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:1.718281971649194 8
Аналитически:1.71828182845904 5
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 соотвествует шагу дискретизации 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 параметром задачи интегрирования. В main классе реализовал метод nonThread(), который последовательно выполняет задания без использования потоков.

Задание 3

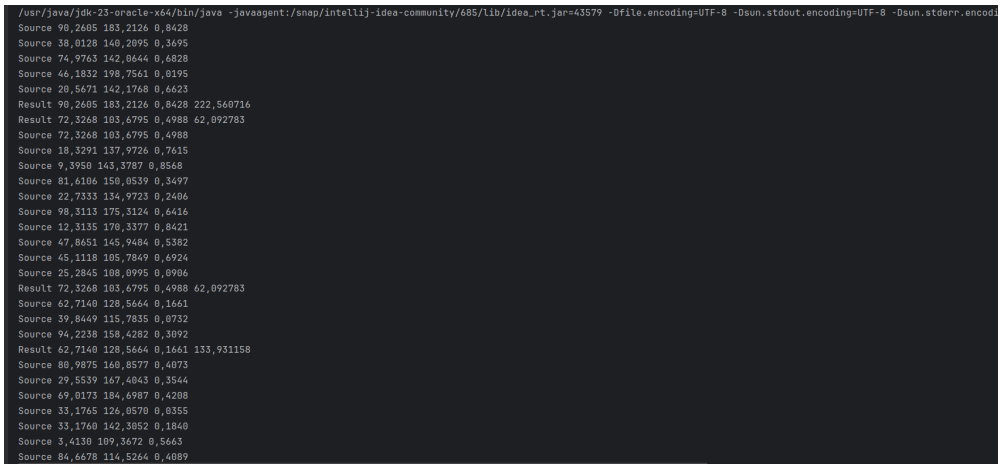
```java
public class SimpleGenerator implements Runnable {  1 usage
    private Task task;  7 usages
    private Random random = new Random();  4 usages

    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);
        }
    }
}
```

```java
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, resul
        }
    }
}
```

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 инициалазации 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.



```
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 114,864500
Result 52,5512 107,7935 0,4413 114,864500
Result 52,5512 107,7935 0,4413 114,864500
Result 52,5512 107,7935 0,4413 114,864500
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
```

```
Source 63,1744 175,4872 0,8086
Source 58,2424 181,4893 0,1246
Result 52,5512 107,7935 0,4413 114,864500
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
```

```
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
Result 58,2424 181,4893 0,1246 289,433775
```

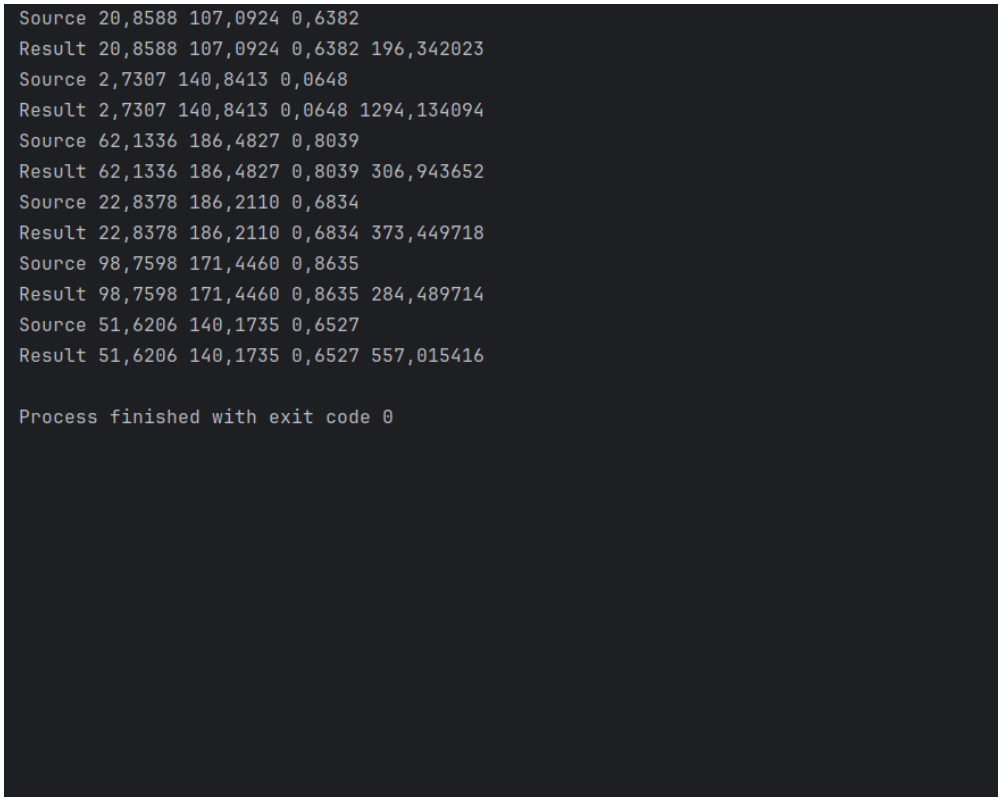


Рис. 1: Вывод метода complicatedThreads()

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