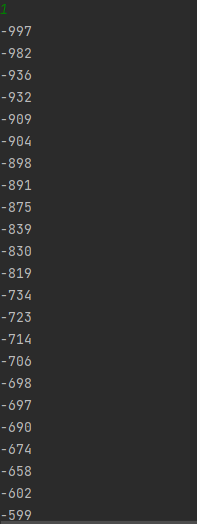
**Задание 3:**

**package** com.company;  
**import** java.io.\*;  
**import** java.util.HashMap;  
**import** java.util.Map;  
**public class** Task3 {  
 **public static void** ThirdTask() {  
 String str = **""**;  
 **int** wordCounter = 0;  
 Map<String, Integer> map = **new** HashMap<>();  
 **try** (BufferedReader br = **new** BufferedReader(**new** FileReader(**"C:\\Users\\user\\Desktop\\Политех20-21\\ООП\\lab9\\src\\com\\company\\Task2.txt"**))) {  
 **while** ((str = br.readLine()) != **null**) {  
 str = str.replaceAll(**"\\W"**, **" "**);  
 System.***out***.println(str);  
  
 **for**(String s: str.split(**" "**)) {  
 **if**(map.containsKey(s)){  
 map.put(s, map.get(s)+1);  
 wordCounter++;  
 }**else** {  
 map.put(s, 1);  
 }  
 }  
 }  
 } **catch** (IOException ex) {  
 System.***out***.println(ex.getMessage());  
 }  
 map.remove(**""**);  
 System.***out***.println(map);  
 System.***out***.println(**"Общее кол-во слов: "**+wordCounter);  
  
 }  
}

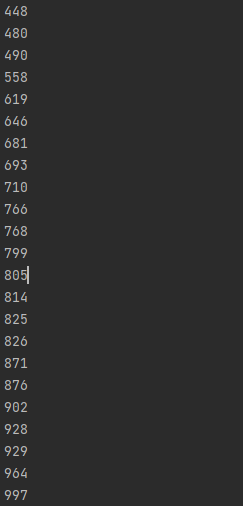
**Задание 4:**

**package** com.company;  
**import** java.io.BufferedReader;  
**import** java.io.FileReader;  
**import** java.io.IOException;  
**import** java.util.\*;**public class** Task4 {  
 **public static void** FourthTask(){  
 ArrayList<String> list = **new** ArrayList<>();  
 String str = **""**;  
 String[] str1;  
 **try** (BufferedReader br = **new** BufferedReader(**new** FileReader(**"C:\\Users\\ user\\Desktop\\Политех20-21\\ООП\\lab9\\src\\com\\company\\Task2.txt"**))) {  
 **while** ((str = br.readLine()) != **null**) {  
 str1 = str.split(**"\\."**);  
 **for**(**int** i =0;i<str1.**length**;i++) {  
 list.add(str1[i]);  
 System.***out***.println(str1[i]);  
 }  
 System.***out***.println();  
 System.***out***.println(list);  
 Collections.*sort*(list);  
 System.***out***.println();  
 System.***out***.println(list);  
 }  
 }  
 **catch** (IOException ex) {  
 System.***out***.println(ex.getMessage());  
 }  
 }  
}

# Скриншоты:



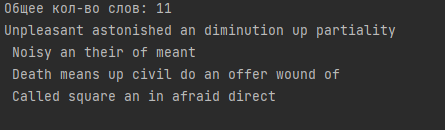
Скриншот 1



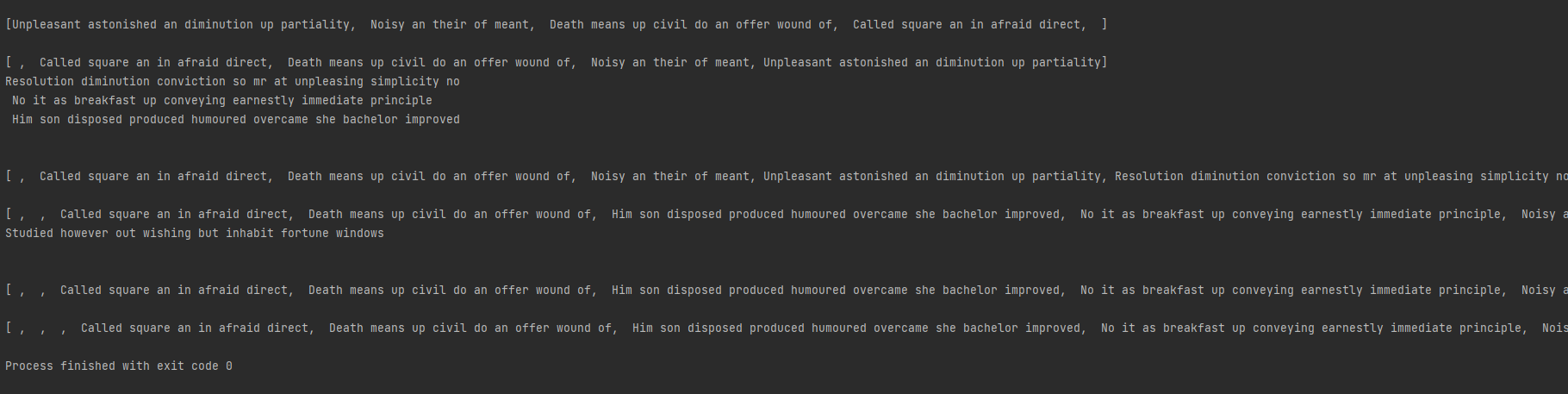
Скриншот 2



Скриншот 3



Скриншот 4



Скриншот 5

Задание 10

**Main**

package com.mycompany.lab.stytskovskyi;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

/\*\*

\*

\* @author USER

\*/

public class Main

{

public static void main(String[] args)

{

Scanner sc = new Scanner (System.in);

List<Parallelepiped> parallelepipedList = new ArrayList<> ();

int amount;

System.out.println("Введите количество параллелепипедов для генерации (10 макс)");

amount = sc.nextInt();

if (amount > 10)

amount = 10;

for (int i = 0; i < 10; i++)

parallelepipedList.add(generateParallelepiped());

parallelepipedSort(parallelepipedList);

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

System.out.println("(" + parallelepipedList.get(i).getLength() + ", " + parallelepipedList.get(i).getWidth() +

", " + parallelepipedList.get(i).getHeight() + ") - Area equals to : " + calculateArea(parallelepipedList.get(i)));

}

public static void parallelepipedSort (List<Parallelepiped> parallelepipedArray)

{

List<Parallelepiped> tempList = new ArrayList<> ();

List<Parallelepiped> comparatorList = new ArrayList<> ();

comparatorList.addAll(parallelepipedArray);

boolean delete;

int areaFirst, areaNext;

int number;

for (int j = 0; j < parallelepipedArray.size(); j++)

{

areaFirst = calculateArea(parallelepipedArray.get(j));

number = j;

delete = false;

for (int i = 1; i < comparatorList.size(); i++)

{

areaNext = calculateArea(comparatorList.get(i));

if (areaFirst > areaNext)

{

areaFirst = areaNext;

number = i;

delete = true;

}

else if (areaFirst == areaNext &&

parallelepipedArray.get(j).getLength() > comparatorList.get(i).getLength())

{

number = i;

delete = true;

}

else if (areaFirst == areaNext &&

parallelepipedArray.get(j).getLength()== comparatorList.get(i).getLength()&&

parallelepipedArray.get(j).getWidth()> comparatorList.get(i).getWidth())

{

number = i;

delete = true;

}

}

tempList.add(parallelepipedArray.get(number));

if (delete == true)

{

comparatorList.remove(number);

parallelepipedArray.remove(number);

j--;

}

}

parallelepipedArray.removeAll(parallelepipedArray);

parallelepipedArray.addAll(tempList);

}

public static Parallelepiped generateParallelepiped()

{ return new Parallelepiped ((int)(0 + Math.random() \* 25),(int)(0 + Math.random() \* 25), (int)(0 + Math.random() \* 25)); }

public static int calculateArea(Parallelepiped a)

{

return 2 \* ((a.getHeight() \* a.getWidth()) +

(a.getWidth() \* a.getLength()) +

(a.getHeight() \* a.getLength()));

}

public static class Parallelepiped

{

private int length, width, height;

public Parallelepiped(int length, int width, int height)

{setLength(length); setWidth(width); setHeight(height);}

public void setLength(int length) { this.length = length; }

public void setWidth(int width) {this.width = width; }

public void setHeight(int height) {this.height = height; }

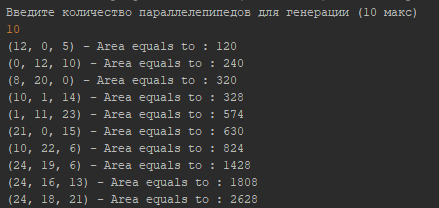
public int getLength() {return length;}

public int getWidth() {return width;}

public int getHeight() {return height;}

}

}



# Вывод:

В ходе выполненной работе мы научились пользоваться коллекциями, и с помощью них смогли выполнить разные задания.