­­МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Вятский государственный университет»**

**(ФГБОУ ВО «ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №2 дисциплины

«Разработка программных систем»

Вариант 4

Выполнил студент группы ИВТб-31\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Жеребцов К. А./

Проверил преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Чистяков Г.А. /

Киров 2023

1. Постановка задачи

• Провести ряд преобразований исходного кода, полученного в ходе выполнения предыдущей работы, с использованием встроенных средств рефакторинга.

• Сопроводить код комментариями с использованием JavaDoc.

• Сгенерировать документацию к разработанным классам.

2. Ход работы

При выполнении лабораторной работы были выполнены следующие действия:

• Заменено название некоторых переменных и методов для улучшения читаемости кода.

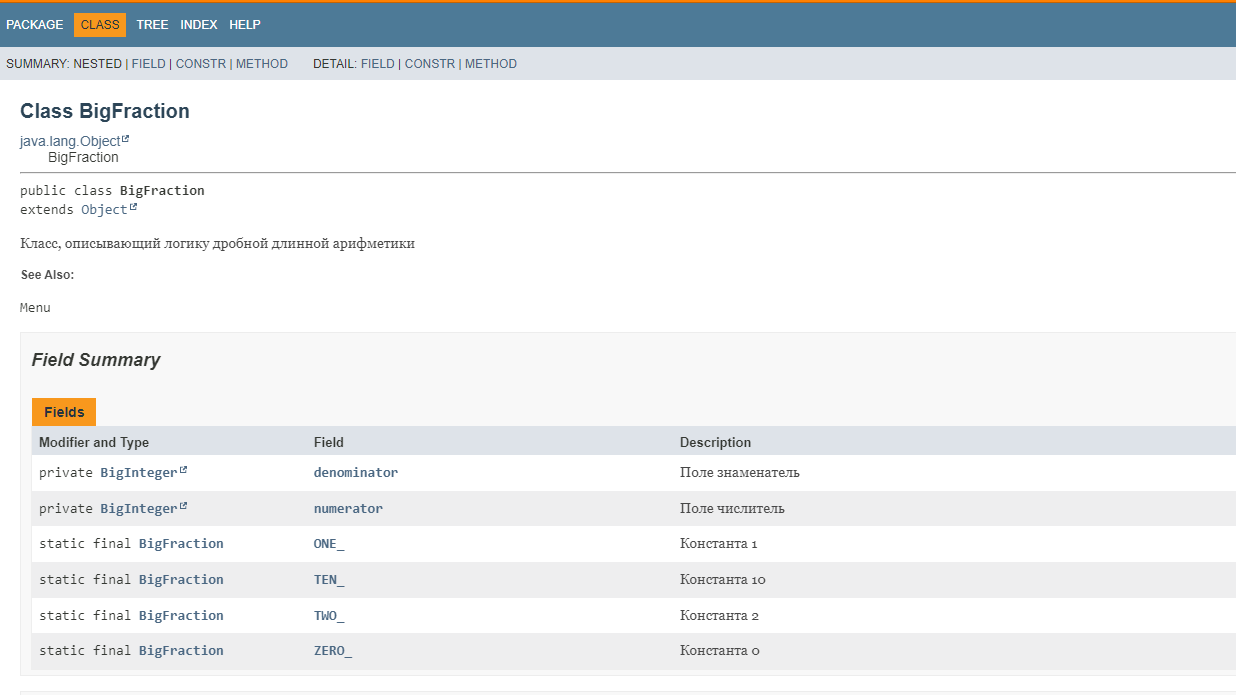
• Код отформатирован встроенными в среду средствами.

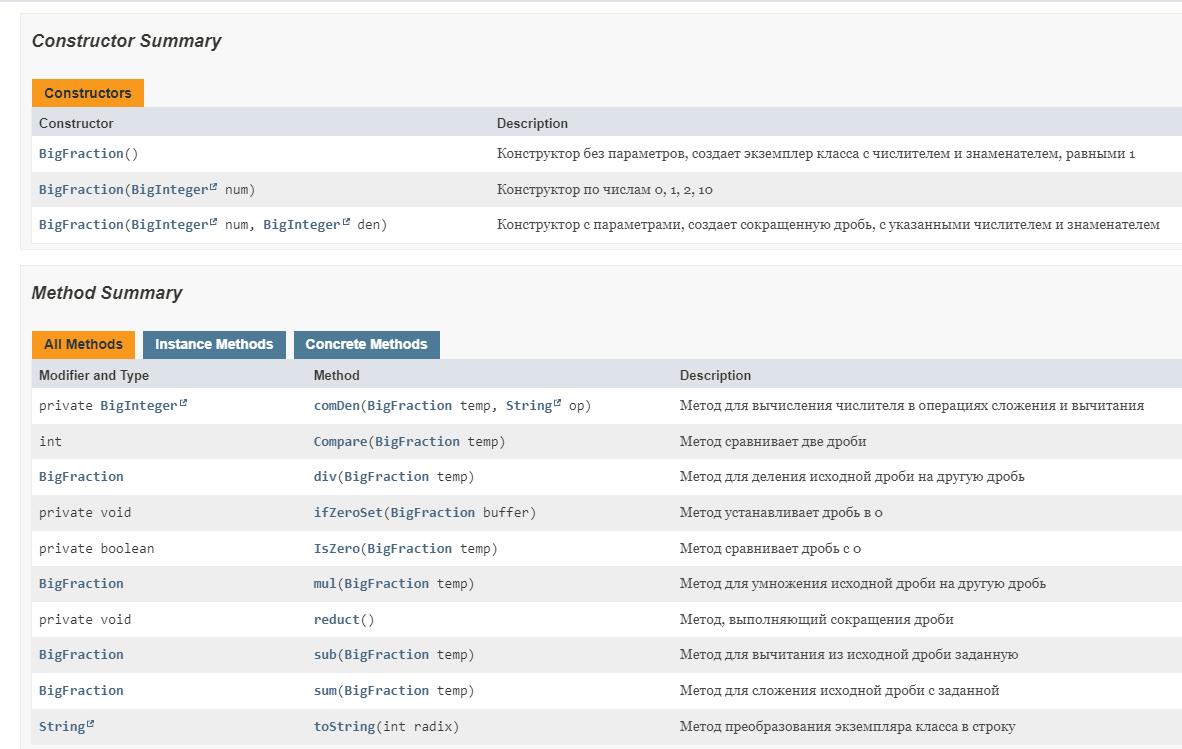
• Выполнены комментарии для генерации JavaDoc.

3. Исходный код

import java.math.BigInteger;  
import java.util.Objects;  
  
*/\*\*  
 \* Класс, описывающий логику дробной длинной арифметики  
 \** ***@author*** *Kirill  
 \** ***@version*** *1.0  
 \** ***@see*** *Menu  
 \*/*public class BigFraction {  
 */\*\* Поле числитель\*/* private BigInteger numerator;  
 */\*\* Поле знаменатель\*/* private BigInteger denominator;  
  
 */\*\* Константа 0\*/* public final static BigFraction *ZERO\_* = new BigFraction(BigInteger.*ZERO*);  
 */\*\* Константа 1\*/* public final static BigFraction *ONE\_* = new BigFraction(BigInteger.*ONE*);  
 */\*\* Константа 2\*/* public final static BigFraction *TWO\_* = new BigFraction(BigInteger.*TWO*);  
 */\*\* Константа 10\*/* public final static BigFraction *TEN\_* = new BigFraction(BigInteger.*TEN*);  
  
 */\*\*  
 \* Конструктор без параметров, создает экземплер класса с числителем и знаменателем, равными 1  
 \*/* public BigFraction() {  
 this.numerator = BigInteger.*ONE*;  
 this.denominator = BigInteger.*ONE*;  
 }  
   
 */\*\*  
 \* Конструктор с параметрами, создает сокращенную дробь, с указанными числителем и знаменателем  
 \** ***@param*** *num числитель  
 \** ***@param*** *den знаменатель  
 \*/* public BigFraction(BigInteger num, BigInteger den) {  
 if (den.compareTo(BigInteger.*ZERO*) == 0) {  
 System.*out*.println("Error! Denominator equals 0");  
 this.numerator = BigInteger.*ZERO*;  
 this.denominator = BigInteger.*ONE*;  
 }  
 if (den.compareTo(BigInteger.*ZERO*) > 0) {  
 if (num.compareTo(BigInteger.*ZERO*) != 0) {  
 this.numerator = num;  
 this.denominator = den;  
 }  
 if (num.compareTo(BigInteger.*ZERO*) == 0) {  
 this.numerator = BigInteger.*ZERO*;  
 this.denominator = BigInteger.*ONE*;  
 }  
 }  
 if (den.compareTo(BigInteger.*ZERO*) < 0) {  
 if (num.compareTo(BigInteger.*ZERO*) > 0) {  
 this.numerator = num.negate();  
 this.denominator = den.negate();  
 }  
 if (num.compareTo(BigInteger.*ZERO*) == 0) {  
 this.numerator = BigInteger.*ZERO*;  
 this.denominator = BigInteger.*ONE*;  
 }  
 if (num.compareTo(BigInteger.*ZERO*) < 0) {  
 this.numerator = num.negate();  
 this.denominator = den.negate();  
 }  
 }  
 this.reduct();  
 }  
   
 */\*\*  
 \* Конструктор по числам 0, 1, 2, 10  
 \** ***@param*** *num число, эквевалентно которому создается дробь  
 \*/* public BigFraction(BigInteger num) {  
 if (Objects.*equals*(num, BigInteger.*ZERO*)) {  
 this.numerator = BigInteger.*ZERO*;  
 this.denominator = BigInteger.*ONE*;  
 }  
 if (Objects.*equals*(num, BigInteger.*ONE*)) {  
 this.numerator = BigInteger.*ONE*;  
 this.denominator = BigInteger.*ONE*;  
 }  
 if (Objects.*equals*(num, BigInteger.*TWO*)) {  
 this.numerator = BigInteger.*TWO*;  
 this.denominator = BigInteger.*ONE*;  
 }  
 if (Objects.*equals*(num, BigInteger.*TEN*)) {  
 this.numerator = BigInteger.*TEN*;  
 this.denominator = BigInteger.*ONE*;  
 }  
 }  
  
 */\*\*  
 \* Метод сравнивает две дроби  
 \** ***@param*** *temp дробь, с которой сравнивается исходная дробь  
 \** ***@return*** *результат сравнения: меньше (-1), равно (0), больше (1)  
 \*/* public int Compare(BigFraction temp) {  
 this.numerator.multiply(temp.denominator);  
 temp.numerator.multiply(this.denominator);  
 return this.numerator.compareTo(temp.numerator);  
 }  
   
 */\*\*  
 \* Метод сравнивает дробь с 0  
 \** ***@param*** *temp Дробь, которую нужно проверить  
 \** ***@return*** *результа сравнения: true - равно 0, false - не равно 0  
 \*/* private boolean IsZero(BigFraction temp) {  
 return temp.Compare(*ZERO\_*) == 0;  
 }  
  
 */\*\*  
 \* Метод устанавливает дробь в 0  
 \** ***@param*** *buffer Дробь, которую нужно установить в 0  
 \*/* private void ifZeroSet(BigFraction buffer) {  
 if (buffer.numerator.compareTo(BigInteger.*ZERO*) == 0) {  
 buffer.numerator = BigInteger.*ZERO*;  
 buffer.denominator = BigInteger.*ONE*;  
 }  
 }  
  
 */\*\*  
 \* Метод для умножения исходной дроби на другую дробь  
 \** ***@param*** *temp Дробь, на которую нужно умножить  
 \** ***@return*** *сокращенный результат  
 \*/* public BigFraction mul(BigFraction temp) {  
 BigFraction buffer = new BigFraction();  
 buffer.denominator = this.denominator.multiply(temp.denominator);  
 buffer.numerator = this.numerator.multiply(temp.numerator);  
 ifZeroSet(buffer);  
 buffer.reduct();  
 return buffer;  
 }  
  
 */\*\*  
 \* Метод для деления исходной дроби на другую дробь  
 \** ***@param*** *temp Дробь, на которую нужно поделить  
 \** ***@return*** *сокращенный результат  
 \*/* public BigFraction div(BigFraction temp) {  
 BigFraction buffer = new BigFraction();  
 if (!IsZero(temp)) {  
 buffer.numerator = this.numerator.multiply(temp.denominator);  
 buffer.denominator = this.denominator.multiply(temp.numerator);  
 ifZeroSet(buffer);  
 } else {  
 System.*out*.println("Error! Division by 0");  
 buffer = null;  
 }  
 if (buffer != null) {  
 buffer.reduct();  
 }  
 return buffer;  
 }  
  
 */\*\*  
 \* Метод для вычисления числителя в операциях сложения и вычитания  
 \** ***@param*** *temp Дробь, которая суммируется или вычитается  
 \** ***@param*** *op Операция: + или -  
 \** ***@return*** *числитель после выполнения операции (числитель результата)  
 \*/* private BigInteger comDen(BigFraction temp, String op) {  
 BigInteger res = null;  
 BigFraction temp1 = new BigFraction();  
 BigFraction temp2 = new BigFraction(temp.numerator, temp.denominator);  
 temp1.numerator = this.numerator.multiply(temp.denominator);  
 temp2.numerator = temp.numerator.multiply(this.denominator);  
 if (op.equals("+")) {  
 res = temp1.numerator.add(temp2.numerator);  
 }  
 if (op.equals("-")) {  
 res = temp1.numerator.subtract(temp2.numerator);  
 }  
 return res;  
 }  
  
 */\*\*  
 \* Метод для сложения исходной дроби с заданной  
 \** ***@param*** *temp Дробь, которую нужно прибавить к исходной  
 \** ***@return*** *сокращенный результат операции  
 \*/* public BigFraction sum(BigFraction temp) {  
 BigFraction buffer = new BigFraction();  
 if (!Objects.*equals*(this.denominator, temp.denominator)) {  
 buffer.numerator = comDen(temp, "+");  
 buffer.denominator = this.denominator.multiply(temp.denominator);  
 }  
 if (Objects.*equals*(this.denominator, temp.denominator)) {  
 buffer.numerator = this.numerator.add(temp.numerator);  
 buffer.denominator = this.denominator;  
 }  
 ifZeroSet(buffer);  
 buffer.reduct();  
 return buffer;  
 }  
   
 */\*\*  
 \* Метод для вычитания из исходной дроби заданную  
 \** ***@param*** *temp Дробь, которую нужно вычесть из исходной  
 \** ***@return*** *сокращенный результат операции  
 \*/* public BigFraction sub(BigFraction temp) {  
 BigFraction buffer = new BigFraction();  
 if (!Objects.*equals*(this.denominator, temp.denominator)) {  
 buffer.numerator = comDen(temp, "-");  
 buffer.denominator = this.denominator.multiply(temp.denominator);  
 }  
 if (Objects.*equals*(this.denominator, temp.denominator)) {  
 buffer.numerator = this.numerator.subtract(temp.numerator);  
 buffer.denominator = this.denominator;  
 }  
 ifZeroSet(buffer);  
 buffer.reduct();  
 return buffer;  
 }  
  
 */\*\*  
 \* Метод, выполняющий сокращения дроби  
 \*/* private void reduct() {  
 //BigFraction temp = new BigFraction(this.numerator.abs(), this.denominator.abs());  
 BigFraction temp = new BigFraction();  
 temp.numerator = this.numerator.abs();  
 temp.denominator = this.denominator.abs();  
 BigInteger NOD = temp.numerator.gcd(temp.denominator);  
 temp.numerator = this.numerator.divide(NOD);  
 temp.denominator = this.denominator.divide(NOD);  
 this.numerator = temp.numerator;  
 this.denominator = temp.denominator;  
 }  
  
 */\*\*  
 \* Метод преобразования экземпляра класса в строку  
 \** ***@param*** *radix Система счисления, в которой необходимо представить результат  
 \** ***@return*** *строковое представление экземпляра класса в заданной системе счисления  
 \*/* public String toString(int radix) {  
 String buffer = "";  
 buffer = this.numerator.toString(radix) + "/" + this.denominator.toString(radix);  
 return buffer;  
 }  
}

4. Экранные формы





Вывод

В результате выполнения лабораторной работы были изучены инструменты рефакторинга среды разработки IntelliJ IDEA. При помощи данных инструментов был переработан код программы для лучшей читаемости. Также были написаны комментарии к коду и сгенерирована документация JavaDoc.