

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЁВА»
КАФЕДРА «ТЕХНИЧЕСКАЯ КИБЕРНЕТИКА»

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ
«Объектно-ориентированное программирование»
ЛАБОРАТОРНАЯ РАБОТА №6

Студент _____ Чячяков А.И.

Группа _____ 6301-030301D

Руководитель _____ Борисов Д. С.

Оценка

Задание 1

Добавил в класс Functions метод, возвращающий значение интеграла функции, вычисленное с помощью численного метода.

В качестве параметров метод получает ссылку типа Function на объект функции, значения левой и правой границы области интегрирования, а также шаг дискретизации.

Если интервал интегрирования выходит за границы области определения функции, метод выбрасывает исключение.

Вычисление значения интеграла выполняется по методу трапеций. Для этого вся область интегрирования разбивается на участки, длина которых (кроме одного) равна шагу дискретизации. На каждом таком участке площадь под кривой, описывающейся заданной функцией, приближается площадью трапеции, две вершины которой расположены на оси абсцисс на границах участка, а ещё две – на кривой в точках границ участка.

В методе main() проверил работу метода интегрирования. Для этого вычислил интеграл для экспоненты на отрезке от 0 до 1. Определил также, какой шаг дискретизации нужен, чтобы рассчитанное значение отличалось от теоретического в 7 знаке после запятой.

Код 1.1

```
public static double integrate(Function f, double leftX, double rightX, double step) 4 usages
    throws FunctionPointIndexOutOfBoundsException {

    if (leftX < f.getLeftDomainBorder() || rightX > f.getRightDomainBorder()) {
        throw new FunctionPointIndexOutOfBoundsException(
            "Интервал интегрирования выходит за границы области определения функции"
        );
    }

    double integral = 0.0;
    double x1 = leftX;

    while (x1 < rightX) {
        double x2 = Math.min(x1 + step, rightX);
        double y1 = f.getFunctionValue(x1);
        double y2 = f.getFunctionValue(x2);
        double trapezoidArea = (y1 + y2) / 2.0 * (x2 - x1);
        integral += trapezoidArea;
        x1 = x2;
    }

    return integral;
}
```

Код 1.2

```

public class Main {
    public static void main(String[] args) throws Exception {
        Function exp = new Exp();
        double theoreticalValue = Math.exp(1) - 1;

        System.out.println("Теоретическое значение интеграла e^x на [0, 1]: " + theoreticalValue);
        System.out.println();

        for (double step = 0.1; step >= 0.0001; step /= 10) {
            try {
                double result = Functions.integrate(exp, leftX: 0, rightX: 1, step);
                double error = Math.abs(result - theoreticalValue);
                System.out.printf("Шаг: %.4f, Результат: %.10f, Ошибка: %.2e%n",
                    step, result, error);
            } catch (FunctionPointIndexOutOfBoundsException e) {
                System.out.println("Ошибка: " + e.getMessage());
            }
        }
    }
}

```

Выход

```

Теоретическое значение интеграла e^x на [0, 1]: 1.718281828459045

Шаг: 0,1000, Результат: 1,7197134914, Ошибка: 1,43e-03
Шаг: 0,0100, Результат: 1,7182961475, Ошибка: 1,43e-05
Шаг: 0,0010, Результат: 1,7182819716, Ошибка: 1,43e-07
Шаг: 0,0001, Результат: 1,7182818299, Ошибка: 1,43e-09

Process finished with exit code 0

```

Задание 2

Далее в приложении один поток инструкций будет генерировать задачи для интегрирования, а второй поток – решать их. Для этого потребуется объект задания, через который эти потоки будут взаимодействовать. В объекте будут храниться параметры задания.

Создал пакет threads, в котором будут размещены классы, связанные с потоками.

В пакете threads описал класс Task, объект которого должен хранить ссылку на объект интегрируемой функции, границы области интегрирования, шаг дискретизации, а также целочисленное поле, хранящее количество выполняемых заданий.

В главном классе программы описал метод nonThread(), реализующий последовательную (без применения потоков инструкций) версию программы.

В методе создал объект класса Task и установить в нём количество выполняемых заданий (минимум 100). После этого в цикле (до количества заданий) нужно выполнить следующие действия:

1. создал и поместил в объект задания объект логарифмической функции, основание которой является случайной величиной, распределённой равномерно на отрезке от 1 до 10;
2. указал в объекте задания левую границу области интегрирования (случайно распределена на отрезке от 0 до 100);
3. указал в объекте задания правую границу области интегрирования (случайно распределена на отрезке от 100 до 200);
4. указал в объекте задания шаг дискретизации (случайно распределён на отрезке от 0 до 1);
5. вывел в консоль сообщение вида "Source <левая граница> <правая граница> <шаг дискретизации>";
6. вычислил значение интеграла для параметров из объекта задания;
7. вывел в консоль сообщение вида "Result <левая граница> <правая граница> <шаг дискретизации> <результат интегрирования>".

Проверил работу метода, вызвав его в методе main().

Код 2.1

```
>Main.java  Task.java  ArrayTabulatedFunction.java  Functions.java
1 package threads;
2
3 import functions.Function;
4
5 public class Task { 3 usages
6     private Function function; 2 usages
7     private double leftX; 2 usages
8     private double rightX; 2 usages
9     private double step; 2 usages
10    private int tasksCount; 2 usages
11
12    public Function getFunction() {
13        return function;
14    }
15
16    public void setFunction(Function function) {
17        this.function = function;
18    }
19
20    public double getLeftX() { 2 usages
21        return leftX;
22    }
23
24    public void setLeftX(double leftX) { 1 usage
25        this.leftX = leftX;
26    }
27
28    public double getRightX() { 2 usages
29        return rightX;
30    }
31
32    public void setRightX(double rightX) { 1 usage
33        this.rightX = rightX;
34    }
35
36    public double getStep() { 2 usages
37        return step;
38    }
39
40    public void setStep(double step) { 1 usage
41        this.step = step;
42    }
```

Код 2.2

```

private static void nonThread() { 1 usage
    Task task = new Task();
    task.setTasksCount(100);

    for (int i = 0; i < task.getTasksCount(); i++) {
        double base = 1 + Math.random() * 9;
        double leftX = Math.random() * 100;
        double rightX = 100 + Math.random() * 100;
        double step = Math.random();

        task.setFunction(new Log(base));
        task.setLeftX(leftX);
        task.setRightX(rightX);
        task.setStep(step);

        System.out.printf("Source %.2f %.2f %.6f%.6f%n",
                           leftX, rightX, step);

        double result = Functions.integrate(
            task.getFunction(),
            task.getLeftX(),
            task.getRightX(),
            task.getStep()
        );

        System.out.printf("Result %.2f %.2f %.6f %.6f%.6f%n",
                           task.getLeftX(), task.getRightX(), task.getStep(), result);
    }
}

```

Выход:

Source 31,79 155,54 0,175262
 Result 31,79 155,54 0,175262 1376,308535
 Source 24,88 178,44 0,474933
 Result 24,88 178,44 0,474933 439,160597
 Source 40,15 151,39 0,457532
 Result 40,15 151,39 0,457532 227,248747
 Source 99,06 127,53 0,658298
 Result 99,06 127,53 0,658298 64,824276
 Source 45,82 174,35 0,025032
 Result 45,82 174,35 0,025032 1862,362203
 Source 19,26 167,06 0,487418
 Result 19,26 167,06 0,487418 349,343854
 Source 60,48 139,68 0,825369

Result 60,48 139,68 0,825369 178,095382
Source 20,11 157,50 0,520015
Result 20,11 157,50 0,520015 382,137892
Source 1,42 163,63 0,389116
Result 1,42 163,63 0,389116 452,438786
Source 51,60 169,55 0,458125
Result 51,60 169,55 0,458125 279,465370
Source 92,47 105,21 0,698736
Result 92,47 105,21 0,698736 46,151781
Source 64,01 117,47 0,978725
Result 64,01 117,47 0,978725 145,663593
Source 39,34 107,62 0,810013
Result 39,34 107,62 0,810013 207,992390
Source 58,14 128,22 0,256314
Result 58,14 128,22 0,256314 168,823726
Source 17,20 117,78 0,667392
Result 17,20 117,78 0,667392 210,804644
Source 50,99 137,31 0,743863
Result 50,99 137,31 0,743863 214,381935
Source 33,43 187,54 0,804635
Result 33,43 187,54 0,804635 323,632691
Source 32,48 127,41 0,028525
Result 32,48 127,41 0,028525 209,953014
Source 53,39 185,01 0,872192
Result 53,39 185,01 0,872192 359,762171
Source 13,55 150,84 0,828526
Result 13,55 150,84 0,828526 325,522723
Source 2,14 169,59 0,182706
Result 2,14 169,59 0,182706 342,548209
Source 91,00 101,78 0,447632
Result 91,00 101,78 0,447632 21,692813
Source 79,99 109,20 0,716048

Result 79,99 109,20 0,716048 71,951348
Source 79,39 129,37 0,253607
Result 79,39 129,37 0,253607 287,527609
Source 54,76 143,23 0,275297
Result 54,76 143,23 0,275297 179,943033
Source 85,66 107,63 0,997057
Result 85,66 107,63 0,997057 58,576306
Source 18,60 119,66 0,735059
Result 18,60 119,66 0,735059 321,401893
Source 26,47 103,21 0,037083
Result 26,47 103,21 0,037083 191,783634
Source 22,40 160,00 0,573540
Result 22,40 160,00 0,573540 647,822952
Source 14,00 107,05 0,790131
Result 14,00 107,05 0,790131 259,444124
Source 25,53 130,39 0,129786
Result 25,53 130,39 0,129786 399,983728
Source 60,42 120,94 0,968459
Result 60,42 120,94 0,968459 118,718470
Source 23,83 122,61 0,277477
Result 23,83 122,61 0,277477 195,648837
Source 76,40 114,95 0,424620
Result 76,40 114,95 0,424620 129,632020
Source 43,81 173,18 0,456697
Result 43,81 173,18 0,456697 346,123819
Source 64,22 196,76 0,668369
Result 64,22 196,76 0,668369 830,976047
Source 15,93 174,50 0,255081
Result 15,93 174,50 0,255081 354,230064
Source 30,65 159,02 0,372404
Result 30,65 159,02 0,372404 279,167262
Source 64,76 156,07 0,979750

Result 64,76 156,07 0,979750 337,119472
Source 70,14 195,53 0,433336
Result 70,14 195,53 0,433336 464,696957
Source 79,07 139,75 0,911425
Result 79,07 139,75 0,911425 205,420906
Source 93,72 102,54 0,404311
Result 93,72 102,54 0,404311 18,547839
Source 49,52 197,22 0,182150
Result 49,52 197,22 0,182150 322,197028
Source 3,88 105,86 0,365678
Result 3,88 105,86 0,365678 195,937012
Source 87,15 163,39 0,468008
Result 87,15 163,39 0,468008 302,362647
Source 40,90 180,14 0,550110
Result 40,90 180,14 0,550110 689,228271
Source 60,19 184,17 0,938251
Result 60,19 184,17 0,938251 621,650868
Source 51,07 151,00 0,288587
Result 51,07 151,00 0,288587 302,398292
Source 12,44 102,10 0,152242
Result 12,44 102,10 0,152242 188,532432
Source 52,07 196,35 0,093655
Result 52,07 196,35 0,093655 448,339959
Source 50,83 108,48 0,845619
Result 50,83 108,48 0,845619 122,208250
Source 96,00 187,63 0,946348
Result 96,00 187,63 0,946348 305,479276
Source 15,19 115,30 0,797089
Result 15,19 115,30 0,797089 203,948192
Source 55,37 137,16 0,103196
Result 55,37 137,16 0,103196 198,672272
Source 68,18 125,77 0,217249

Result 68,18 125,77 0,217249 593,575320
Source 82,56 142,59 0,989722
Result 82,56 142,59 0,989722 140,243143
Source 68,00 106,64 0,350857
Result 68,00 106,64 0,350857 193,158409
Source 18,00 144,39 0,996830
Result 18,00 144,39 0,996830 266,459788
Source 12,18 191,15 0,233940
Result 12,18 191,15 0,233940 518,975671
Source 47,55 137,47 0,246488
Result 47,55 137,47 0,246488 201,588097
Source 64,92 165,86 0,797157
Result 64,92 165,86 0,797157 222,585584
Source 34,44 102,96 0,643125
Result 34,44 102,96 0,643125 224,294131
Source 6,22 120,05 0,756825
Result 6,22 120,05 0,756825 788,217642
Source 76,18 151,05 0,777134
Result 76,18 151,05 0,777134 336,530421
Source 65,40 161,08 0,985004
Result 65,40 161,08 0,985004 390,033229
Source 61,80 191,08 0,496891
Result 61,80 191,08 0,496891 331,497267
Source 27,18 124,39 0,300058
Result 27,18 124,39 0,300058 351,043969
Source 29,84 169,70 0,215708
Result 29,84 169,70 0,215708 287,682270
Source 22,93 131,42 0,869089
Result 22,93 131,42 0,869089 203,968666
Source 52,48 143,31 0,270106
Result 52,48 143,31 0,270106 500,482527
Source 45,23 174,22 0,663659

Result 45,23 174,22 0,663659 271,701564
Source 96,17 175,63 0,895829
Result 96,17 175,63 0,895829 191,548252
Source 48,46 130,06 0,660197
Result 48,46 130,06 0,660197 244,244566
Source 34,17 176,53 0,638003
Result 34,17 176,53 0,638003 473,688894
Source 23,39 134,70 0,726508
Result 23,39 134,70 0,726508 259,375690
Source 11,39 123,05 0,261843
Result 11,39 123,05 0,261843 283,231126
Source 58,63 167,71 0,385782
Result 58,63 167,71 0,385782 289,609339
Source 86,96 128,75 0,480313
Result 86,96 128,75 0,480313 111,142695
Source 26,85 128,47 0,899508
Result 26,85 128,47 0,899508 291,193287
Source 4,84 152,20 0,065151
Result 4,84 152,20 0,065151 266,007764
Source 58,68 106,35 0,360707
Result 58,68 106,35 0,360707 188,123240
Source 67,89 160,89 0,157660
Result 67,89 160,89 0,157660 242,854528
Source 92,08 173,44 0,738889
Result 92,08 173,44 0,738889 251,797137
Source 15,88 191,29 0,845678
Result 15,88 191,29 0,845678 779,262678
Source 10,23 153,56 0,068128
Result 10,23 153,56 0,068128 636,715474
Source 32,63 171,36 0,323119
Result 32,63 171,36 0,323119 561,025312
Source 5,68 145,93 0,219681

```
Result 5,68 145,93 0,219681 285,480166
Source 32,95 196,30 0,198510
Result 32,95 196,30 0,198510 464,706190
Source 33,75 107,03 0,950541
Result 33,75 107,03 0,950541 164,642998
Source 4,48 173,64 0,768919
Result 4,48 173,64 0,768919 427,489083
Source 43,03 119,23 0,169338
Result 43,03 119,23 0,169338 171,379773
Source 5,02 178,74 0,325678
Result 5,02 178,74 0,325678 494,692250
Source 95,83 199,60 0,610989
Result 95,83 199,60 0,610989 249,307907
Source 40,66 186,62 0,808317
Result 40,66 186,62 0,808317 355,593770
Source 16,62 127,55 0,361516
Result 16,62 127,55 0,361516 243,781484
Source 75,16 171,21 0,373107
Result 75,16 171,21 0,373107 226,111517
Source 31,16 194,66 0,924415
Result 31,16 194,66 0,924415 347,631499
Source 45,83 187,95 0,692618
Result 45,83 187,95 0,692618 964,546623
Source 84,31 105,37 0,770011
Result 84,31 105,37 0,770011 61,165721
Source 86,51 180,89 0,052712
Result 86,51 180,89 0,052712 636,430808
```

Задание 3

В пакете threads создал два следующих класса. При их реализации воспользовался фрагментами последовательной версии программы из предыдущего задания.

Класс SimpleGenerator реализовывает интерфейс Runnable, получает в конструкторе и сохранять в своё поле ссылку на объект типа Task, а в методе run() в цикле должны формироваться задачи и заносятся в полученный объект задания, а также выводиться сообщения в консоль.

Класс SimpleIntegrator реализовывает интерфейс Runnable, получает в конструкторе и сохраняет в своё поле ссылку на объект типа Task, а в методе run() в цикле должны решаться задачи, данные для которых берутся из полученного объекта задания, а также выводиться сообщения в консоль.

В главном классе программы создал метод simpleThreads(). В нём создайте объект задания, укажите количество выполняемых заданий (минимум 100), создайте и запустите два потока вычислений, основанных на описанных классах SimpleGenerator и SimpleIntegrator.

Код 3.1

```
>Main.java  SimpleGenerator.java  SimpleIntegrator.java  Task.java  ArrayTabulatedFun
1 package threads;
2
3 import functions.basic.Log;
4 import functions.Functions;
5
6 public class SimpleGenerator implements Runnable { no usages
7     private final Task task; 7 usages
8
9     public SimpleGenerator(Task task) { no usages
10         this.task = task;
11     }
12
13     @Override
14     public void run() {
15         for (int i = 0; i < task.getTasksCount(); i++) {
16
17             double base = 1 + Math.random() * 9;
18             double leftX = Math.random() * 100;
19             double rightX = 100 + Math.random() * 100;
20             double step = Math.random();
21
22             synchronized (task) {
23                 task.setFunction(new Log(base));
24                 task.setLeftX(leftX);
25                 task.setRightX(rightX);
26                 task.setStep(step);
27
28                 System.out.printf("Source %.2f %.2f %.6f%n", leftX, rightX, step);
29             }
30
31             try {
32                 Thread.sleep( millis: 5);
33             } catch (InterruptedException e) {
34                 Thread.currentThread().interrupt();
35             }
36         }
37     }
38 }
```

Код 3.2

The screenshot shows a Java code editor with several tabs at the top: Main.java, SimpleGenerator.java, SimpleIntegrator.java (which is currently selected), Task.java, and ArrayTabulatedFunc. The code in SimpleIntegrator.java is as follows:

```
1 package threads;
2
3 import functions.Functions;
4
5 public class SimpleIntegrator implements Runnable { 2 usages
6     private final Task task; 8 usages
7
8     public SimpleIntegrator(Task task) { 1 usage
9         this.task = task;
10    }
11
12    @Override
13    public void run() {
14
15        for (int i = 0; i < task.getTasksCount(); i++) {
16
17            double leftX, rightX, step, result;
18
19            synchronized (task) {
20                if (task.getFunction() == null) {
21                    continue;
22                }
23
24                leftX = task.getLeftX();
25                rightX = task.getRightX();
26                step = task.getStep();
27                result = Functions.integrate(task.getFunction(), leftX, rightX, step);
28            }
29
30            System.out.printf("Result %.2f %.2f %.6f %.6f%n",
31                leftX, rightX, step, result);
32
33            try {
34                Thread.sleep( millis: 5);
35            } catch (InterruptedException e) {
36                Thread.currentThread().interrupt();
37            }
38        }
39    }
40}
```

Код 3.3

```
private static void simpleThreads() { 1 usage
    Task task = new Task();
    task.setTasksCount(100);

    Thread generator = new Thread(new SimpleGenerator(task), "Generator");
    Thread integrator = new Thread(new SimpleIntegrator(task), "Integrator");

    generator.start();
    integrator.start();

    try {
        generator.join();
        integrator.join();
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
    }

    System.out.println("Оба потока завершили работу");
}
```

Выход:

Задание 3 многопоточная версия

```
Source 92,19 191,68 0,715287
Result 92,19 191,68 0,715287 213,933651
Result 92,19 191,68 0,715287 213,933651
Source 34,03 112,61 0,741975
Result 34,03 112,61 0,741975 170,795408
Source 26,00 161,27 0,356091
Result 26,00 161,27 0,356091 3837,253211
Source 20,89 185,16 0,828992
Source 90,27 162,63 0,159238
Result 20,89 185,16 0,828992 373,548790
Result 90,27 162,63 0,159238 430,603597
Source 40,51 143,03 0,873269
Result 40,51 143,03 0,873269 206,273621
Source 45,95 194,78 0,186991
Result 45,95 194,78 0,186991 360,474609
Source 65,44 140,80 0,727004
```

Result 65,44 140,80 0,727004 151,692267
Source 66,65 173,55 0,737792
Result 66,65 173,55 0,737792 269,172742
Source 94,61 132,31 0,459958
Source 85,95 194,69 0,154626
Result 85,95 194,69 0,154626 274,512810
Result 85,95 194,69 0,154626 274,512810
Source 48,18 180,69 0,070441
Result 48,18 180,69 0,070441 270,588606
Source 94,67 193,40 0,502103
Result 94,67 193,40 0,502103 221,906432
Source 50,26 122,96 0,973209
Result 50,26 122,96 0,973209 150,208133
Source 19,92 122,54 0,491154
Result 19,92 122,54 0,491154 910,274255
Source 43,40 123,67 0,768192
Result 43,40 123,67 0,768192 413,537342
Source 42,45 196,76 0,280257
Result 42,45 196,76 0,280257 576,983779
Source 4,19 177,55 0,043943
Result 4,19 177,55 0,043943 674,554063
Source 33,95 174,77 0,032095
Result 33,95 174,77 0,032095 736,012034
Source 54,01 145,78 0,219727
Result 54,01 145,78 0,219727 273,608525
Source 21,03 177,17 0,354549
Source 84,04 135,83 0,680151
Result 21,03 177,17 0,354549 24788,162280
Result 84,04 135,83 0,680151 173,618586
Source 7,41 161,86 0,004942
Result 7,41 161,86 0,004942 348,298515
Source 38,84 159,45 0,706239

Result 38,84 159,45 0,706239 262,769368
Source 11,02 104,56 0,486422
Result 11,02 104,56 0,486422 200,350736
Source 27,42 128,62 0,956866
Result 27,42 128,62 0,956866 192,637412
Source 97,66 156,55 0,415417
Result 97,66 156,55 0,415417 218,746932
Source 56,06 174,40 0,504445
Result 56,06 174,40 0,504445 266,800224
Source 17,80 176,86 0,789046
Result 17,80 176,86 0,789046 1122,594567
Source 2,06 198,16 0,678789
Result 2,06 198,16 0,678789 445,506016
Source 78,24 161,33 0,531393
Result 78,24 161,33 0,531393 543,781978
Source 72,48 190,16 0,223041
Result 72,48 190,16 0,223041 1333,425320
Source 5,19 115,70 0,947572
Result 5,19 115,70 0,947572 197,895163
Source 16,97 175,22 0,515579
Result 16,97 175,22 0,515579 320,150543
Source 80,52 144,94 0,964462
Source 89,02 170,16 0,861816
Result 89,02 170,16 0,861816 190,626561
Result 89,02 170,16 0,861816 190,626561
Source 57,67 156,30 0,287031
Result 57,67 156,30 0,287031 201,139224
Source 74,53 162,84 0,109786
Result 74,53 162,84 0,109786 193,164369
Source 89,00 198,07 0,849167
Result 89,00 198,07 0,849167 306,618467
Source 32,07 147,57 0,093422

Result 32,07 147,57 0,093422 471,379332
Source 21,12 181,50 0,187890
Result 21,12 181,50 0,187890 857,170450
Source 13,72 176,65 0,657453
Result 13,72 176,65 0,657453 1561,217407
Source 39,96 158,18 0,619890
Result 39,96 158,18 0,619890 449,300739
Source 60,93 112,73 0,267795
Source 72,04 149,88 0,349267
Result 72,04 149,88 0,349267 193,306545
Result 72,04 149,88 0,349267 193,306545
Source 94,25 191,91 0,461370
Source 73,77 153,50 0,885393
Result 73,77 153,50 0,885393 665,951599
Result 73,77 153,50 0,885393 665,951599
Source 41,26 132,18 0,639240
Source 65,31 168,05 0,399803
Result 65,31 168,05 0,399803 406,306093
Result 65,31 168,05 0,399803 406,306093
Source 71,79 147,10 0,956978
Source 90,63 116,14 0,901014
Result 90,63 116,14 0,901014 81,438508
Source 34,52 149,41 0,316166
Result 34,52 149,41 0,316166 308,804243
Result 34,52 149,41 0,316166 308,804243
Source 10,48 157,13 0,356186
Result 10,48 157,13 0,356186 337,158108
Source 14,19 154,28 0,710941
Result 14,19 154,28 0,710941 270,583344
Source 59,84 114,32 0,138525
Result 59,84 114,32 0,138525 127,182296
Source 53,46 117,19 0,433483

Result 53,46 117,19 0,433483 209,234882
Source 26,54 163,28 0,429337
Result 26,54 163,28 0,429337 458,004665
Source 8,68 165,91 0,892363
Result 8,68 165,91 0,892363 336,802402
Source 32,64 153,78 0,453641 Result 32,64 153,78 0,453641 331,874504
Source 33,58 111,58 0,047648
Result 33,58 111,58 0,047648 209,083790
Source 25,24 162,42 0,678246
Result 25,24 162,42 0,678246 358,958461
Source 10,14 148,51 0,716671
Result 10,14 148,51 0,716671 1919,442691
Source 8,44 198,32 0,351191
Result 8,44 198,32 0,351191 631,376265
Source 70,66 177,56 0,254376
Result 70,66 177,56 0,254376 228,821785
Source 72,89 109,11 0,011470
Result 72,89 109,11 0,011470 112,361383
Source 80,34 112,79 0,362143
Result 80,34 112,79 0,362143 73,565670
Source 4,75 159,47 0,847934
Result 4,75 159,47 0,847934 612,723073
Source 74,27 156,26 0,028996
Result 74,27 156,26 0,028996 189,029614
Source 80,70 179,53 0,036282
Result 80,70 179,53 0,036282 225,957630
Source 68,25 150,55 0,859105
Result 68,25 150,55 0,859105 451,996215
Source 80,45 105,31 0,433047
Result 80,45 105,31 0,433047 79,003296
Source 87,30 143,97 0,289478
Result 87,30 143,97 0,289478 201,933678

Source 73,65 158,60 0,255481
Result 73,65 158,60 0,255481 276,859937
Source 81,29 128,12 0,451773
Result 81,29 128,12 0,451773 103,749873
Source 62,65 138,40 0,626254
Result 62,65 138,40 0,626254 211,319918
Source 21,51 132,58 0,682101
Result 21,51 132,58 0,682101 210,212501
Source 87,82 121,30 0,111856
Result 87,82 121,30 0,111856 67,874640
Source 97,67 186,50 0,932761
Result 97,67 186,50 0,932761 221,642052
Source 27,11 192,95 0,633625
Result 27,11 192,95 0,633625 649,605088
Source 59,10 181,85 0,589788
Result 59,10 181,85 0,589788 325,133295
Source 46,33 158,04 0,380304
Source 69,69 198,14 0,450237
Result 46,33 158,04 0,380304 382,893287
Result 69,69 198,14 0,450237 378,101235
Source 82,09 100,95 0,601019
Source 70,30 173,25 0,863093
Result 70,30 173,25 0,863093 565,312070
Result 70,30 173,25 0,863093 565,312070
Source 63,67 194,27 0,407609
Result 63,67 194,27 0,407609 317,106236
Source 68,12 130,34 0,287737
Result 68,12 130,34 0,287737 215,784789
Source 5,79 177,66 0,965190
Result 5,79 177,66 0,965190 1175,783006
Source 96,93 195,39 0,773843
Result 96,93 195,39 0,773843 304,663019

Source 28,66 154,12 0,699875
Result 28,66 154,12 0,699875 279,118446
Source 17,48 165,29 0,116643
Source 87,21 176,05 0,402550
Result 87,21 176,05 0,402550 333,025025
Result 87,21 176,05 0,402550 333,025025
Source 52,06 111,16 0,691441
Result 52,06 111,16 0,691441 191,642690
Source 15,85 118,30 0,357613
Result 15,85 118,30 0,357613 268,095716
Source 72,50 126,74 0,221305
Result 72,50 126,74 0,221305 113,740806
Source 20,66 160,50 0,875584
Source 85,62 156,50 0,981820
Result 85,62 156,50 0,981820 187,368976
Result 85,62 156,50 0,981820 187,368976
Source 75,49 126,02 0,858488
Result 75,49 126,02 0,858488 226,882161
Source 85,43 163,38 0,196829
Result 85,43 163,38 0,196829 163,896950
Source 3,67 149,60 0,276925
Result 3,67 149,60 0,276925 293,308571
Source 66,03 120,79 0,629056
Оба потока завершили работу

Задание 4

Определил причину того, что не все сгенерированные задания оказываются выполнены интегрирующим потоком.

Для устранения этой проблемы потребуется дополнительный объект, представляющий собой одноместный семафор, различающий операции чтения и записи в защищаемый объект (пример класса таких объектов был приведён в лекции).

В пакете threads создал два следующих класса. При их реализации воспользовался фрагментами классов из предыдущего задания.

Класс Generator должен расширять класс Thread, получать в конструкторе и сохранять в свои поля ссылки на объект типа Task и на объект семафора, а в методе run() должны выполняться те же действия, что и в предыдущей версии генерирующего класса.

Класс Integrator должен расширять класс Thread, получать в конструкторе и сохранять в свои поля ссылки на объект типа Task и на объект семафора, а в методе run() должны выполняться те же действия, что и в предыдущей версии интегрирующего класса.

Отличие этих классов от предыдущих версий должно заключаться в том, что вместо средств синхронизации в методах run() должны использоваться возможности семафора.

В главном классе программы создал метод complicatedThreads(). В нём создайте объект задания, укажите количество выполняемых заданий (минимум 100), создайте и запустите два потока вычислений классов Generator и Integrator.

Попробовал запускать программу (несколько раз). Попробовал запускать программу, изменяя приоритеты потоков перед их запуском. Сделайте так, чтобы после создания ваших потоков основной поток программы выжидал 50 миллисекунд, после чего прерывал работу ваших потоков путём вызова метода interrupt(). Изменил код ваших классов потоков так, чтобы прерывание происходило и происходило корректно.

Код 4.1

```
>Main.java  Semaphore.java  Generator.java  Integrator.java  SimpleGenerator.java
1 package threads;
2
3 public class Semaphore { no usages
4     private boolean available = true; 3 usages
5
6     public synchronized void acquire() throws InterruptedException { no usages
7         while (!available) {
8             wait();
9         }
10        available = false;
11    }
12
13    public synchronized void release() { no usages
14        available = true;
15        notifyAll();
16    }
17 }
```

Код 4.2

```
>Main.java    Semaphore.java    Generator.java x    Integrator.java    SimpleGenerator.java
6  public class Generator extends Thread { 2 usages
7
8      public Generator(Task task, Semaphore semaphore) { 1 usage
9          this.task = task;
10         this.semaphore = semaphore;
11     }
12
13
14
15     @Override
16     public void run() {
17         try {
18             for (int i = 0; i < task.getTasksCount(); i++) {
19                 if (Thread.interrupted()) {
20                     System.out.println("Generator прерван");
21                     break;
22                 }
23
24
25                 double base = 1 + Math.random() * 9;
26                 double leftX = Math.random() * 100;
27                 double rightX = 100 + Math.random() * 100;
28                 double step = Math.random();
29
30
31                 semaphore.acquire();
32
33                 task.setFunction(new Log(base));
34                 task.setLeftX(leftX);
35                 task.setRightX(rightX);
36                 task.setStep(step);
37
38
39                 System.out.printf("Source %.2f %.2f %.6f%n", leftX, rightX, step);
40
41                 semaphore.release();
42
43
44                 try {
45                     Thread.sleep( millis: 5);
46                 } catch (InterruptedException e) {
47                     Thread.currentThread().interrupt();
48                 }
49             }
50         } catch (InterruptedException e) {
51             Thread.currentThread().interrupt();
52             System.out.println("Generator InterruptedException");
53         }
54     }
55 }
```

Код 4.3

```
@Override
public void run() {
    try {
        for (int i = 0; i < task.getTasksCount(); i++) {
            if (Thread.interrupted()) {
                System.out.println("Integrator прерван");
                break;
            }

            double leftX, rightX, step, result;

            semaphore.acquire();

            if (task.getFunction() == null) {
                semaphore.release();
                continue;
            }

            leftX = task.getLeftX();
            rightX = task.getRightX();
            step = task.getStep();
            result = Functions.integrate(task.getFunction(), leftX, rightX, step);

            task.setFunction(null);

            semaphore.release();

            System.out.printf("Result %.2f %.2f %.6f %.6f%n",
                leftX, rightX, step, result);

            try {
                Thread.sleep(5);
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            }
        }
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        System.out.println("Integrator InterruptedException");
    }
}
```

Код 4.4

```
private static void complicatedThreads() { 1 usage
    Task task = new Task();
    task.setTasksCount(100);

    Semaphore semaphore = new Semaphore();

    Generator generator = new Generator(task, semaphore);
    Integrator integrator = new Integrator(task, semaphore);

    generator.start();
    integrator.start();

    try {
        Thread.sleep( millis: 50 );
        generator.interrupt();
        integrator.interrupt();

        generator.join();
        integrator.join();
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
    }

    System.out.println("Оба потока завершили работу (прерваны)");
}
```

Выход:

```
Задание 4: с семафором
Source 80,66 137,09 0,227278
Result 80,66 137,09 0,227278 898,458056
Result 80,66 137,09 0,227278 898,458056
Source 20,87 169,39 0,330753
Result 20,87 169,39 0,330753 7295,483538
Source 79,10 128,53 0,160063
Result 79,10 128,53 0,160063 129,843629
Source 44,72 103,80 0,551638
Result 44,72 103,80 0,551638 114,687836
Source 23,22 148,62 0,792388
Result 23,22 148,62 0,792388 600,205205
Source 47,99 133,04 0,250273
Source 81,11 151,81 0,388224
Result 81,11 151,81 0,388224 177,967439
Result 81,11 151,81 0,388224 177,967439
Source 23,43 138,49 0,707319
Result 23,43 138,49 0,707319 301,335318
Source 61,26 134,75 0,127084
Generator прерван
Integrator прерван
Оба потока завершили работу (прерваны)

Process finished with exit code 0
```

Код 4.5

```
private static void complicatedThreads() { 1 usage
    Task task = new Task();
    task.setTasksCount(100);

    Semaphore semaphore = new Semaphore();

    Generator generator = new Generator(task, semaphore);
    Integrator integrator = new Integrator(task, semaphore);

    generator.setPriority(Thread.MAX_PRIORITY);
    integrator.setPriority(Thread.MIN_PRIORITY);
```

Вывод:

```
Задание 4: с семафором
Source 67,78 139,01 0,888254
Result 67,78 139,01 0,888254 191,732423
Result 67,78 139,01 0,888254 191,732423
Source 92,23 187,92 0,267712
Result 92,23 187,92 0,267712 506,360227
Source 4,90 116,98 0,531976
Result 4,90 116,98 0,531976 301,857374
Source 14,44 118,65 0,506307
Result 14,44 118,65 0,506307 6426,112994
Source 85,92 132,67 0,344654
Result 85,92 132,67 0,344654 17411,469936
Source 80,15 156,42 0,762962
Result 80,15 156,42 0,762962 171,753128
Source 65,88 172,94 0,537918
Result 65,88 172,94 0,537918 327,441700
Source 96,85 183,18 0,852704
Result 96,85 183,18 0,852704 217,159591
Source 28,75 166,62 0,048870
Result 28,75 166,62 0,048870 275,584595
Generator прерван
Integrator прерван
Оба потока завершили работу (прерваны)
```

Код 4.6

```
Semaphore semaphore = new Semaphore();
Generator generator = new Generator(task, semaphore);
Integrator integrator = new Integrator(task, semaphore);

generator.setPriority(Thread.MIN_PRIORITY);
integrator.setPriority(Thread.MAX_PRIORITY);
```

Вывод:

```
Задание 4: с семафором
Source 64,99 196,51 0,130768
Result 64,99 196,51 0,130768 818,186491
Source 29,07 194,63 0,022302
Result 29,07 194,63 0,022302 1488,033659
Source 10,35 114,73 0,125268
Result 10,35 114,73 0,125268 191,699392
Source 9,10 136,11 0,337141
Result 9,10 136,11 0,337141 947,477374
Source 8,41 167,43 0,550851
Result 8,41 167,43 0,550851 296,128395
Source 58,92 122,60 0,561400
Result 58,92 122,60 0,561400 157,714167
Source 17,32 197,03 0,045001
Result 17,32 197,03 0,045001 381,779002
Source 47,74 186,43 0,530690
Result 47,74 186,43 0,530690 424,474111
Source 74,14 135,19 0,465020
Result 74,14 135,19 0,465020 345,563868
Generator прерван
Integrator прерван
Оба потока завершили работу (прерваны)
```