Міністерство Освіти і Науки України

Національний технічний університет

«Харківський політехнічний інститут»

Кафедра «Програмної інженерії та інформаційних технологій управління»

Лабораторна робота № 1

# Робота з узагальненнями та колекціями в Java

**Виконав:**

Студент групи КН-36А

Кулик В.В.

**Перевірила:**

Козуля М.М.

Харків 2017

**Завданная**

### 1.1 Індивідуальне завдання

Розробити ієрархію класів для представлення сутностей індивідуального завдання. Базовий абстрактний клас, який представляє першу з сутностей індивідуального завдання [лабораторної роботи № 5 курсу "Алгоритмізація та програмування" (друга частина)](http://iwanoff.96.lt/algorithmization_2/LabTraining05.html), повинен містити абстрактні функції для доступу до елементів типу другого класу попередньої роботи та сортування цих елементів за певною ознакою. Похідні класи від створеного абстрактного класу повинні відповідно представляти послідовність елементів у вигляді масиву та списку. Здійснити тестування обох реалізацій. Тест повинен включати виконання завдання попередньої лабораторної роботи, а також сортування за визначеними ознаками відповідно до завдання [лабораторної роботи № 6 курсу "Алгоритмізація та програмування" (друга частина)](http://iwanoff.96.lt/algorithmization_2/LabTraining06.html).

Для сортування слід використовувати методи sort() класів Arrays та Collections відповідно. Для визначення ознак сортування використати лямбда-вирази.

### 1.2 Мінімум функції

Реалізувати програму, що дозволяє знайти мінімум деякої функції на заданому інтервалі. Алгоритм знаходження мінімуму полягає в послідовному переборі з певним кроком точок інтервалу і порівнянні значень функції в поточній точці з раніше знайденим мінімумом.

Знайти та застосувати відповідний стандартний функціональний інтерфейс для опису функції.

Реалізувати два підходи - через використання лямбда-виразів і через використання вказівників на методи.

### 1.3 Узагальнений клас

Створити узагальнений клас для зберігання довільних даних у масиві. Реалізувати функцію додавання елементу в кінець масиву, видалення елементу, додавання групи (іншого масиву) елементів.

### 1.4 Створення бібліотеки узагальнених функцій для роботи з масивами та списками

Реалізувати клас зі статичними узагальненими методами, які реалізують таку функціональність:

* обмін місцями двох груп елементів
* обмін місцями усіх пар сусідніх елементів (з парним і непарним індексом)
* заміна групи елементів іншим масивом (списком) елементів
* вставлення у масив (список) іншого масиву (списку) елементів у вказане місце

Реалізувати наведені функції для масивів і для списків. Здійснити демонстрацію роботи усіх методів з використанням даних різних типів (Integer, Double, String) .

### 1.5 Реалізація інтерфейсу Comparable

Створити клас Circle, який реалізує інтерфейс Comparable. Більшим вважається коло з більшим радіусом. Здійснити сортування списку об'єктів типу Circle.

### 1.6 Реалізація інтерфейсу Comparator

Створити клас Triangle. Трикутник визначати довжинами сторін. Площа трикутника в цьому випадку може бути обчислена за формулою Герона:

http://iwanoff.96.lt/oop_kn/Images/07_Heron.png

де *a*, *b* і *c* - довжини сторін трикутника. Здійснити сортування списку трикутників за зменшенням площі. Для визначення ознаки сортування використовувати об'єкт, який реалізує інтерфейс Comparator.

**Хід роботи**

### 1.1 Індивідуальне завдання

*Код програми 1.1:*

**Doctor**

package F\_ex;

public abstract class Doctor {

private String name;

private String spec;

abstract public Reception getReception(int i);

abstract public void setReception(int i,Reception recep);

abstract public int recepCount();

abstract public void addReception(Reception recep);

abstract public void clearRecep();

abstract public void sortByDay();

abstract public void sortByVisit();

public Doctor() {

this.name = name;

this.spec = spec;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;}

public String getSpec() {

return spec;

}

public void setSpec(String spec) {

this.spec = spec;

}

public int allVisitors(){

int s=0;

for(int i=0;i<recepCount();i++){

s+=getReception(i).getNumOfVisit();

}

return s;

}

public int minVisitors() {

Reception recep=getReception(0);

for (int i = 1; i < recepCount(); i++) {

if (recep.getNumOfVisit() > getReception(i).getNumOfVisit()) {

recep = getReception(i);

}

}

return recep.getNumOfVisit();

}

public void showData(String title) {

System.out.println(title);

System.out.println(" Имя : " + getName() + ", Специальность : " + getSpec());

for (int i = 0; i < recepCount(); i++) {

System.out.println(" Посетители : " + getReception(i).getNumOfVisit() + ", День : "

+ getReception(i).getDay() + ", Смена : " + getReception(i).getShift());

}

System.out.println();

}

public void testing() {

setName("Курченко");

setSpec("Терапефт");

addReception(new Reception(15, "12.10.17", "Утро"));

addReception(new Reception(20, "23.10.17", "Вечер"));

addReception(new Reception(13, "16.10.17", "Обед"));

showData("Введенные данные: ");

// System.out.println("");

sortByDay();

showData("Сортировка по дням: ");

// System.out.println("");

sortByVisit();

showData("Сортировка по посетителям: ");

System.out.println("Минимальное количество посетителей за смену : " + minVisitors());

System.out.println("Все посетители" + " " + allVisitors());

}

}

**RealizationByList**

package F\_ex;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class RealizationByList extends Doctor {

List<Reception> b = new ArrayList<>();

@Override

public Reception getReception(int i) {

return b.get(i);

}

@Override

public void setReception(int i, Reception recep) {

b.set(i, recep);

}

@Override

public int recepCount() {

return b.size();

}

@Override

public void addReception(Reception recep) {

b.add(recep);

}

@Override

public void sortByDay() {

Collections.sort(b, (a, b) -> a.getDay().compareTo(b.getDay()));

}

@Override

public void sortByVisit() {

Collections.sort(b, (b, a) -> Integer.compare(a.getNumOfVisit(), b.getNumOfVisit()));

}

public static void main(String[] args) {

new RealizationByList().testing();

}

@Override

public void clearRecep() {

// TODO Auto-generated method stub

}

}

**RealizationByArray**

package F\_ex;

import java.util.Arrays;

public class RealizByArr extends Doctor {

Reception []h = null;

@Override

public Reception getReception(int i) {

return h[i];

}

@Override

public void setReception(int i, Reception recep) {

h[i] = recep;

}

@Override

public int recepCount() {

return h == null ? 0 : h.length;

}

@Override

public void addReception(Reception recep) {

Reception[] temp = new Reception[recepCount() + 1];

if (h != null)

System.arraycopy(h, 0, temp, 0, h.length);

temp[recepCount()] = recep ;

h = temp;

}

@Override

public void sortByDay() {

Arrays.sort(h, (a, b) -> a.getDay().compareTo(b.getDay()));

}

@Override

public void sortByVisit() {

Arrays.sort(h, (a, b) -> Integer.compare(a.getNumOfVisit(), b.getNumOfVisit()));

}

public static void main(String[] args) {

new RealizByArr().testing();

}

@Override

public void clearRecep() {

// TODO Auto-generated method stub

}

}

**Reception**

package F\_ex;

public class Reception {

private int numOfVisit;

private String day;

private String shift;

public Reception(int numOfVisit, String day, String shift) {

this.numOfVisit = numOfVisit;

this.day = day;

this.shift = shift;

}

public int getNumOfVisit() {

return numOfVisit;

}

public void setNumOfVisit(int numOfVisit) {

this.numOfVisit = numOfVisit;

}

public String getDay() {

return day;

}

public void setDay(String day) {

this.day = day;

}

public String getShift() {

return shift;

}

public void setShift(String shift) {

this.shift = shift;

}

}

### 1.2 Мінімум функції

*Код програми 1.2:*

**Лямбда-вираз**

**Result**

**package Sec\_Ex;**

**import java.util.Scanner;**

**public class Result {**

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

**System.out.print("Enter the start of interval: ");String st=new Scanner(System.in).nextLine();**

**if(st.matches("\\D")){**

**do{**

**System.out.println("Not correct value. Try again: ");st=new Scanner(System.in).nextLine();}while(st.matches("\\D"));**

**}**

**double st1 = Double.parseDouble(st);**

**System.out.print("Enter the end of interval: ");String en=new Scanner(System.in).nextLine();**

**if(en.matches("\\D")){**

**do{**

**System.out.println("Not correct value. Try again: ");en=new Scanner(System.in).nextLine();}while(en.matches("\\D"));**

**}**

**double en1 = Double.parseDouble(en);**

**System.out.print("Enter the step: ");String step=new Scanner(System.in).nextLine();**

**if(step.matches("\\D")){**

**do{**

**System.out.println("Not correct value. Try again: ");step=new Scanner(System.in).nextLine();}while(step.matches("\\D"));**

**}**

**double step1 = Double.parseDouble(step);**

**if(en1<st1){**

**System.out.println("The start of interval under the end, they'll be swapped");**

**double temp = st1;**

**st1 = en1;**

**en1 = temp;**

**}**

**if(st==en){**

**System.out.println("It's no interval here,because start and end are same. There will be just a solution of equation.");**

**}**

**if(step1==0){**

**do{**

**System.out.println("Ridiculous!");**

**System.out.print("Enter the step of interval again: ");step1=new Scanner(System.in).nextDouble();**

**}while(step1<=0);**

**}**

**System.out.println("The minimum of our function on this interval is: "+minOfFunc.solve(st1, en1, step1, new F()));**

**System.out.println("The minimum of our function on this interval is: "+interFunc.solve(st1, en1, step1, x -> x\*x\*x - 2\*x + 15));**

**}**

**}**

**minOfFunc**

**package Sec\_Ex;**

**import java.util.\*;**

**public class minOfFunc {**

**static double solve(double a, double b, double step, FindOfFunc func) {**

**double x = 0;**

**double temp = 0;**

**for( ; a<=b;a+=step){**

**x = func.function(a);**

**temp = x;**

**if (x<temp)temp=x;**

**}**

**return temp;**

**}**

**}**

**interFunc**

**package Sec\_Ex;**

**public interface interFunc {**

**double f(double x);**

**static double solve(double a, double b, double step, FindOfFunc func) {**

**double x = 0;**

**double temp = 0;**

**for( ; a<=b;a+=step){**

**temp = func.function(a);**

**x = temp;**

**if (temp<x)x=temp;**

**}**

**return x;**

**}**

**}**

**FindOfFunc**

**package Sec\_Ex;**

**public interface FindOfFunc {**

**double function(double x);**

**}**

**F**

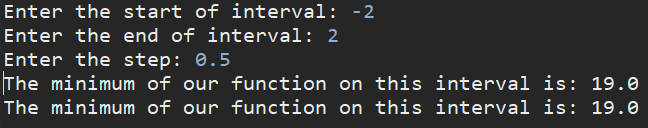
**package Sec\_Ex;**

**public class F implements FindOfFunc{**

**public double function(double x) {**

**return x\*x\*x - 2\*x + 15 ; }**

**}**



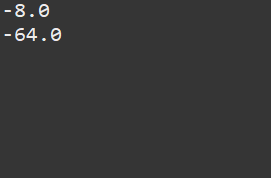
**Вказівники на методи**

**package** lab1.minvkazonmethod;  
  
**public interface** Func {  
 **double** f(**double** x);  
}

**package** lab1.minvkazonmethod;  
  
**public class** MinMethod {  
 **public double** min(**double** from, **double** to, **double** step, Func c) {  
 **if**(from <= to){  
 **double** min = c.f(from);  
 **for** (**double** x = from + step; x <= to; x += step) {  
 **if** (min > c.f(x)) {  
 min = c.f(x);  
 }  
 }  
 **return** min;  
 }  
 **else**{  
 System.***out***.println(**"Ошибка "**);  
 System.***out***.println(**"Начало отрезка меньше его конца "**);  
 **return** -1;  
 }  
 }  
  
 **public double** function(**double** x) {  
 **return** x \* x \* x;  
 }  
}

**package** lab1.minvkazonmethod;  
  
**public interface** MinOperation {  
 **double** min(**double** from, **double** to, **double** step, Func c);  
}

**package** lab1.minvkazonmethod;  
  
**public class** TestMethod {  
 **static void** print(MinOperation op, **double** from, **double** to, **double** step, Func c) {  
 System.***out***.println(op.min(from, to, step, c));  
 }  
  
 **static double** ret\_min(Func op, **double** p) {  
 **return** op.f(p);  
 }  
  
 **public static void** main(String[] args) {  
 MinMethod dm = **new** MinMethod();  
 *print*(dm::min, 2, 4,0.5, dm::function);  
 *print*(dm::min, 6, 4,0.5, dm::function);  
 }  
}



### 1.3 Узагальнений клас

*Код програми 1.3:*

**Mass**

**package Third\_ex;**

**import java.util.Arrays;**

**import java.util.Scanner;**

**import java.util.regex.Matcher;**

**import java.util.regex.Pattern;**

**public class Mass<T> {**

**Object[] mass;**

**public Mass(Object[] mass) {**

**super();**

**this.mass = mass;**

**}**

**int size(){**

**return mass.length;**

**}**

**public void addElement(Object someadd){**

**Object[] mass1 = Arrays.copyOf(mass,mass.length+1);**

**mass1[mass.length]=someadd;**

**mass=mass1;**

**}**

**void removeElement(int index){**

**Object[]mass1 = Arrays.copyOf(mass, mass.length-1);**

**boolean count =false;**

**for(int i=0;i<mass.length-1;i++)**

**{**

**if(index==i)**

**{**

**count=true;**

**mass1[i]=mass[i+1];**

**}**

**if(index!=i&&!count)**

**mass1[i]=mass[i];**

**if(index!=i&&count)**

**mass1[i]=mass[i+1];**

**}**

**mass=mass1;**

**}**

**void addGroup(int index, Object[] addmass){**

**Object[] mass1 = Arrays.copyOf(mass, mass.length+addmass.length);**

**for(int i=0;i<index;i++)**

**mass1[i]=mass[i];**

**for(int i=index,j=0;i<index+addmass.length;i++,j++)**

**mass1[i]=addmass[j];**

**for(int i=index+addmass.length,j=index;i<mass.length+addmass.length;i++,j++)**

**mass1[i]=mass[j];**

**mass=mass1;**

**}**

**void printAll() {**

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

**System.out.print(mass[i] + " ");**

**}**

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

**}**

**public static class Result {**

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

**Scanner s = new Scanner(System.in);**

**System.out.print("Введите размер массива: ");**

**String v=s.nextLine();Pattern p = Pattern.compile("-[0-9]+|[0]");Matcher m = p.matcher(v);**

**if(v.matches("\\D+")||m.find()){**

**do{**

**System.out.print("Не корректное значение. Попробуйте еще раз: ");v=new Scanner(System.in).nextLine();m = p.matcher(v);}while(v.matches("\\D+")||m.find());**

**}**

**int v1 = Integer.parseInt(v);**

**String[] f = new String[v1];**

**System.out.println("Введите элементы массива:");**

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

**f[i] = new Scanner(System.in).nextLine();**

**}**

**Mass<String> a = new Mass<String>(f);**

**System.out.println("Исходный массив");**

**a.printAll();**

**System.out.print("Введите элемент для добавления в конец массива: ");**

**String l = new Scanner(System.in).nextLine();**

**System.out.println("Добавление элемента в конец массива");**

**a.addElement(l);**

**a.printAll();**

**System.out.println("Введите индекс элемента для удаления из массива:");**

**String deleteindex=s.nextLine();Pattern p1 = Pattern.compile("-[0-9]+");Matcher m1 = p1.matcher(deleteindex);**

**if(deleteindex.matches("\\D+")||m1.find()){**

**do{**

**System.out.print("Не корректное значение. Попробуйте еще раз: ");**

**deleteindex=new Scanner(System.in).nextLine();m1 = p1.matcher(deleteindex);}while(deleteindex.matches("\\D+")||m1.find());**

**}**

**int ind = Integer.parseInt(deleteindex);**

**if(ind>=a.size()){**

**do{**

**System.out.println("Ошибка, индекс выходит за пределы массива, повторите попытку");**

**deleteindex=s.nextLine();**

**if(deleteindex.matches("\\D+")||m1.find()){**

**do{**

**System.out.print("Не корректное значение. Попробуйте еще раз: ");deleteindex=new Scanner(System.in).nextLine();m1 = p1.matcher(deleteindex);}while(v.matches("\\D+")||m1.find());**

**}**

**ind = Integer.parseInt(deleteindex);**

**}while(ind>=a.size());}**

**else{**

**System.out.println("Удаление элемента с индексом "+deleteindex);**

**a.removeElement(ind);**

**a.printAll();**

**}**

**System.out.print("Введите количество элементов группы, которые хотите добавить в массив: ");**

**String b=s.nextLine();Matcher m2 = p1.matcher(b);**

**if(b.matches("\\D+")||m2.find()){**

**do{**

**System.out.print("Не корректное значение. Попробуйте еще раз: ");b=new Scanner(System.in).nextLine();m2 = p1.matcher(b);}while(b.matches("\\D+")||m2.find());**

**}**

**int b1 = Integer.parseInt(b);**

**String c[] = new String[b1];**

**System.out.println("Введите элементы данной группы:");**

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

**c[i] = new Scanner(System.in).nextLine();**

**}**

**System.out.println("Введите индекс места для добавления группы элементов в массив: ");**

**String e=s.nextLine();Matcher m3 = p1.matcher(e);**

**if(e.matches("\\D+")||m3.find()){**

**do{**

**System.out.print("Не корректное значение. Попробуйте еще раз: ");e=new Scanner(System.in).nextLine();m3 = p1.matcher(e);}while(e.matches("\\D+")||m3.find());**

**}**

**int e1 = Integer.parseInt(e);**

**if(e1>f.length){do{**

**System.out.println("Ошибка, индекс выходит за пределы массива, повторите попытку");**

**e=s.nextLine();**

**if(e.matches("\\D+")||m3.find()){**

**do{**

**System.out.print("Не корректное значение. Попробуйте еще раз: ");e=new Scanner(System.in).nextLine();m3 = p1.matcher(e);}while(e.matches("\\D+")||m3.find());**

**}**

**e1 = Integer.parseInt(e);}while(e1>f.length);**

**System.out.println("Добавление группы элементов в массив: ");**

**a.addGroup(e1, c);**

**a.printAll();**

**}**

**else{**

**System.out.println("Добавление группы элементов в массив: ");**

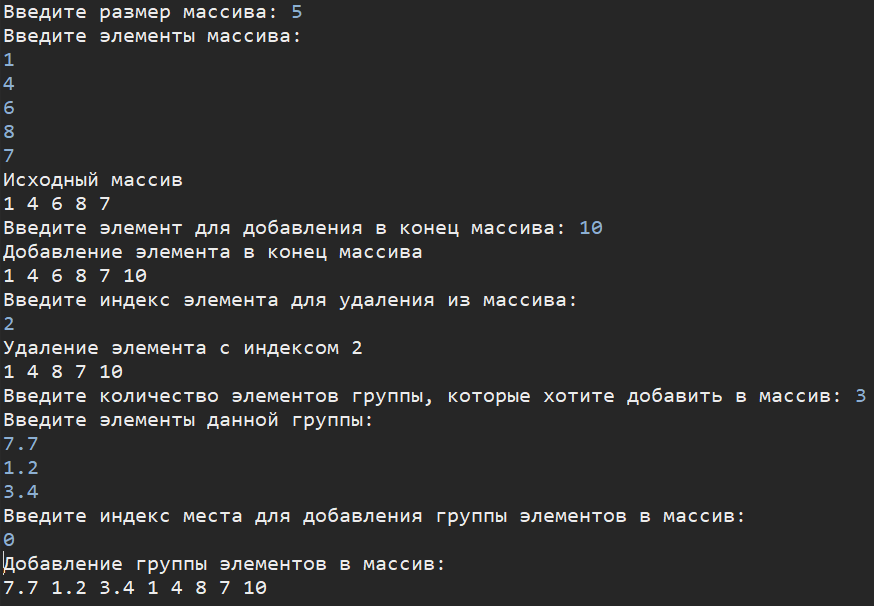
**a.addGroup(e1, c);**

**a.printAll();}**

**}**

**}**

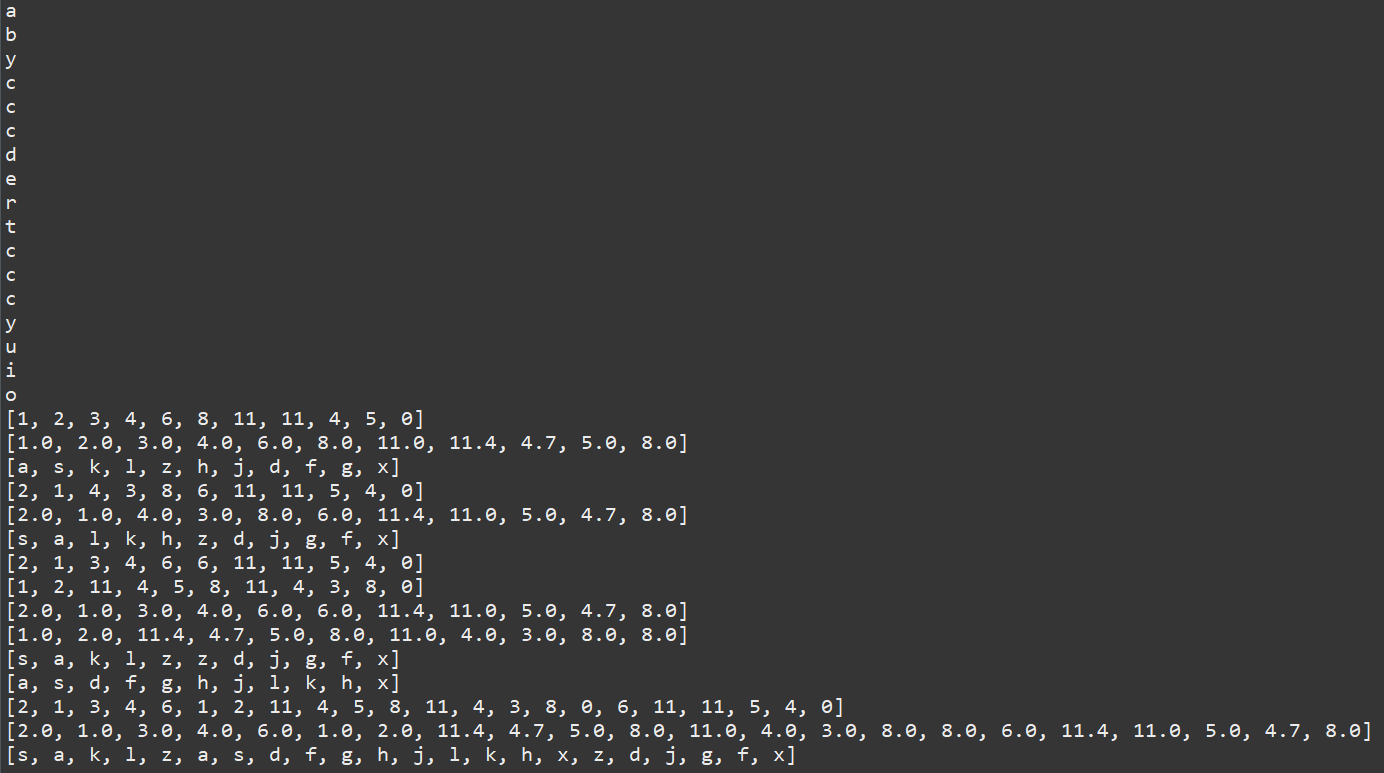
**}**



### 1.4 Створення бібліотеки узагальнених функцій для роботи з масивами та списками

*Код програми 1.4:*

**package** lab1.uzagal;  
  
**import** java.util.Arrays;  
**import** java.util.\*;  
  
**public class** UzagArray {  
 **public static** <E> **void** ZamenaGroup(List<E> arr, **int** ot1, **int** do1, **int** ot2, **int** do2) {  
 **for** (**int** i = 0; i <= do1 - ot1; i++) {  
 E temp = arr.get(ot2 + i);  
 arr.add(ot2 + i,arr.get(ot1 + i));  
 arr.add(ot1 + i,temp);  
 }  
 **for** (**int** i = 0; i < arr.size(); i++) {  
 System.***out***.println(arr.get(i));  
 }  
 }  
 **public static** <E> **void** ZamenaGroup(E[] arr, **int** ot1, **int** do1, **int** ot2, **int** do2) {  
 **for** (**int** i = 0; i <= do1 - ot1; i++) {  
 E temp = arr[ot2 + i];  
 arr[ot2 + i] = arr[ot1 + i];  
 arr[ot1 + i] = temp;  
 }  
 System.***out***.println(Arrays.*toString*(arr));  
 }  
  
 **public static** <E> **void** ParAndNepar(List<E> arr) {  
 **int** x = arr.size();  
 **if** (arr.size() % 2 != 0) {  
 x -= 1;  
 }  
 **for** (**int** i = 0; i < x; i += 2) {  
 E temp = arr.get(i);  
 arr.add(i, arr.get(i + 1));  
 arr.add(i + 1, temp);  
 }  
 **for** (**int** i = 0; i < arr.size(); i++) {  
 System.***out***.println(arr.get(i));  
 }  
 }  
 **public static** <E> **void** ParAndNepar(E[] arr) {  
 **int** x = arr.**length**;  
 **if** (arr.**length** % 2 != 0) {  
 x -= 1;  
 }  
 **for** (**int** i = 0; i < x; i += 2) {  
 E temp = arr[i];  
 arr[i] = arr[i + 1];  
 arr[i + 1] = temp;  
 }  
 System.***out***.println(Arrays.*toString*(arr));  
 }  
  
 **public static** <E> **void** ZamenaGroupArray(List<E> arr1, List<E> arr2, **int** ot1, **int** do1, **int** ot2, **int** do2) {  
 **for** (**int** i = 0; i <= do1 - ot1; i++) {  
 E temp = arr2.get(ot2 + i);  
 arr2.add(ot2 + i, arr1.get(ot1 + i));  
 arr1.add(ot1 + i, temp);  
  
 }  
 **for** (**int** i = 0; i < arr1.size(); i++) {  
 System.***out***.println(arr1.get(i));  
 }  
 **for** (**int** i = 0; i < arr2.size(); i++) {  
 System.***out***.println(arr2.get(i));  
 }  
 }  
 **public static** <E> **void** ZamenaGroupArray(E[] arr1, E[] arr2, **int** ot1, **int** do1, **int** ot2, **int** do2) {  
 **for** (**int** i = 0; i <= do1 - ot1; i++) {  
 E temp = arr2[ot2 + i];  
 arr2[ot2 + i] = arr1[ot1 + i];  
 arr1[ot1 + i] = temp;  
  
 }  
 System.***out***.println(Arrays.*toString*(arr1));  
 System.***out***.println(Arrays.*toString*(arr2));  
 }  
  
 **public static** <E> **void** InsertArray(List<E> arr1, List<E> arr2, **int** psn) {  
 List<E> temp = **new** ArrayList<E>();  
 **for** (**int** i = 0; i < psn; i++) {  
 temp.add(i, arr1.get(i));  
 }  
 **int** t = 0;  
 **for** (**int** i = psn; i < arr2.size() + psn; i++) {  
 temp.add(i, arr2.get(t));  
 t++;  
 }  
 t = psn;  
 **for** (**int** i = arr2.size() + psn; i < arr1.size() + arr2.size(); i++) {  
 temp.add(i, arr1.get(t));  
 t++;  
 }  
 arr1 = temp;  
 **for** (**int** i = 0; i < arr1.size(); i++) {  
 System.***out***.println(arr1.get(i));  
 }  
 }  
 **public static** <E> **void** InsertArray(E[] arr1, E[] arr2, **int** psn) {  
 Object[] temp = **new** Object[arr1.**length** + arr2.**length**];  
 **for** (**int** i = 0; i < psn; i++) {  
 temp[i] = arr1[i];  
 }  
 **int** t = 0;  
 **for** (**int** i = psn; i < arr2.**length** + psn; i++) {  
 temp[i] = arr2[t];  
 t++;  
 }  
 t = psn;  
 **for** (**int** i = arr2.**length** + psn; i < arr1.**length** + arr2.**length**; i++) {  
 temp[i] = arr1[t];  
 t++;  
 }  
 arr1 = (E[]) temp;  
 System.***out***.println(Arrays.*toString*(arr1));  
 }  
 **public static void** main(String[] args) {  
 Integer[] a = {1, 2, 11, 4, 5, 8, 11, 3, 4, 6, 0};  
 Integer[] d = {1, 2, 11, 4, 5, 8, 11, 3, 4, 6, 0};  
 List<String> list = **new** ArrayList<String>();  
 list.add(**"a"**);  
 list.add(**"b"**);  
 list.add(**"c"**);  
 list.add(**"d"**);  
 list.add(**"e"**);  
 list.add(**"r"**);  
 list.add(**"t"**);  
 list.add(**"y"**);  
 list.add(**"u"**);  
 list.add(**"i"**);  
 list.add(**"o"**);  
 Double[] b = {1.0, 2.0, 11.4, 4.7, 5.0, 8.0, 11.0, 3.0, 4.0, 6.0, 8.0};  
 Double[] f = {1.0, 2.0, 11.4, 4.7, 5.0, 8.0, 11.0, 3.0, 4.0, 6.0, 8.0};  
 String[] c = {**"a"**, **"s"**, **"d"**, **"f"**, **"g"**, **"h"**, **"j"**, **"k"**, **"l"**, **"z"**, **"x"**};  
 String[] e = {**"a"**, **"s"**, **"d"**, **"f"**, **"g"**, **"h"**, **"j"**, **"k"**, **"l"**, **"z"**, **"x"**};  
 *ZamenaGroup*(list, 2, 4, 7, 9);  
 *ZamenaGroup*(a, 2, 4, 7, 9);  
 *ZamenaGroup*(b, 2, 4, 7, 9);  
 *ZamenaGroup*(c, 2, 4, 7, 9);  
 *ParAndNepar*(a);  
 *ParAndNepar*(b);  
 *ParAndNepar*(c);  
 *ZamenaGroupArray*(a, d, 2, 4, 7, 9);  
 *ZamenaGroupArray*(b, f, 2, 4, 7, 9);  
 *ZamenaGroupArray*(c, e, 2, 4, 7, 9);  
 *InsertArray*(a, d, 5);  
 *InsertArray*(b, f, 5);  
 *InsertArray*(c, e, 5);  
 }  
}



### 1.5 Реалізація інтерфейсу Comparable

*Код програми 1.5:*

**Circle**

package Fif\_ex;

import java.util.Arrays;

public class Circle implements Comparable<Circle> {

double rad;

Circle(double rad) {

this.rad = rad;

}

public int compareTo(Circle r) {

if (rad == r.rad)

return 0;

else if (rad > r.rad)

return 1;

else

return -1;

}

public double getRad() {

return rad;

}

public void setRad(double rad) {

this.rad = rad;

}

@Override

public String toString() {

String s = "" + getRad();

return s;

}

}

**Test**

package Fif\_ex;

import java.awt.List;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.HashMap;

import java.util.HashSet;

import java.util.LinkedHashSet;

import java.util.Scanner;

import java.util.Set;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class Test {

public static void main(String[] args) throws Exception {

ArrayList<Circle> al = new ArrayList<Circle>();

ArrayList<String> al1 = new ArrayList<String>();

System.out.println("Введите кол-во окружностей:");

@SuppressWarnings("resource")

String s = new Scanner(System.in).nextLine();

Pattern p = Pattern.compile("-[0-9]+|[0]");

Matcher m = p.matcher(s);

if (s.matches("(\\D|\\s)+") || m.find()) {

do {

System.out.print("Не корректное значение. Попробуйте еще раз: ");

s = new Scanner(System.in).nextLine();

m = p.matcher(s);

} while (s.matches("(\\D|\\s)+") || m.find());

}

int size = Integer.parseInt(s);

System.out.println("Введите радиусы окружностей(Integer or Double):");

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

al1.add(new Scanner(System.in).nextLine());

Matcher m1 = p.matcher(al1.get(i));

if (al1.get(i).matches("(\\D|\\s)+") || m1.find()) {

do {

System.out.print("Не корректное значение. Попробуйте еще раз: ");

al1.remove(al1.get(i));

al1.add(new Scanner(System.in).nextLine());

m1 = p.matcher(al1.get(i));

} while (al1.get(i).matches("(\\D|\\s)+") || m1.find());

}

}

for (int i = 0; i < al1.size(); i++) {

al.add(new Circle(Double.parseDouble(al1.get(i))));

}

System.out.print("Введенные радиусы: ");

for (Circle st : al) {

System.out.print(st.rad + " ");

}

Collections.sort(al);

Set<String> a3=new LinkedHashSet<>();

for (Circle string : al) {

a3.add(string.rad+"");

}

al.clear();

for (String string : a3) {

al.add(new Circle(Double.parseDouble(string)));

}

System.out.println();

System.out.println();

System.out.print("Отсортированные радиусы: ");

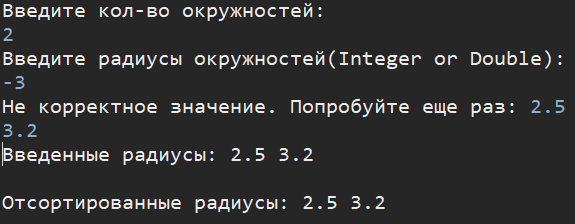
for (Circle st1 : al) {

System.out.print(st1.rad + " ");

}

}

}



### 1.6 Реалізація інтерфейсу Comparator

*Код програми 1.6:*

**Triangle**

package Six\_ex;

import java.util.\*;

public class Triangle {

private double a;

private double b;

private double c;

public double getA() {

return a;

}

public double getB() {

return b;

}

public double getC(){

return c;

}

public void setA(double a) {

this.a = a;

}

public void setB(double b) {

this.b = b;

}

public void setC(double c) {

this.c = c;

}

public Triangle(double a, double b, double c) {

super();

this.a = a;

this.b = b;

this.c = c;

}

public double squareT() {

if (a < 0 || b < 0 || c < 0){

System.out.println("Введено отрицательное значение!");

}

else {

double s = 0.25\*Math.sqrt((a+b+c)\*(a+b-c)\*(a+c-b)\*(b+c-a));

return s;

}

return 0;

}

@Override

public String toString() {

String s=""+(0.25)\*Math.sqrt(( getA()+getB()+getC())\*( getB()+getC()-getA())\*(getA()+getC()-getB())\*( getA()+getB()-getC()));

return s;

}

public static void main(String args[]) {

ArrayList<Triangle> list = new ArrayList<Triangle>();

list.add(new Triangle(20, 15, 25));

list.add(new Triangle(7, 8, 10));

list.add(new Triangle(8, 5, 12));

Collections.sort(list, new CompareBySqr());

System.out.println("Отсортированные площади треугольников: ");

for(Triangle lists:list){

System.out.println(lists+" ");

}

}

**}**

**CompareBySqr**

package Six\_ex;

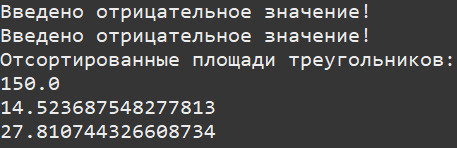
import java.util.Comparator;

public class CompareBySqr implements Comparator<Triangle> {

public int compare(Triangle s1, Triangle s2) {

return Double.compare(s1.squareT(), s2.squareT());

}



***Висновок:***

У даній роботі я за допомогою мови програмування Java та використовуючи попередній досвід я розробив різного роду програми, під час розробки яких я навчився працювати з узагальненнями та колекціями в Java.