

Самарский национальный исследовательский университет имени
академика С. П. Королёва

Лабораторная работа №6

Выполнил:
Студент группы 6301-030301D
Абдуллин А.Р.

Задания 1

```
//Машинный эпсилон
double DEFAULT_EPSILON = 1e-9;

//-----
Function exp = new Exp();
double exact = Math.E - 1;
double step = 0.001;
double value = Functions.integrate(exp, left: 0, right: 1, step);
System.out.println("Численно:" + value);
System.out.println("Аналитически:" + exact);
System.out.println("Ошибка:" + Math.abs(value - exact));
```

```
Численно:1.7182819716491948
Аналитически:1.718281828459045
Ошибка:1.4319014973729338E-7
```

Реализовал в классе Functions метод численного интегрирования (методом трапеций), выбрасывающий исключение в случае выхода за область определения функции. Проверил работу метода на интеграле \exp . Погрешность до 7 знака соответствует шагу дискретизации $\text{step} = 0.001$.

Задания 2

```
package threads;

import functions.Function;

public class Task { 14 usages
    private Function function; 2 usages
    private double left; 2 usages
    private double right; 2 usages
    private double step; 2 usages
    private int tasksCount; 2 usages

    public Function getFunction() { return function; }
    public void setFunction(Function function) { this.function = function; }
    public double getLeft() { return left; }
    public void setLeft(double left) { this.left = left; }
    public double getRight() { return right; }
    public void setRight(double right) { this.right = right; }
    public double getStep() { return step; }
    public void setStep(double step) { this.step = step; }
    public int getTasksCount() { return tasksCount; }
    public void setTasksCount(int tasksCount) { this.tasksCount = tasksCount; }
}
```

Создал класс Task, служащий хранилищем параметров задачи интегрирования. В main классе реализовал метод `nonThread()`, который последовательно выполняет задания без использования потоков.

Задание 3

```
public class SimpleGenerator implements Runnable { 1 usage
    private Task task; 7 usages
    private Random random = new Random(); 4 usages

    public SimpleGenerator(Task task) { 1 usage
        this.task = task;
    }
    public void run() {
        for (int i = 0; i < task.getTasksCount(); i++) {
            double base = 1 + 9 * random.nextDouble();
            Function function = new Log(base);
            double left = random.nextDouble() * 100;
            double right = 100 + random.nextDouble() * 100;
            double step = random.nextDouble();
            if (step == 0) step = 0.1;
            synchronized (task) {
                task.setFunction(function);
                task.setLeft(left);
                task.setRight(right);
                task.setStep(step);
            }
            System.out.printf("Source %.4f %.4f %.4f\n", left, right, step);
        }
    }
}
```

```
package threads;

import functions.Functions;

public class SimpleIntegrator implements Runnable { 1 usage
    private Task task; 7 usages
    public SimpleIntegrator(Task task) { 1 usage
        this.task = task;
    }
    public void run() {
        for (int i = 0; i < task.getTasksCount(); i++) {
            double left, right, step, result;
            synchronized (task) {
                left = task.getLeft();
                right = task.getRight();
                step = task.getStep();
                result = Functions.integrate(task.getFunction(), left, right, step);
            }
            System.out.printf("Result %.4f %.4f %.4f %.6f\n", left, right, step, result);
        }
    }
}
```

Создал классы SimpleGenerator и SimpleIntegrator, которые могут работать в разных потоках и взаимодействовать с общим объектом Task. Во время работы иногда программа завершалась с исключением NullPointerException, проблема была связана с тем, что SimpleIntegrator работал до инициализации всех полей Task. Так же в некоторых случаях интегрирующий поток выводит сообщение с набором данных, который не встречается в сообщениях генерирующего потока. Исправил эту проблему синхронизацией чтения и записи

Задание 4

Вывод метода `simpleThreads()` показывает, что не все сгенерированные задания оказываются выполнены интегрирующим потоком, так как генератор мог обновлять данные задачи, перед тем как интегратор их обработает.

```
/usr/java/jdk-25-oracle-x64/bin/java -javaagent:/snap/intellij-idea-community/685/lib/idea_rt.jar=43579 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8
Source 98,2685 183,2126 0,8428
Source 38,0128 140,2895 0,3695
Source 74,9763 142,0644 0,6828
Source 46,1832 198,7561 0,0195
Source 20,5671 142,1768 0,6623
Result 98,2685 183,2126 0,8428 222,568716
Result 72,3268 103,6795 0,4988 62,092783
Source 72,3268 103,6795 0,4988
Source 18,3291 137,9726 0,7615
Source 9,3950 143,3787 0,8568
Source 81,6106 150,0539 0,3497
Source 22,7333 134,9723 0,2406
Source 98,1113 175,1124 0,6416
Source 12,3135 170,3377 0,8421
Source 47,8651 145,9484 0,5382
Source 45,1118 105,7849 0,6924
Source 25,2845 108,0995 0,0986
Result 72,3268 103,6795 0,4988 62,092783
Source 62,7140 128,5664 0,1661
Source 39,8449 115,7835 0,0732
Source 94,2238 158,4282 0,3092
Result 62,7140 128,5664 0,1661 133,931158
Source 88,9875 160,8577 0,4073
Source 29,5539 167,4043 0,3544
Source 69,0173 184,6987 0,4288
Source 33,1765 124,0570 0,0355
Source 33,1760 142,3852 0,1840
Source 3,4130 189,3672 0,5663
Source 86,6678 114,5264 0,4089
```

```
Source 7,3392 120,4780 0,3951
Source 77,2145 189,1206 0,6505
Source 46,3862 115,4934 0,0999
Source 72,7201 156,3459 0,7991
Source 32,7129 137,5365 0,0367
Source 54,6220 189,7861 0,1082
Source 81,4303 193,2699 0,6862
Result 88,9875 160,8577 0,4073 205,115866
Result 56,5871 108,1161 0,1922 115,938717
Result 56,5871 108,1161 0,1922 115,938717
Source 56,5871 108,1161 0,1922
Source 48,7537 130,0810 0,2381
Result 56,5871 108,1161 0,1922 115,938717
Result 98,5707 138,5087 0,8987 567,425784
Source 98,5707 138,5087 0,8987
Source 71,1632 197,1358 0,4988
Source 49,4969 127,3861 0,7645
Source 85,6483 198,5051 0,8656
Source 56,3379 169,2202 0,5736
Source 36,1165 171,5637 0,0999
Result 98,5707 138,5087 0,8987 567,425784
Result 36,1165 171,5637 0,0999 347,987780
Source 9,9789 149,0835 0,3365
Source 96,0433 148,8690 0,2920
Source 38,7105 119,6548 0,4869
Source 62,4413 186,7196 0,9539
Source 47,6550 158,3817 0,2739
Source 1,2276 168,5979 0,7712
Result 9,9789 149,0835 0,3365 290,335503
Result 1,2276 168,5979 0,7712 370,917053
```

```
Source 46,3636 175,5849 0,5897
Result 46,3636 175,5849 0,5897 268,453657
Source 49,0667 114,7008 0,8781
Source 32,3846 168,6447 0,7283
Source 1,7675 169,7592 0,7688
Source 23,0711 144,6554 0,7578
Source 44,3121 110,3523 0,2075
Source 54,2423 154,4498 0,6621
Source 2,5953 149,9174 0,7514
Source 86,1703 111,4955 0,2107
Source 21,9521 104,6604 0,7499
Result 49,0667 114,7008 0,8781 134,667500
Source 86,8483 189,6894 0,3252
Source 2,5513 120,5676 0,9320
Source 83,5795 121,3892 0,5117
Source 46,7346 130,7252 0,9472
Source 1,0950 140,7499 0,4844
Source 98,3778 153,1464 0,4719
Source 28,0680 144,7435 0,6503
Source 79,3524 162,3114 0,7441
Source 81,4935 164,7693 0,6236
Source 53,1476 192,4021 0,6412
Source 58,3601 172,3550 0,7049
Source 70,8673 154,9761 0,6632
Source 95,3442 162,9729 0,1295
Source 79,2647 188,6614 0,4346
Result 86,8483 189,6894 0,3252 355,326333
Source 77,2119 189,2346 0,5991
Source 96,5879 139,7762 0,5884
Source 62,1400 139,0866 0,2162
```

```
Source 85,6187 194,3760 0,1457
Source 16,4410 179,7101 0,4100
Source 79,4002 179,5526 0,7539
Source 76,2728 112,8290 0,8823
Source 39,5918 182,6458 0,3971
Source 78,9097 112,1273 0,7995
Result 77,2119 189,2346 0,5991 367,165798
Result 52,5512 107,7935 0,4413 114,864580
Result 52,5512 107,7935 0,4413 114,864580
Result 52,5512 107,7935 0,4413 114,864580
Result 52,5512 107,7935 0,4413 114,864580
Result 52,5512 107,7935 0,4413 114,864580
Source 52,5512 107,7935 0,4413
Source 31,1532 138,4815 0,7200
Source 93,2191 130,0212 0,1286
Source 7,4613 153,7088 0,9959
Source 11,8864 130,0754 0,8830
Source 95,3731 178,5360 0,6363
Source 23,9223 118,2033 0,0415
Source 30,8089 165,4911 0,8714
Source 7,5394 180,0821 0,8732
Source 90,8214 182,4213 0,9058
Source 39,6764 103,1016 0,0007
Source 36,3783 127,3402 0,5980
Source 25,0750 166,2020 0,5856
Source 22,7532 120,3076 0,0563
Source 98,6293 101,4711 0,8932
Source 20,1033 143,0447 0,3869
Source 6,0861 160,7037 0,0980
Source 11,8120 121,9884 0,0857
Source 93,1746 145,1261 0,4442
```

[illegible][illegible]

[illegible]

```
Source 20,8588 107,0924 0,6382
Result 20,8588 107,0924 0,6382 196,342023
Source 2,7307 140,8413 0,0648
Result 2,7307 140,8413 0,0648 1294,134094
Source 62,1336 186,4827 0,8039
Result 62,1336 186,4827 0,8039 306,943652
Source 22,8378 186,2110 0,6834
Result 22,8378 186,2110 0,6834 373,449718
Source 98,7598 171,4460 0,8635
Result 98,7598 171,4460 0,8635 284,489714
Source 51,6206 140,1735 0,6527
Result 51,6206 140,1735 0,6527 557,015416

Process finished with exit code 0
```

Рис. 1: Вывод метода complicatedThreads()

Чтобы это решить был написан семафор, для операций чтения и записи. Так же написаны классы Generator и Integrator, использующие семафор.