Отчёт по лабораторной работе №6

Студент: Прокопенко А. П.

Группа: 6201-120303D

Задание 1

Реализация численного интегрирования методом трапеций

Был добавлен метод в класс Functions для вычисления интеграла функции методом трапеций. Метод принимает параметры:

- ссылку на объект функции

- левую и правую границы интегрирования

- шаг дискретизации

Проверка точности: Для экспоненты на отрезке [0,1] был определён шаг дискретизации, обеспечивающий точность до 7 знака после запятой.

Код:

```
public static double integrate(Function function, double a, double b, double step) {  3 usages
    if (a > b) {
        throw new IllegalArgumentException("Левая граница не может быть больше правой");
    }

    // Проверяем границы области определения
    if (a < function.getLeftDomainBorder() || b > function.getRightDomainBorder()) {
        throw new IllegalArgumentException("Интервал интегрирования выходит за границы области определения функции");
    }

    if (step <= 0) {
        throw new IllegalArgumentException("Шаг дискретизации должен быть положительным");
    }

    double integral = 0.0;
    double x1 = a;
    double x2;

    // Проходим по всем сегментам
    while (x1 < b) {
        x2 = Math.min(x1 + step, b);
        double y1 = function.getFunctionValue(x1);
        double y2 = function.getFunctionValue(x2);

        // Площадь трапеции
        double segmentArea = (y1 + y2) * (x2 - x1) / 2.0;
        integral += segmentArea;

        x1 = x2;
    }

    return integral;
}
```

Задание 2

Реализация последовательной версии программы

Создан пакет threads с классом Task, содержащим:

- ссылку на интегрируемую функцию

- границы области интегрирования

- шаг дискретизации

- количество выполняемых заданий

Реализован метод nonThread() с последовательным выполнением:

1) Генерация случайных параметров для задания

2) Вывод исходных данных

3) Вычисление интеграла

4) Вывод результатов

Код:

```
package threads;

import functions.Function;

public class Task {  14 usages
    private Function function;  2 usages
    private double leftBound;  2 usages
    private double rightBound;  2 usages
    private double discretizationStep;  2 usages
    private int taskCount;  2 usages
    private double base;  2 usages

    // Конструктор, геттеры и сеттеры...
    public Task() {}  3 usages

    public Function getFunction() { return function; }
    public void setFunction(Function function) { this.function = function; }

    public double getLeftBound() { return leftBound; }  2 usages
    public void setLeftBound(double leftBound) { this.leftBound = leftBound; }  2 usages

    public double getRightBound() { return rightBound; }  2 usages
    public void setRightBound(double rightBound) { this.rightBound = rightBound; }  2 usages

    public double getDiscretizationStep() { return discretizationStep; }  2 usages
    public void setDiscretizationStep(double discretizationStep) { this.discretizationStep = discretizationStep; }

    public int getTaskCount() { return taskCount; }  4 usages
    public void setTaskCount(int taskCount) { this.taskCount = taskCount; }  3 usages

    public double getBase() { return base; }  2 usages
    public void setBase(double base) { this.base = base; }  2 usages
}
```

Результат (первые 25 из 100):

№	Основание	От	До	Шаг	Результат
1	1,063	60,2	198,6	0,858	10978,743503
2	9,933	87,0	177,6	0,682	191,924609
3	3,581	39,3	118,2	0,750	267,136072
4	8,182	59,0	109,5	0,633	106,016432
5	9,701	57,1	194,1	0,998	288,213506
6	7,247	84,8	140,9	0,939	133,562420
7	6,665	26,2	156,1	0,255	302,000050
8	4,849	70,3	140,6	0,868	206,524951
9	3,438	31,2	139,8	0,834	384,636292
10	4,840	25,4	112,1	0,240	228,406435
11	8,723	48,5	138,9	0,694	187,749906
12	9,448	40,6	132,4	0,633	180,188831
13	2,222	44,7	153,4	0,613	617,960125
14	2,106	51,3	136,6	0,238	516,431137
15	2,386	18,7	139,2	0,185	588,578059
16	7,254	56,2	163,9	0,203	253,101107
17	3,166	75,5	186,0	0,572	464,099846
18	9,735	98,9	105,7	0,917	13,841294
19	4,526	49,1	177,7	0,696	397,663689
20	4,661	36,9	123,3	0,227	243,028621
21	3,699	98,5	113,8	0,763	54,768433
22	9,885	39,2	124,2	0,402	161,397376
23	5,464	61,0	195,6	0,239	380,812729
24	1,455	6,0	114,2	0,260	1125,104842
25	3,939	27,7	104,6	0,127	231,790207

Задание 3

Реализация простой многопоточной версии

Созданы классы:

- SimpleGenerator (реализует Runnable) - генерирует задания

- SimpleIntegrator (реализует Runnable) - вычисляет интегралы

Выявленные проблемы:

1) Исключение NullPointerException - решено простой проверкой

2) Несогласованность данных в заданиях - решено использованием synchronized блоков

Реализован метод simpleThreads() для запуска двух потоков.

Код SimpleGenerator:

```
public class SimpleGenerator implements Runnable {  1 usage
    private final Task task;  8 usages
    private final Random random = new Random();  4 usages

    public SimpleGenerator(Task task) {  1 usage
        this.task = task;
    }

    @Override
    public void run() {
        try {
            for (int i = 0; i < task.getTaskCount(); i++) {
                // Генерируем параметры
                double base = 1 + random.nextDouble() * 9;
                double leftBound = 1 + random.nextDouble() * 99;
                double rightBound = 100 + random.nextDouble() * 100;
                double step = 0.1 + random.nextDouble() * 0.9;

                // Создаем функцию
                Log logFunction = new Log(base);

                // Синхронизируем доступ к объекту Task
                synchronized (task) {
                    // Устанавливаем параметры в задание
                    task.setFunction(logFunction);
                    task.setLeftBound(leftBound);
                    task.setRightBound(rightBound);
                    task.setDiscretizationStep(step);
                    task.setBase(base);
                }

                // Небольшая пауза для наглядности
                Thread.sleep( millis: 10);
            }
        } catch (InterruptedException e) {
            System.out.println("Генератор был прерван: " + e.getMessage());
            Thread.currentThread().interrupt();
        }
    }
}
```

ть

Код SimpleIntegrator:

```
public class SimpleIntegrator implements Runnable {  1 usage
    private final Task task;  8 usages

    public SimpleIntegrator(Task task) { this.task = task; }

    @Override
    public void run() {
        try {
            for (int i = 0; i < task.getTaskCount(); i++) {
                double leftBound, rightBound, step, base;
                Function function;

                // Синхронизируем доступ к объекту Task
                synchronized (task) {
                    // Получаем параметры из задания
                    function = task.getFunction();
                    leftBound = task.getLeftBound();
                    rightBound = task.getRightBound();
                    step = task.getDiscretizationStep();
                    base = task.getBase();
                }

                // Проверяем, что функция не null
                if (function == null) {
                    synchronized (System.out) {
                        System.out.printf("%3d | %9s | %5s | %6s | %5s | %11s\n",
                                i + 1, "ОШИБКА", "-", "-", "-", "NULL");
                    }
                    continue;
                }
```

```
                try {
                    // Вычисляем интеграл
                    double integralResult = Functions.integrate(function, leftBound, rightBound, step);

                    // Синхронизируем вывод всей строки
                    synchronized (System.out) {
                        System.out.printf("%3d | %9.3f | %5.1f | %6.1f | %5.3f | %11.6f\n",
                                i + 1, base, leftBound, rightBound, step, integralResult);
                    }

                } catch (IllegalArgumentException e) {
                    synchronized (System.out) {
                        System.out.printf("%3d | %9.3f | %5.1f | %6.1f | %5.3f | %11s\n",
                                i + 1, base, leftBound, rightBound, step, "ОШИБКА");
                    }
                }

                // Небольшая пауза для наглядности
                Thread.sleep( millis: 10);
            }
        } catch (InterruptedException e) {
            System.out.println("Интегратор был прерван: " + e.getMessage());
            Thread.currentThread().interrupt();
        }
    }
}
```

Результат:

```
=== МНОГОПОТОЧНАЯ ВЕРСИЯ ===
Многопоточное выполнение 100 заданий:
================================================================
№   | Основание | От    | До     | Шаг   | Результат
----|-----------|-------|--------|-------|-----------
Приоритет генератора: 5
Приоритет интегратора: 5
Все многопоточные задания выполнены!
Время выполнения: 1118 мс
```

```
=== МНОГОПОТОЧНАЯ ВЕРСИЯ ===
Многопоточное выполнение 100 заданий:
================================================================
№   | Основание | От    | До     | Шаг   | Результат
----|-----------|-------|--------|-------|-----------
Приоритет генератора: 10
Приоритет интегратора: 1
Все многопоточные задания выполнены!
Время выполнения: 1129 мс
```

```
=== МНОГОПОТОЧНАЯ ВЕРСИЯ ===
Многопоточное выполнение 100 заданий:
================================================================
№   | Основание | От    | До     | Шаг   | Результат
----|-----------|-------|--------|-------|-----------
Приоритет генератора: 1
Приоритет интегратора: 10
Все многопоточные задания выполнены!
Время выполнения: 1106 мс
```

Задание 4

Реализация усовершенствованной многопоточной версии

Созданы классы:

- Generator (расширяет Thread) с использованием семафора

- Integrator (расширяет Thread) с использованием семафора

Решены проблемы:

- Обеспечена полная обработка всех заданий

- Реализована корректная обработка прерывания потоков

Код Generator:

```
public class Generator extends Thread {  2 usages
    private final Task task;  7 usages
    private final Semaphore semaphore;  3 usages
    private final Random random = new Random();  4 usages

    public Generator(Task task, Semaphore semaphore) {...}

    @Override
    public void run() {
        try {
            for (int i = 0; i < task.getTaskCount(); i++) {
                // Проверяем, не прервали ли нас
                if (isInterrupted()) {
                    System.out.println("Generator: меня прервали, выхожу!");
                    return;
                }

                // Генерируем параметры
                double base = 1 + random.nextDouble() * 9;
                double leftBound = 1 + random.nextDouble() * 99;
                double rightBound = 100 + random.nextDouble() * 100;
                double step = 0.1 + random.nextDouble() * 0.9;

                Log logFunction = new Log(base);

                // Используем семафор вместо synchronized
                semaphore.beginWrite();
                try {
                    task.setFunction(logFunction);
                    task.setLeftBound(leftBound);
                    task.setRightBound(rightBound);
                    task.setDiscretizationStep(step);
                    task.setBase(base);

                    // Вывод для отладки (можно убрать)
                    System.out.printf("Generator: создал задание %d\n", i + 1);
                } finally {
                    semaphore.endWrite();
                }
```

```
                // Небольшая пауза
                Thread.sleep( millis: 10);
            }
        } catch (InterruptedException e) {
            System.out.println("Generator: прервали во время сна, выхожу!");
            Thread.currentThread().interrupt();
        }
    }
}
```

Код Integrator:

```
public class Integrator extends Thread {  2 usages
    private final Task task;  7 usages
    private final Semaphore semaphore;  3 usages

    public Integrator(Task task, Semaphore semaphore) {...}

    @Override
    public void run() {
        try {
            for (int i = 0; i < task.getTaskCount(); i++) {
                // Проверяем прерывание
                if (isInterrupted()) {
                    System.out.println("Integrator: меня прервали, выхожу!");
                    return;
                }

                double leftBound, rightBound, step, base;
                Function function;

                // Используем семафор вместо synchronized
                semaphore.beginRead();
                try {
                    function = task.getFunction();
                    leftBound = task.getLeftBound();
                    rightBound = task.getRightBound();
                    step = task.getDiscretizationStep();
                    base = task.getBase();
                } finally {
                    semaphore.endRead();
                }

                // Проверяем данные
                if (function == null) {
                    System.out.printf("Integrator: задание %d - нет функции\n", i + 1);
                    continue;
                }
```

```
                try {
                    // Вычисляем интеграл
                    double integralResult = Functions.integrate(function, leftBound, rightBound, step);

                    // Выводим результат
                    synchronized (System.out) {
                        System.out.printf("%3d | %9.3f | %5.1f | %6.1f | %5.3f | %11.6f\n",
                                i + 1, base, leftBound, rightBound, step, integralResult);
                    }
                } catch (IllegalArgumentException e) {
                    synchronized (System.out) {
                        System.out.printf("%3d | %9.3f | %5.1f | %6.1f | %5.3f | %11s\n",
                                i + 1, base, leftBound, rightBound, step, "ОШИБКА");
                    }
                }

                Thread.sleep( millis: 10);
            }
        } catch (InterruptedException e) {
            System.out.println("Integrator: прервали во время сна, выхожу!");
            Thread.currentThread().interrupt();
        }
    }
}
```

Результат:

```
=== СЛОЖНАЯ МНОГОПОТОЧНАЯ ВЕРСИЯ (С СЕМАФОРОМ) ===
СЛОЖНАЯ многопоточная версия с семафором:
================================================================
№   | Основание | От    | До     | Шаг   | Результат
----|-----------|-------|--------|-------|------------
Приоритет генератора: 5
Приоритет интегратора: 5
Generator: создал задание 1
  1 |     8,467 |  24,5 |  166,2 | 0,319 |  294,876161
Generator: создал задание 2
  2 |     9,174 |  26,2 |  150,9 | 0,901 |  246,752778
Generator: создал задание 3
  3 |     5,808 |  70,2 |  149,8 | 0,867 |  211,618785
Generator: создал задание 4
  4 |     8,874 |  83,8 |  169,0 | 0,168 |  188,211691
Generator: создал задание 5
  5 |     5,443 |  28,8 |  126,6 | 0,556 |  246,790012

ОСНОВНОЙ ПОТОК: ПРЕРЫВАЮ РАБОТУ ПОТОКОВ!
Generator: прервали во время сна, выхожу!
Integrator: прервали во время сна, выхожу!
================================================================
Сложная многопоточная версия завершена (возможно, не все задания)
```