МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Вятский государственный университет»

(ФГБОУ ВО «ВятГУ»)

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Разработка программных систем

Знакомство с языком программирования Java

Вариант 4

Выполнил студент группы ИВТ-31 _____/Кудяшев Я.Ю./

Проверил преподаватель_____/Чистяков Г.А./

Киров 2022

1. Задание

Разработать класс BigFraction для работы с дробной длинной арифметикой. Класс должен содержать следующие публичные методы: сложение, вычитание, умножение, деление, сокращение дроби.

Сигнатура методов должна иметь вид «public BigFraction operation(BigFtaction arg)». Представление дроби должно инкапсулироваться посредством двух экземпляров классов BigInteger. Класс должен иметь не менее двух конструкторов. Для корректного представления экземпляров класса при их выводе на экран требуется переопределить метод toString().

2. Листинг программы

Листинг программы приведен в приложении А.

3. Вывод

В ходе выполнения лабораторной работы были изучены основные конструкции языка программирования Java, структура программы, стандартные средства ввода/вывода; изучен основной функционал интегрированной среды разработки IntelliJ IDEA; написана программа для работы в дробной длинной арифметикой.

Приложение А

(обязательное)

Листинг программы

Main.java

```
package com.company;

import java.math.BigInteger;
import java.util.Scanner;

/**
 * Main class
 * @author Yaroslav Kudyashev
 * @version 1.0
 */
public class Main {

    /**
     *  Flag to exit the menu
     */
    public static boolean menu_element = false;

    /**
     * Flags to indicate the input of the numerator and denominator
     */
    public static boolean numerator_flag = false, denominator_flag = false;

    /**
     * Flag for dividing by 0
     */
    public static boolean flag_division = false;

    /**
     * Class variable for the first argument
     */
    public static BigFraction arg1 = new BigFraction(BigInteger.ONE, BigInteger.ONE);
```

```
/**
 * Class variable for the second argument
 */
public static BigFraction arg2 = new BigFraction(BigInteger.ONE, BigInteger.ONE);



/**
 * Menu implementation for working with fractions
 * @param args Common parameters
 */
public static void main(String[] args) {
    while (menu_element != true) {
        System.out.println("Use 'help' for reference.");
        System.out.println("Enter the command:");
        Scanner in = new Scanner(System.in);
        String element = in.nextLine();

        switch (element) {

            /**
             * List of menu functions
             */
            case "Help":
                System.out.println("1. Addition");
                System.out.println("2. Subtraction");
                System.out.println("3. Multiplication");
                System.out.println("4. Division");
                System.out.println("5. Exit");
                break;

            /**
             * @see BigFraction#Addition(BigFraction)
             */
            case "Addition":
                menu_minimization();
                BigFraction.answer = arg1.Addition(arg1);
                System.out.println(BigFraction.answer);
```

```
    break;

/**
 * @see BigFraction#Subtraction(BigFraction)
 */
case "Subtraction":
    menu_minimization();
    BigFraction.answer = arg1.Subtraction(arg1);
    System.out.println(BigFraction.answer);
    break;

/**
 * @see BigFraction#Multiplication(BigFraction)
 */
case "Multiplication":
    menu_minimization();
    BigFraction.answer = arg1.Multiplication(arg1);
    System.out.println(BigFraction.answer);
    break;

/**
 * @see BigFraction#Division(BigFraction)
 */
case "Division":
    flag_division = true;
    menu_minimization();
    BigFraction.answer = arg1.Division(arg1);
    System.out.println(BigFraction.answer);
    flag_division = false;
    break;

/**
 * Exit from the program
 */
case "Exit":
    menu_element = true;
    break;
```

```
        }

    }
}

/**
 * Input function for code minimization
 */
public static void menu_minimization() {
    numerator_flag = false;
    denominator_flag = false;
    input_numerator(1);
    numerator_flag = false;
    denominator_flag = false;
    input_numerator(2);
}

/**
 * Function for numerator and denominator input
 * @param number Operand number
 * @throws IndexOutOfBoundsException If one of the arguments went beyond 1000000
 * @throws ArithmeticException If any of the denominators is 0 or number is less than 0
 * @throws Exception If input number is not of the int type and all other situations
 */
public static void input_numerator(int number) {
    try {
        while (numerator_flag == false) {
            System.out.print("Enter the " + number + " numerator: ");
            Scanner first = new Scanner(System.in);

            if (number == 1) {
                arg1.numerator = first.nextBigInteger();
            } else arg2.numerator = first.nextBigInteger();


            int check1 = arg1.numerator.compareTo(BigInteger.valueOf(1000000));
            int check2 = arg2.numerator.compareTo(BigInteger.valueOf(1000000));
```

```
    int check3 = arg1.numerator.compareTo(BigInteger.valueOf(-1000000));
    int check4 = arg2.numerator.compareTo(BigInteger.valueOf(-1000000));
    int check5 = arg1.numerator.compareTo(BigInteger.valueOf(0));
    int check6 = arg2.numerator.compareTo(BigInteger.valueOf(0));

    if (check1 == 1 || check2 == 1) {
       throw
          new IndexOutOfBoundsException("Numerator is too big");
    }
    else if (check3 == (-1) || check4 == (-1)) {
       throw
          new IndexOutOfBoundsException("Numerator is too small");
    }
    else if (check5 <= 0 || check6 <= 0) throw
          new ArithmeticException("Entering negative numbers is prohibited");
    else
       numerator_flag = true;
 }
 while (denominator_flag == false) {
    System.out.print("Enter the " + number + " denominator: ");
    Scanner second = new Scanner(System.in);

    if (number == 1) {
       arg1.denominator = second.nextBigInteger();
    } else arg2.denominator = second.nextBigInteger();


    int check1 = arg1.denominator.compareTo(BigInteger.valueOf(1000000));
    int check2 = arg2.denominator.compareTo(BigInteger.valueOf(1000000));
    int check3 = arg1.denominator.compareTo(BigInteger.valueOf(-1000000));
    int check4 = arg2.denominator.compareTo(BigInteger.valueOf(-1000000));
    int check5 = arg1.denominator.compareTo(BigInteger.valueOf(0));
    int check6 = arg2.denominator.compareTo(BigInteger.valueOf(0));

    if (check1 == 1 || check2 > 1) {
       throw
            new IndexOutOfBoundsException("Denominator is too big");
```

```
            }
            else if (check3 == (-1) || check4 == (-1)){
                throw
                    new IndexOutOfBoundsException("Denominator is too small");
            }
            else if (check5 == 0 || check6 == 0) throw
                    new ArithmeticException("Denominator cannot be 0");
            else if (number == 2 && flag_division == true && check5 == 0) throw
                    new ArithmeticException("Denominator cannot be 0");
            else if (check5 <= 0 || check6 <= 0) throw
                    new ArithmeticException("Entering negative numbers is prohibited");
            else
                denominator_flag = true;
        }
    } catch (IndexOutOfBoundsException exp) {
        System.out.println(exp);
        input_numerator(number);
    } catch (ArithmeticException exp) {
        System.out.println(exp);
        input_numerator(number);
    } catch (Exception exp) {
        System.out.println("This is not a number or value is too big");
        input_numerator(number);
    }
  }

}
```

BigFraction.java

```
package com.company;


import java.math.BigInteger;


/**
 * Class for working with fractional arithmetic
 * @author Yaroslav Kudyashev
```

```
 * @version 1.0
 */
public class BigFraction {

    /**
     * Variables for input and output values
     */
    BigInteger numerator = BigInteger.valueOf(788);
    BigInteger denominator = BigInteger.valueOf(788);

    public static BigFraction answer = new BigFraction(BigInteger.ONE, BigInteger.ONE);

    // public BigFraction arg1 =

    /**
     * Constructor for input values
     * @param numerator Numerator of the first fraction
     * @param denominator Denominator of the first fraction
     */
    public BigFraction(BigInteger numerator, BigInteger denominator) {
        this.numerator = numerator;
        this.denominator = denominator;
    }

    /**
     * Method toString for correct representation of class instances
     * @return String with input values and result
     */
    public String toString() {
```

```
        return "The first operator is " + Main.arg1.numerator + "/" + Main.arg1.denominator +
                "\nThe second operator is " + Main.arg2.numerator + "/" + Main.arg2.denominator +
                "\nThe answer is " + answer.numerator + "/" + answer.denominator;
    }

    /**
     * Function for adding two fractions
     * @param arg1 Numerator and denominator of two fractions
     * @return Result of adding two fractions (numerator and denominator) + initial values of the arg parameter
     */
    public BigFraction Addition(BigFraction arg1) {
        answer.numerator =
(arg1.numerator.multiply(Main.arg2.denominator)).add(Main.arg2.numerator.multiply(arg1.denominator));
        answer.denominator = arg1.denominator.multiply(Main.arg2.denominator);
        BigFraction last_answer = new BigFraction(answer.numerator, answer.denominator);
        last_answer = Reduction(last_answer);
        return last_answer;
    }

    /**
     * Function for subtracting two fractions
     * @param arg1 Numerator and denominator of two fractions
     * @return Result of subtracting two fractions (numerator and denominator) + initial values of the arg parameter
     */
    public BigFraction Subtraction(BigFraction arg1) {
        answer.numerator =
(arg1.numerator.multiply(Main.arg2.denominator)).subtract(Main.arg2.numerator.multiply(arg1.denominator));
        answer.denominator = arg1.denominator.multiply(Main.arg2.denominator);
        BigFraction last_answer = new BigFraction(answer.numerator, answer.denominator);
        last_answer = Reduction(last_answer);
```

```
        return last_answer;
    }



    /**
     * Function for multiplication two fractions
     * @param arg Numerator and denominator of two fractions
     * @return Result of multiplication two fractions (numerator and denominator) + initial values of the arg parameter
     */
    public BigFraction Multiplication(BigFraction arg) {
        answer.numerator = arg.numerator.multiply(Main.arg2.numerator);
        answer.denominator = arg.denominator.multiply(Main.arg2.denominator);
        BigFraction last_answer = new BigFraction(answer.numerator, answer.numerator);
        last_answer = Reduction(last_answer);
        return last_answer;
    }


    /**
     * Function for division two fractions
     * @param arg Numerator and denominator of two fractions
     * @return Result of division two fractions (numerator and denominator) + initial values of the arg parameter
     */
    public BigFraction Division(BigFraction arg) {
        answer.numerator = arg.numerator.multiply(Main.arg2.denominator);
        answer.denominator = arg.denominator.multiply(Main.arg2.numerator);
        BigFraction last_answer = new BigFraction(answer.numerator, answer.denominator);
        last_answer = Reduction(last_answer);
        return last_answer;
```

```
}

/**
 * Finding the maximum common divider
 * @param numerator Numerator of the number
 * @param denominator Denominator of the number
 * @return Maximum common divider
 */
private BigInteger Checker(BigInteger numerator, BigInteger denominator) {

    /**
     * Variable for compare
     */
    int compare = denominator.compareTo(numerator);

    while (compare !=0) {
    compare =  denominator.compareTo(numerator);
        if (compare==1) denominator = denominator.subtract(numerator);
        else numerator = numerator.subtract(denominator);
    }
    return denominator;
}

/**
 * Function for reducing fractions
 * @param arg Numerator and denominator of two fractions
 * @return Reduced numerators and denominators of operands and result
 */
public BigFraction Reduction(BigFraction arg) {
```

```
        /**
         * Variable for additional calculations
         */
        BigInteger dop;

       /*  dop = Checker(numerator_1, denominator_1);
        numerator_1 = numerator_1 / dop;
        denominator_1 = denominator_1 / dop;

        dop = Checker(numerator_2, denominator_2);
        numerator_2 = numerator_2 / dop;
        denominator_2 = denominator_2 / dop;*/

        dop = Checker(answer.numerator, answer.denominator);
        answer.numerator = answer.numerator.divide(dop);
        answer.denominator = answer.denominator.divide(dop);

        return answer;
    }
}
```