Самарский национальный исследовательский университет имени академика С. П. Королёва

Лабораторная работа №6

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

Задания 1

```java
//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

```java
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; }
}
```

```
Source 65,4653 173,9494 0,9892
Result 65,4653 173,9494 0,9892 292,545469
Source 6,8437 129,6373 0,6488
Result 6,8437 129,6373 0,6488 323,224120
Source 22,3493 160,3932 0,1339
Result 22,3493 160,3932 0,1339 267,996784
Source 53,9635 103,2059 0,1060
Result 53,9635 103,2059 0,1060 93,509416
Source 50,3207 167,8217 0,3133
Result 50,3207 167,8217 0,3133 289,639261
Source 26,9248 106,1686 0,1656
Result 26,9248 106,1686 0,1656 161,005841
Source 50,1116 167,4472 0,3600
Result 50,1116 167,4472 0,3600 1073,710228
Source 7,4698 162,5626 0,4271
Result 7,4698 162,5626 0,4271 1559,991179
Source 5,2613 183,9329 0,0515
Result 5,2613 183,9329 0,0515 415,804189
Source 76,8420 139,6969 0,1575
Result 76,8420 139,6969 0,1575 173,221436
Source 95,1633 164,9599 0,0454
Result 95,1633 164,9599 0,0454 371,761830
Source 57,1445 166,9526 0,3121
Result 57,1445 166,9526 0,3121 246,481485
Source 48,0352 145,8422 0,8989
Result 48,0352 145,8422 0,8989 248,042048
Численно:1.7182819716491948
Аналитически:1.718281828459045
Ошибка:1.4319014973729338E-7
```

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

```
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
```

Заметил, что не все сгенерированные задания оказываются выполнены интегрирующим потоком, так как генератор мог обновлять данные задачи, перед тем как интегратор их обработает. Чтобы это решить был написан семафор, для операций чтения и записи. Так же написаны классы Generator и Integrator, использующие семафор.