МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «Програмна інженерія та інформаційні технології управління»

Звіт з лабораторної роботи №1
З предмету «Об’єктно-орієнтоване програмування»

Виконав
Студент групи КН-36а
Рубан Ю.Д.
Перевірив:
Козуля М.М.
Кізілов О.С.

Харків 2017

Робота з узагальненнями та колекціями в Java

1 Завдання на лабораторну роботу

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), 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).

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)
- вставлення у масив (список) іншого масиву (списку) елементів у вказане місце
- заміна групи елементів іншим масивом (списком) елементів

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:

$$S_{\triangle} = \frac{1}{4}\sqrt{(a+b+c)(b+c-a)(a+c-b)(a+b-c)}$$

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.

1.7 Реалізація масиву точок за допомогою списку дійсних чисел (додаткове завдання)

Реалізувати функціональність абстрактного класу AbstractArrayOfPoints, наведеного в прикладі лабораторної роботи № 6 курсу "Алгоритмізація та програмування" (друга частина), через використання списку дійсних чисел. Кожна пара чисел у масиві має відповідати точці.

1.8 Пошук різних слів у реченні (додаткове завдання)

Увести речення, створити список (SortedSet) різних слів речення та вивести ці слова в алфавітному порядку.

2 Хід виконання роботи

2.1 Індивідуальне завдання

Розроблено 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), містить абстрактні функції для доступу до елементів типу другого класу попередньої роботи та сортує ці елементи за певною ознакою. Похідні класи від створеного абстрактного класу відповідно представляють послідовність елементів у вигляді масиву та списку. Здійснено тестування обох реалізацій. Тест включає виконання завдання попередньої лабораторної роботи, а також сортування за визначеними ознаками відповідно до завдання лабораторної роботи № 6 курсу "Алгоритмізація та програмування" (друга частина).

Для сортування використано методи sort() класів Arrays та Collections відповідно. Для визначення ознак сортування використно лямбда-вирази.

Код програми:

Клас Main:

```
import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        ArrayList<Day> days =new ArrayList<>();
        Day d1=new Day(30,"день 1");
        Day d2=new Day(25 ,"на след день получился большой коментарий");
        Day d3=new Day(26 ,"AAAAA");
        days.add(d1);
        ShowList s=new ShowList("выставка","автор",days);
        s.pushDay(d2);
        s.pushDay(d3);
        s.sortViaVisitors();
        System.out.println("LIST\n"+s);
        Day[]arrDay = new Day[3];
        arrDay[0] = new Day(23,"Семен");
        arrDay[1] = new Day(64,"тест");
        arrDay[2] = new Day(11,"гламбург");
        ShowArray sa = new ShowArray("коммуникация","ancient",arrDay);
        sa.sortViaABC();
        System.out.print("ARRAY\n"+sa);
       }
}
```

Клас AbsShow:

```
public abstract class AbsShow {
    abstract void pushDay(Day d);
    void setName(String n){name = n;}
    String getName(){return name;}
    void setTitle(String t){title = t;}
    String getTitle(){return title;}
    abstract int visitors();
    abstract int maxVisitors();
    abstract int dayWithLargeComment();
    abstract void sortViaVisitors();
    abstract void sortViaABC();
    protected String title;
    protected String name;

}
```

Клас Day:

```
public class Day
{
    Day(int count,String c)
    {
        this.comment=c;
        this.countOfvisitors=count;
    }

    @Override
    public String toString() {
        return "Day{" +
                "countOfvisitors=" + countOfvisitors +
                ", comment='" + comment + '\'' +
                "}"+'\n';
    }
    boolean equals(Day obj)
    {
        if(countOfvisitors == obj.countOfvisitors && comment == obj.comment)
        {
            return true;
        }
        else
        {
            return false;
        }
    }

    public void setComment(String c) {
        this.comment = c;
    }

    public void setCountOfvisitors(int count) {
        this.countOfvisitors = count;
    }

    public int getCountOfvisitors() {
        return this.countOfvisitors;
    }

    public String getComment() {
        return this.comment;
    }

    private int countOfvisitors;
    private String comment;

}
```

Клас ShowList:

```
import java.util.*;
public class ShowList extends AbsShow
{
    private ArrayList<Day>D;
    ShowList(){};
    ShowList(String t,String n,ArrayList<Day> d)
    {
        this.title = t;
        this.name = n;
        this.D=d;
    }
    void pushDay(Day d)
    {
        boolean check = false;
        for (Day t:D)
        {
            if(t.equals(d))
            {
                check = true;
            }
        }
        if(!check)
        {
            D.add(D.size(), d);
        }
    }
    @Override
    public String toString() {
        return "Show{" +
                "Days=" + D +
                ", title='" + title + '\'' +
                ", name='" + name + '\'' +
                '}'+'\n';
    }
    boolean equals(ShowList obj)
    {
        if(name == obj.name && title == obj.title)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    public int visitors()
    {
        int count = 0;
        for(Day d : D)
        {
            count+=d.getCountOfvisitors();
        }
        return count;
    }
    public int maxVisitors()
    {
        int max = 0;
        for(Day d : D)
        {
            if(max<d.getCountOfvisitors())
            {
                max = d.getCountOfvisitors();
            }
        }
        return max;
    }
    public int dayWithLargeComment()
    {
```

```
        int max = 0;
       int index=0;
        for(int i =0;i<D.size();i++)
        {
            if(max<D.get(i).getComment().length())
            {
                max = D.get(i).getComment().length();
                index = i;
            }
        }
        return index;
    }
    public void sortViaVisitors()
    {
        Collections.sort(D,(Day o1,Day o2)-
>(Integer.compare(o2.getCountOfvisitors(),o1.getCountOfvisitors())));
    }
    public void sortViaABC()
    {
        Collections.sort(D,(Day o1,Day o2)-
>o1.getComment().compareTo(o2.getComment()));
    }
}
```

Клас ShowArray:

```
import java.util.Arrays;

public class ShowArray extends AbsShow {
    private Day[] D;
    ShowArray(String t, String n, Day[] d) {
        this.title = t;
        this.name = n;
        this.D = d;
    }

    void pushDay(Day d) {
        boolean check = false;
        for (Day t : D) {
            if (t.equals(d)) {
                check = true;
            }
        }
        if (!check) {
            Day[] temp = new Day[D.length + 1];
            System.arraycopy(D, 0, temp, 0, D.length);
            temp[D.length] = d;
            D = temp;
        }
    }

    @Override
    public String toString() {
        return "Show{" +
                "Days=" + show() +
                ", title='" + title + '\'' +
                ", name='" + name + '\'' +
                '}';
    }
    private String show()
    {
        String output="[";
        for(int i = 0;i<D.length;i++)
        {
            output += D[i];
            if(i<D.length-1)
            {
                output +=", ";
            }
        }
```

```
        output+="]";
        return output;
    }
    boolean equals(ShowList obj) {
        if (name == obj.name && title == obj.title) {
            return true;
        } else {
            return false;
        }
    }
    public int visitors()
    {
        int count = 0;
        for(Day d : D)
        {
            count+=d.getCountOfvisitors();
        }
        return count;
    }
    public int maxVisitors()
    {
        int max = 0;
        for(Day d : D)
        {
            if(max<d.getCountOfvisitors())
            {
                max = d.getCountOfvisitors();
            }
        }
        return max;
    }
    public int dayWithLargeComment()
    {
        int max = 0;
        int index=0;
        for(int i =0;i<D.length;i++)
        {
            if(max<D[i].getComment().length())
            {
                max = D[i].getComment().length();
                index = i;
            }
        }
        return index;
    }
    public void sortViaVisitors()
    {
        Arrays.sort(D,(Day o1, Day o2)-
>(Integer.compare(o2.getCountOfvisitors(),o1.getCountOfvisitors())));
    }
    public void sortViaABC()
    {
        Arrays.sort(D,(Day o1,Day o2)->o1.getComment().compareTo(o2.getComment()));
    }
}
```

2.2 Мінімум функції

Реалізовано 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.

Знайдено та застосовано відповідний стандартний функціональний інтерфейс для опису функції.

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

Код програми:

Клас main:

```
public class main {
    public static void main(String[] args) {
        MinFunk obj = new MinFunk();
        System.out.println(obj.minimum(x->Math.sin(x),1,5));
        System.out.println(obj.minimum(Math::cos,1,5));
    }
}
```

Клас MinFunk:

```
import java.util.function.Function;

public class MinFunk {
    public double minimum(Function<Double,Double> funk,double begin,double
end)
    {
        double min = funk.apply(begin);
        for(double i = begin;i<=end;i+=0.0001)
        {
            if(funk.apply(i)<min)
            {
                min=funk.apply(i);
            }
        }
        return min;
    }
}
```

2.3 Узагальнений клас

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

Код програми:

Клас main:

```
public class main {
    public static void main(String[] args) {
        ArrayTop<Integer>arr = new ArrayTop<>(5);
        for(Integer i =0;i<arr.len();i++)
        {
            arr.set(i,i+1);
        }
        ArrayTop<Integer>arr2 = new ArrayTop<>(5);
        for(Integer i =0;i<arr2.len();i++)
        {
            arr2.set(i,i+1);
        }
        arr.group(arr2,5);
        arr.delIndex(9);
        arr.add(10);
        for(int i = 0;i<arr.len();i++)
        {
            System.out.print(arr.get(i)+" ");
        }
    }
}
```

Клас ArrayTop:

```
import java.lang.reflect.Array;

public class ArrayTop<T>
{
    private Object[] arr = {};
    ArrayTop(int size)
    {
         arr = new Object[size];
    }
    public void add(T elem)
    {
        Object []temp = new Object[arr.length+1];
        System.arraycopy(arr,0,temp,0,arr.length);
        arr=temp;
        arr[arr.length-1]=elem;
    }
    public void del()
    {
        if(arr.length!=0) {
            Object[] temp = new Object[arr.length - 1];
            System.arraycopy(arr, 0, temp, 0, arr.length - 1);
            arr = temp;
        }
    }
```

```
    public void delIndex(int index)
    {
        Object[] temp = new Object[arr.length-1];
        for(int i = 0;i<index;i++)
        {
            temp[i] = arr[i];
        }
        for(int i = index+1;i<arr.length;i++)
        {
            temp[i-1] = arr[i];
        }
        arr=temp;
    }
    public T get(int i)
    {
        return (T)arr[i];
    }
    public void set(int i, T elem)
    {
        arr[i]=elem;
    }
    public int len(){return arr.length;}
    public void group(ArrayTop<T>arra, int begin)
    {
        Object[]array = arra.arr;
        int newSize = array.length + arr.length;
        Object[] tmp = new Object[newSize];
        int arrSize = array.length;
        int k = 0;
        int l = 0;
        boolean flag = false;
        for(int i =0;i<newSize;i++)
        {
            if(i!=begin && !flag)
            {
                tmp[i] = arr[l];
                l++;
            }
            else if(i==begin)
            {
                flag = true;
            }
            if(flag)
            {
                tmp[i] = array[k];
                k++;
                arrSize--;
            }
            if(arrSize==0){flag=false;}
        }
        arr = tmp;
    }
}
```

2.4 Створення бібліотеки узагальнених функцій для роботи з масивами та списками

Реалізовано 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)
- вставлення у масив (список) іншого масиву (списку) елементів у вказане місце
- заміна групи елементів іншим масивом (списком) елементів

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

Код програми:

Клас Main:

```
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
public class Main {
    public static void main(String[] args) {
        TopFunctional a = new TopFunctional();
        List<Integer> b = new LinkedList<>();
        List<Integer> c = new LinkedList<>();
        List<Integer> d = new LinkedList<>();
        Object []e = new Double[3];
        Object []f = new Double[3];
        b.add(4);
        b.add(5);
        b.add(6);
        b.add(1);
        b.add(2);
        b.add(3);
        c.add(8);
        c.add(7);
        d.add(99);
        d.add(100);
        d.add(101);
        e[0] = new Double(3);
        e[1] = new Double(2);
        e[2] = new Double(1);
        f[0] = new Double(12);
        f[1] = new Double(13);
        f[2] = new Double(14);
        b=a.groupChange(b,0,3,3);
        b=a.indexChange(b);
        b=a.addGroup(b,c,6);
        b=a.replaceGroup(b,d,0);
        e=a.addGroup(e,e,1);
        e=a.replaceGroup(e,f,2);
        e=a.indexChange(e);
```

```
        e=a.groupChange(e,0,2,2);
        System.out.println("LIST");
        for(Object i : b)
        {
            System.out.println(i);
        }
        System.out.println("ARRAY");
        for(Object i : e)
        {
            System.out.println(i);
        }
    }
}
```

Клас TopFunctional:

```
import java.util.List;

public class TopFunctional <T>{
    public List<?> groupChange(List<T> array, int first_begin, int
second_begin, int count)
    {
       T temp;
       for(int i =0;i<count;i++)
       {
            temp = array.get(i+first_begin);
            array.set(i+first_begin,array.get(i+second_begin));
            array.set(i+second_begin, temp);
       }
       return array;
    }
    public List<?> indexChange(List<T> array)
    {
        T temp;
        if(array.size()%2==0){
        for(int i=0;i<array.size();i+=2)
        {
            temp = array.get(i);
            array.set(i,array.get(i+1));
            array.set(i+1,temp);
        }
        }
        else
            {
                for(int i=0;i<array.size()-1;i+=2)
                {
                    temp = array.get(i);
                    array.set(i,array.get(i+1));
                    array.set(i+1,temp);
                }
            }

        return array;
    }
    public List<?> addGroup(List<T>array,List<T>buffer,int place)
    {
        array.addAll(place,buffer);
        return array;
    }
    public List<?> replaceGroup(List<T>array,List<T>buffer,int begin)
    {
        for(int i=begin,k=0;k<buffer.size() && i<array.size();i++,k++)
```

```
        {
            array.set(i,buffer.get(k));
        }
        return array;
    }
    public T[] groupChange(T[] array, int first_begin, int second_begin, int
count)
    {
        T temp;
        for(int i =0;i<count;i++)
        {
            temp = array[i+first_begin];
            array[i+first_begin] = array[i+second_begin];
            array[i+second_begin] =  temp;
        }
        return array;
    }
    public T[] indexChange(T[] array)
    {
        T temp;
        if(array.length%2==0){
            for(int i=0;i<array.length;i+=2)
            {
                temp = array[i];
                array[i] = array[i+1];
                array[i+1] = temp;
            }
        }
        else
        {
            for(int i=0;i<array.length-1;i+=2)
            {
                temp = array[i];
                array[i] = array[i+1];
                array[i+1] = temp;
            }
        }

        return array;
    }
    public T[] addGroup(T[] array,T[] buffer,int place)
    {
        int newSize = array.length + buffer.length;
        T[] temp = (T[])new Object[newSize];
        for(int i =0,l=0,k=0;i<newSize;i++)
        {
            if(i<place)
            {
                temp[i] = array[k];
                k++;
            }
            else if(i>=place && i<=buffer.length)
            {
                temp[i]=buffer[l];
                l++;
            }
            else if(i>=place && i>=buffer.length && k<array.length)
            {
                temp[i] = array[k];
                k++;
            }

        }
        array= temp;
```

```
        return  array;
    }
    public T[] replaceGroup(T[] array,T[] buffer,int begin)
    {
        for(int i=begin,k=0;k<buffer.length && i<array.length;i++,k++)
        {
            array[i]=buffer[k];
        }
        return array;
    }
}
```

2.5 Реалізація інтерфейсу Comparable

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

Код програми:

Клас Main:

```
import java.util.ArrayList;
import java.util.Collections;

public class Main {
    public static void main(String[] args) {
        ArrayList<Circle> circle = new ArrayList<>();
        circle.add(new Circle(3));
        circle.add(new Circle(1));
        circle.add(new Circle(4));
        circle.add(new Circle(6));
        circle.add(new Circle(8));
        Collections.sort(circle);
        for(Circle o : circle)
        {
            System.out.print(o);
        }
    }
}
```

Клас Circle:

```

public class Circle implements Comparable<Circle>
{
    private Double radius;
    Circle(double rad)
    {
        radius = rad;
    }
    @Override
    public int compareTo(Circle o) {
        return radius.compareTo(o.radius);
    }

    @Override
    public String toString() {
        return "radius = "+ radius + "\n";
    }
}
```

2.6 Реалізація інтерфейсу Comparator

Створено клас Triangle. Трикутник визначається довжинами сторін. Площа трикутника в цьому випадку обчислюється за формулою Герона:

$$S_{\triangle} = \frac{1}{4}\sqrt{(a+b+c)(b+c-a)(a+c-b)(a+b-c)}$$

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

Код програми:

Клас Main:

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        ArrayList<Triangle> list = new ArrayList<>();
        list.add(new Triangle(3,5,7));
        list.add(new Triangle(7,4,9));
        list.add(new Triangle(17,16,14));
        list.add(new Triangle(4,8,5));
        list.add(new Triangle(6,3,7));
        for(Triangle o : list)
        {
            System.out.println(o);
        }
        System.out.println("after sort:");
        Collections.sort(list, new Comparator<Triangle>() {
            @Override
            public int compare(Triangle o1, Triangle o2) {
                return o2.square.compareTo(o1.square);
            }
        });
        for(Triangle o : list)
        {
            System.out.println(o);
        }
    }
}
```

Клас Triangle:

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        ArrayList<Triangle> list = new ArrayList<>();
        list.add(new Triangle(3,5,7));
        list.add(new Triangle(7,4,9));
        list.add(new Triangle(17,16,14));
        list.add(new Triangle(4,8,5));
        list.add(new Triangle(6,3,7));
        for(Triangle o : list)
        {
            System.out.println(o);
```

```
        }
        System.out.println("after sort:");
        Collections.sort(list, new Comparator<Triangle>() {
            @Override
            public int compare(Triangle o1, Triangle o2) {
                return o2.square.compareTo(o1.square);
            }
        });
        for(Triangle o : list)
        {
            System.out.println(o);
        }
    }
}
```

2.7 Реалізація масиву точок за допомогою списку дійсних чисел (додаткове завдання)

Реалізувано функціональність абстрактного класу AbstractArrayOfPoints, наведеного в прикладі лабораторної роботи № 6 курсу "Алгоритмізація та програмування" (друга частина), через використання списку дійсних чисел. Кожна пара чисел у масиві відповідає точці.

Код програми:

Клас Main:

```
public class Main {
    public static void main(String[] args) {
        ArrayOfPoints points = new ArrayOfPoints();
        points.test();
    }
}
```

Клас ArrayOfPoints:

```
import java.util.ArrayList;
import java.util.List;

public class ArrayOfPoints extends AbstractArrayOfPoints
{
    private List<Double> points;
    ArrayOfPoints()
    {
       points = new ArrayList<>();
    }
    public void setPoint(int i, double x, double y)
    {
        points.set(2*i,x);
        points.set(2*i+1,y);
    }

    // Отримання X точки i:
    public double getX(int i)
    {
        return points.get(2*i);
    }
```

```
    // Отримання Y точки i:
    public double getY(int i)
    {
        return points.get(2*i+1);
    }

    // Отримання кількості точок:
    public  int count()
    {
        return points.size()/2;
    }

    // Додавання точки в кінець масиву:
    public  void addPoint(double x, double y)
    {
        points.add(x);
        points.add(y);
    }

    // Видалення останньої точки:
    public  void removeLast()
    {
        points.remove(points.size()-1);
        points.remove(points.size()-1);
    }
}
```

2.8 Пошук різних слів у реченні (додаткове завдання)

Увести речення, створити список (SortedSet) різних слів речення та вивести ці слова в алфавітному порядку.

Код програми:

Клас Main:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc =new Scanner(System.in);
        String sentence = sc.nextLine();
        WordFinder f = new WordFinder();
        f.findWord(sentence);
    }
}
```

Клас WordFinder:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.TreeSet;

public class WordFinder {
    public void findWord(String sentence)
    {
```

```
        TreeSet<String>s=new TreeSet<>();
        ArrayList<Character>del =  new ArrayList<>(Arrays.asList(' ', '.',
',', ':', ';', '?', '!', '-', '(', ')', '\"'));
        String[] split = sentence.split(" ");
        for(String str : split)
        {
            if(!del.contains(str))
            {
                s.add(str);
            }
        }
        System.out.print(s);
    }
}
```

3 Висновки

У даній лабораторній роботі я навчився працювати з узагальненнями та колекціями. Виконавши завдання, я закріпив знання теоретичного матеріалу та покращив свої знання з мови програмування JAVA.