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 №2 дисциплины «Разработка программных систем»

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

Киров 2023

1. Задание

Подготовить комплект технической документации на разработанный ранее набор классов. Для выполнения лабораторной работы необходимо решить следующие задачи.

- Провести ряд преобразований программного кода, полученного в ходе выполнения предыдущей работы, с использованием встроенных средств рефакторинга.
- Сопроводить код комментариями с использованием Javadoc.
- Сгенерировать документацию к разработанным классам.

2. Листинг программы

Исходный код класса Decomposition с комментариями приведен в приложении А.

3. Документация

Пример документации к классу Decomposition приведен в приложении Б.

4. Вывод

В ходе выполнения лабораторной работы были изучены основы Javadoc – генератора документации в HTML-формате из комментариев исходного кода на Java. Была разработана документация для класса Decomposition.

Приложение А.

Листинг программы

```
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;

/**
 * Execution of interval operations by means of sqrt decomposition
 *
 * @author Ilya Kryuchkov
*/
public class Decomposition {

    /** Array of elements */
    private ArrayList<Number> data;

    /** Number of elements */
    private int n;

    /** Maximum number of elements */
    private final int MAX_DATA_SIZE = 1000;

    /** Maximum value of the element */
    private final long MAX_VALUE = 10_000_000_000L;

    /** Array of blocks */
    private ArrayList<Number> blocks;

    /** Root Value */
    private int rt;

    /**
    * Constructs new sqrt decomposition class
    * @param filename Name input file
    * @throws DecompositionException Decomposition error
    * @throws FileNotFoundException File not found
    * @throws InputMismatchException File has an incorrect format
    */
    public Decomposition(String filename) throws FileNotFoundException,
DecompositionException, InputMismatchException {
        readData(filename);

        calcBlocks();
    }

    /**
    * Reading data from a file
    * @param filename Name input file
    * @throws DecompositionException Decomposition error
    * @throws FileNotFoundException File not found
    * @throws InputMismatchException File has an incorrect format
    */
    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();
    }

    /**
    * Splitting into intervals and calculating the sum on each interval
    */
    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;
            }
        }
    }

    /**
```

```
* Calculating the sum at a given interval
* @param a Starting point of the interval
* @param b The end point of the interval
* @return The sum of the values in the interval from a to b
* @throws DecompositionException Decomposition error
*/
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;
}

/**
* Changing the value at a given point
* @param id Index of the item to change
* @param x New value
* @throws DecompositionException Decomposition error
*/
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);
}

/**
* Changing values at a given interval
* @param a Starting point of the interval
* @param b The end point of the interval
* @param x New value
* @throws DecompositionException Decomposition error
*/
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);
        }
    }

    /**
    * Getting the number of elements
    * @return Number of elements
    */
    public int getLen() {
        return n;
    }

    /**
    * Getting the maximum value of an element
    * @return Maximum value of the element
    */
    public long getMaxValue() {
        return MAX_VALUE;
    }
}
```

Приложение Б.

Пример документации

Package rpslab1

Class Decomposition

java.lang.Object
    rpslab1.Decomposition

```
public class Decomposition
extends Object
```

Execution of interval operations by means of sqrt decomposition

Constructor Summary

Constructors

Constructor	Description
`Decomposition(String filename)`	Constructs new sqrt decomposition class

Method Summary

All Methods **Instance Methods** **Concrete Methods**

Modifier and Type	Method	Description
int	`getLen()`	Getting the number of elements
long	`getMaxValue()`	Getting the maximum value of an element
double	`getSum(int a, int b)`	Calculating the sum at a given interval
void	`updateValue(int id, Number x)`	Changing the value at a given point
void	`updateValues(int a, int b, Number x)`	Changing values at a given interval

Constructor Details

Decomposition

```
public Decomposition(String filename)
              throws FileNotFoundException,
                     rpslab1.DecompositionException,
                     InputMismatchException
```

Constructs new sqrt decomposition class

Parameters:

`filename` - Name input file

Throws:

`rpslab1.DecompositionException` - Decomposition error

`FileNotFoundException` - File not found

`InputMismatchException` - File has an incorrect format

Method Details

getSum

```
public double getSum(int a,
                     int b)
              throws rpslab1.DecompositionException
```

Calculating the sum at a given interval

Parameters:

`a` - Starting point of the interval

`b` - The end point of the interval

Returns:

The sum of the values in the interval from a to b

Throws:

`rpslab1.DecompositionException` - Decomposition error