Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №1 дисциплины «Разработка программных систем»

Выполнил студент группы ИВТ-31_____/Крючков И. С/
Проверил_____/Чистяков Г. А./

Киров 2023

1. Задание

Разработать класс, реализующий функционал по выполнению интервальных операций посредством sqrt-декомпозиции.

Требуется реализовать методы для:

- изменения значения в заданной точке;
- изменения значений на интервале;
- определения суммы значений на интервале.

Класс должен корректно работать со всеми примитивными числовыми типами данных. В случае возникновения нештатной ситуации должно инициироваться исключение.

2. Листинг программы

Листинг программной реализации приведен в приложении А.

3. Вывод

В ходе выполнения лабораторной работы были изучены основные конструкции языка программирования Java, структура программы, стандартные средства ввода/вывода, механизм исключений. Разработана программа, выполняющая интервальные операции посредством sqrt-декомпозиции.

Приложение А.

Листинг программы

Lab1.java

```
package rpslab1;

import java.util.Scanner;
import java.util.Locale;
import java.io.FileNotFoundException;
import java.util.InputMismatchException;

public class Lab1{
    private static Decomposition dn;
    private static Scanner reader;

    public static void main(String args[]) {

        try {
            dn = new Decomposition("input.txt");
        } catch (DecompositionException e) {
            System.out.println(e.getMessage());
            return;
        } catch (FileNotFoundException e) {
            System.out.println("input.txt не найден");
            return;
        } catch (InputMismatchException e) {
            System.out.println("input.txt имеет неверный формат");
            return;
        }

        reader = new Scanner(System.in).useLocale(Locale.US);

        while(true) {
            System.out.println("Выберите действие\n" +
                               "1. Изменить значение в точке\n" +
                               "2. Изменить значения на интервале\n" +
                               "3. Сумма значений на интервале\n" +
                               "4. Выход");

            int c;

            if (reader.hasNextInt()) {
                c = reader.nextInt();
            } else {
                System.out.println("Неизвестная команда");
                reader.next();
                System.out.println();
                            continue;
            }

            if (c == 1) {
                updValue();
            } else if (c == 2) {
                updValues();
            } else if (c == 3) {
                System.out.println(getSum());
            } else if (c == 4) {
                break;
            } else {
                System.out.println("Неизвестная команда");
            }
            System.out.println();
        }

        reader.close();
    }
```

```
private static void updValue() {
    int n = dn.getLen();
    long maxValue = dn.getMaxValue();
    int a;
    Number x;

    System.out.printf("Введите точку [0-%s]\n", n-1);
    a = inputInt(0, n-1);

    System.out.println("Введите новое значение");
    x = inputElement(maxValue);

    try {
        dn.updateValue(a, x);
        System.out.println("Значение обновлено");
    } catch (DecompositionException e) {
        System.out.println(e.getMessage());
    }
}


private static void updValues() {
    int n = dn.getLen();
    long maxValue = dn.getMaxValue();
    int a, b;
    Number x;

    System.out.printf("Введите начальную точку [0-%s]\n", n-1);
    a = inputInt(0, n-1);

    System.out.printf("Введите конечную точку [%s-%s]\n", a, n-1);
    b = inputInt(a, n-1);

    System.out.println("Введите новое значение");
    x = inputElement(maxValue);

    try {
        dn.updateValues(a, b, x);
        System.out.println("Значения обновлены");
    } catch (DecompositionException e) {
        System.out.println(e.getMessage());
    }

}


private static double getSum() {
    int n = dn.getLen();
    int a, b;
    double s = 0.0;

    System.out.printf("Введите начальную точку [0-%s]\n", n-1);
    a = inputInt(0, n-1);

    System.out.printf("Введите конечную точку [%s-%s]\n", a, n-1);
    b = inputInt(a, n-1);

    try {
        s = dn.getSum(a, b);
    } catch (DecompositionException e) {
        System.out.println(e.getMessage());
    }
    return s;
}


private static int inputInt(int minVal, int maxVal) {
```

```
        int a;
        while (true) {
            if (reader.hasNextInt()) {
                a = reader.nextInt();
            } else {
                System.out.println("Введите целое положительное число");
                reader.next();
                continue;
            }

            if (a < minVal || a > maxVal) {
                System.out.printf("Введите значение из интервала [%s-%s]\n", minVal, maxVal);
                continue;
            }

            return a;
        }
    }

    private static Number inputElement(long maxValue) {
        Number x;
        while (true) {
            if (reader.hasNextLong() || reader.hasNextDouble()) {
                if (reader.hasNextLong()) {
                    long t = reader.nextLong();

                    if (t > maxValue) {
                        System.out.printf("Максимальное значение элемента - %s\n", maxValue);
                        continue;
                    }

                    if (t < -maxValue) {
                        System.out.printf("Минимальное значение элемента - %s\n", -maxValue);
                        continue;
                    }

                    x = t;
                } else {
                    double t = reader.nextDouble();

                    if (t > maxValue) {
                        System.out.printf("Максимальное значение элемента - %s\n", maxValue);
                        continue;
                    }

                    if (t < -maxValue) {
                        System.out.printf("Минимальное значение элемента - %s\n", -maxValue);
                        continue;
                    }

                    x = t;
                }
            } else {
                System.out.println("Введите числовое значение");
                reader.next();
                continue;
            }

            return x;
        }
    }
}
```

Decomposition.java

```
package rpslab1;
import java.io.File;
import java.io.FileNotFoundException;
```

```
import java.util.Scanner;
import java.util.InputMismatchException;
import java.util.ArrayList;
import java.util.Locale;

public class Decomposition {

    private ArrayList<Number> data;
    private int n;
    private final int MAX_DATA_SIZE = 1000;
    private final long MAX_VALUE = 10_000_000_000L;
    private ArrayList<Number> blocks;
    private int rt;

    public Decomposition(String filename) throws FileNotFoundException,
DecompositionException, InputMismatchException {
        readData(filename);

        calcBlocks();
    }

    private void readData(String filename) throws FileNotFoundException,
DecompositionException, InputMismatchException {
        Scanner in = new Scanner(new File(filename)).useLocale(Locale.US);

        n = in.nextInt();

        if (n > MAX_DATA_SIZE) {
            throw new DecompositionException(String.format("Максимальное количество элементов
- %s", MAX_DATA_SIZE));
        }

        data = new ArrayList<Number>(n);

        int i  = 0;
        while ( (in.hasNextLong() || in.hasNextDouble()) && i < n) {
            if (in.hasNextLong()) {
                long t = in.nextLong();

                if (t > MAX_VALUE) {
                    throw new DecompositionException(String.format("Максимальное значение
элемента - %s", MAX_VALUE));
                }

                if (t < -MAX_VALUE) {
                    throw new DecompositionException(String.format("Минимальное значение
элемента - %s", -MAX_VALUE));
                }

                data.add(t);
            } else {
                double t = in.nextDouble();

                if (t > MAX_VALUE) {
                    throw new DecompositionException(String.format("Максимальное значение
элемента - %s", MAX_VALUE));
                }

                if (t < -MAX_VALUE) {
                    throw new DecompositionException(String.format("Минимальное значение
элемента - %s", -MAX_VALUE));
                }

                data.add(t);
            }

            i++;
        }
```

```
    if (i == 0) {
        throw new InputMismatchException();
    }

    data.trimToSize();
    n = data.size();
}


private void calcBlocks() {
    rt = (int) Math.ceil(Math.sqrt(n));
    blocks = new ArrayList<Number>(rt);

    for (int i = 0; i < rt - 1; ++i) {
        blocks.add(0);

        final int idx = i * rt;
        int j = 0;
        while (j < rt && idx + j < n){
            Number v = blocks.get(i);
            v = v.doubleValue() + data.get(idx + j).doubleValue();

            blocks.set(i, v);
            ++j;
        }
    }
}

public double getSum(int a, int b) throws DecompositionException {
    if (a < 0 || a > b || a >= n || b < 0 || b >= n) {
        throw new DecompositionException("Интервал некорректный");
    }

    double sum = 0;
    final int startBlock = a/rt;
    final int endBlock = b/rt;

    if (startBlock == endBlock) {
        for (int i = a; i <= b; ++i) {
            sum += data.get(i).doubleValue();
        }
    } else {
        for (int i = startBlock+1; i < endBlock; ++i) {
            sum += blocks.get(i).doubleValue();
        }

        final int aIdx = a % rt;
        for (int i = aIdx; i < rt; ++i) {
            sum += data.get(startBlock*rt + i).doubleValue();
        }

        final int bIdx = b % rt;
        for (int i = 0; i <= bIdx; ++i) {
            sum += data.get(endBlock * rt + i).doubleValue();
        }
    }

    return sum;
}

public void updateValue(int id, Number x) throws DecompositionException {
    if (id < 0 || id >= n) {
        throw new DecompositionException("Индекс некорректный");
    }

    int bid = id / rt;

    double v = data.get(id).doubleValue();
    double bv = blocks.get(bid).doubleValue();
```

```
        data.set(id, x);
        blocks.set(bid, x.doubleValue() - v + bv);
    }


    public void updateValues(int a, int b, Number x) throws DecompositionException {
        if (a < 0 || a > b || a >= n || b < 0 || b >= n) {
            throw new DecompositionException("Интервал некорректный");
        }

        for(int i = a; i <= b; ++i) {
            updateValue(i, x);
        }
    }

    public int getLen() {
        return n;
    }

    public long getMaxValue() {
        return MAX_VALUE;
    }
}
```

DecompositionException.java

```
package rpslab1;

public class DecompositionException extends Exception{

    public DecompositionException(String message){

        super(message);
    }
}
```