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 Волков Д.Н.

Группа 6301-030301D

Руководитель Борисов Д. С.

Оценка _____

САМАРА 2025

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 должно выполняться по методу трапеций.

```
public static double integrate(Function function, double leftX, double rightX, double step) {
    if (Math.abs(leftX - function.getLeftDomainBorder()) < 1e-9 || Math.abs(rightX - function.getRightDomainBorder()) < 1e-9) {
        throw new IllegalArgumentException("Интервал интегрирования [" + leftX + ", " + rightX + "] выходит за границы области определения функции ");
    }

    if (step <= 0) {
        throw new IllegalArgumentException(s: "Шаг дискретизации должен быть положительным.");
    }
    if (leftX > rightX) {
        return -integrate(function, rightX, leftX, step);
    }
    if (leftX == rightX) {
        return 0;
    }

    double integralSum = 0.0;
    double currentX = leftX;

    while (currentX < rightX) {
        step = Math.min(step, rightX - currentX);
        double x1 = currentX;
        double x2 = currentX + step;
        double y1 = function.getFunctionValue(x1);
        double y2 = function.getFunctionValue(x2);
        integralSum += (y1 + y2) / 2.0 * step;
        currentX += step;
    }
    return integralSum;
}
}
```

В main() создал проверку для работы метода интегрирования

```
Run | Debug | Run main | Debug main
public static void main(String[] args) throws IOException, FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException,
    System.out.println(x: "\t\t Нахождение приближенного решения интеграла");
    Intergral();

    System.out.println(x: "\n\t\t Генерирование и решение Task-ов последовательно\n");
    nonThread();

    //System.out.println("\n\t\t Генерирование и решение Task-ов с помощью SimpleGenerator и SimpleIntegrator\n");
    //simpleThreads();

    //System.out.println("\n\t\t Генерирование и решение Task-ов с помощью Generator и Integrator\n");
    //complicatedThreads();
}

public static void Intergral(){
    Function exp = new Exp();
    int leftX = 0;
    int rightX = 1;
    double step = 0.000125;
    double theoreticalResult = Math.E - 1;
    double result = Functions.integrate(exp, leftX, rightX, step);

    while(Math.abs(theoreticalResult - result) > 1e-8){
        result = Functions.integrate(exp, leftX, rightX, step);
        step = step / 2;
    }
    System.out.printf(format: "\nЗначение интерграла = %.7f с точностью 1e-7 и с шагом = %.7f", result, step);
    System.out.printf(format: "\nЗначение интеграла %.7f\n", theoreticalResult);
}
```

```
                Нахождение приближенного решения интеграла

Значение интерграла = 1,7182818 с точностью 1e-7 и с шагом = 0,0001250
Значение интеграла 1,7182818
```

Шаг дискретизации для точности вычисления 1e-7 равен 0,000125.

Задание 2

Создал пакет threads.

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

```java
package threads;

import functions.Function;


public class Task {
    private Function function;
    private double leftBorder;
    private double rightBorder;
    private double step;
    private int taskCount;
    private volatile boolean taskReadyForIntegration = false;

    public Task(int taskCount) {
        this.taskCount = taskCount;
    }

    public Task() {
        this.taskCount = 0;
    }

    public Function getFunction() {
        return function;
    }

    public synchronized void setFunction(Function function) {
        this.function = function;
        this.taskReadyForIntegration = true;
        notifyAll();
    }

    public double getLeftBorder() {
        return leftBorder;
    }

    public void setLeftBorder(double leftBorder) {
        this.leftBorder = leftBorder;
    }

    public double getRightBorder() {
        return rightBorder;
    }

    public void setRightBorder(double rightBorder) {
        this.rightBorder = rightBorder;
    }
```

```java
    public double getStep() {
        return step;
    }

    public void setStep(double step) {
        this.step = step;
    }

    public void setTaskCount(int taskCount) {
        this.taskCount = taskCount;
    }

    public double getTaskCount() {
        return taskCount;
    }

    public synchronized boolean isTaskReadyForIntegration() {
        return taskReadyForIntegration;
    }

    public synchronized void resetTaskReadiness() {
        this.taskReadyForIntegration = false;
    }
}
```

Также в классе есть поле taskReadyForIntegration и методы для флага, которые далее будут необходимы для нормальной работы simpleGenerator и SimpleIntegrator.

В main() создал метод последовательно генерирующий и решающий task-и со всеми необходимыми условиями, поставленными в задании.

```java
public static void nonThread() {
    System.out.println(x: "Последовательная версия (nonThread)");

    Random random = new Random();
    Task task = new Task();
    task.setTaskCount(taskCount: 100);

    for (int i = 0; i < task.getTaskCount(); i++) {
        try {
            double base = 1 + random.nextDouble() * 9;
            Function logFunction = new Log(base);

            double left = random.nextDouble() * 100;
            double right = 100 + random.nextDouble() * 100;
            double step = random.nextDouble();

            task.setFunction(logFunction);
            task.setLeftBorder(left);
            task.setRightBorder(right);
            task.setStep(step);

            System.out.printf(format: "Задание: левая граница = %.6f, правая граница = %.6f, шаг = %.6f%n", left, right, step);
            double result = Functions.integrate(logFunction, left, right, step);
            System.out.printf(format: "Результат: левая граница = %.6f, правая граница = %.6f, шаг = %.6f, значение интеграла =  %.6f", left, right, step, result);
            System.out.printf(format: "\n---------\n");
        } catch (IllegalArgumentException e) {
            System.out.println("Ошибка при итерации " + (i + 1) + ": " + e.getMessage());
        }
    }
    System.out.println(x: "Конец последовательной версии(nonThread)");
}
```

```
                Генерирование и решение Task-ов последовательно

Последовательная версия (nonThread)
Задание: левая граница = 27,292170, правая граница = 179,825554, шаг = 0,647127
Результат: левая граница = 27,292170, правая граница = 179,825554, шаг = 0,647127, значение интеграла =  383,475218
---------
Задание: левая граница = 95,774597, правая граница = 116,882949, шаг = 0,015114
Результат: левая граница = 95,774597, правая граница = 116,882949, шаг = 0,015114, значение интеграла =  57,961123
---------
Задание: левая граница = 49,169760, правая граница = 119,833482, шаг = 0,681847
Результат: левая граница = 49,169760, правая граница = 119,833482, шаг = 0,681847, значение интеграла =  197,052003
---------
Задание: левая граница = 1,173059, правая граница = 134,186518, шаг = 0,588795
Результат: левая граница = 1,173059, правая граница = 134,186518, шаг = 0,588795, значение интеграла =  255,509924
---------
Задание: левая граница = 62,888295, правая граница = 157,263190, шаг = 0,779372
Результат: левая граница = 62,888295, правая граница = 157,263190, шаг = 0,779372, значение интеграла =  192,728537
---------
Задание: левая граница = 13,020451, правая граница = 120,676234, шаг = 0,174103
Результат: левая граница = 13,020451, правая граница = 120,676234, шаг = 0,174103, значение интеграла =  216,207999
---------
Задание: левая граница = 63,564929, правая граница = 170,885789, шаг = 0,070186
Результат: левая граница = 63,564929, правая граница = 170,885789, шаг = 0,070186, значение интеграла =  250,671622
---------
Задание: левая граница = 46,191591, правая граница = 101,471908, шаг = 0,891349
Результат: левая граница = 46,191591, правая граница = 101,471908, шаг = 0,891349, значение интеграла =  165,220805
---------
Задание: левая граница = 25,088444, правая граница = 100,237182, шаг = 0,160206
Результат: левая граница = 25,088444, правая граница = 100,237182, шаг = 0,160206, значение интеграла =  337,512946
---------
Задание: левая граница = 75,721059, правая граница = 161,850520, шаг = 0,142844
Результат: левая граница = 75,721059, правая граница = 161,850520, шаг = 0,142844, значение интеграла =  258,978521
---------
Задание: левая граница = 55,780363, правая граница = 182,465844, шаг = 0,124176
Результат: левая граница = 55,780363, правая граница = 182,465844, шаг = 0,124176, значение интеграла =  724,962128
---------
Задание: левая граница = 34,532790, правая граница = 196,597027, шаг = 0,812616
Результат: левая граница = 34,532790, правая граница = 196,597027, шаг = 0,812616, значение интеграла =  4454,033524
---------
Задание: левая граница = 39,853161, правая граница = 138,503118, шаг = 0,943942
Результат: левая граница = 39,853161, правая граница = 138,503118, шаг = 0,943942, значение интеграла =  507,558002
---------
```

.....

```
---------
Задание: левая граница = 7,131324, правая граница = 155,454168, шаг = 0,835457
Результат: левая граница = 7,131324, правая граница = 155,454168, шаг = 0,835457, значение интеграла =  320,452783
---------
Задание: левая граница = 44,423885, правая граница = 152,973642, шаг = 0,177776
Результат: левая граница = 44,423885, правая граница = 152,973642, шаг = 0,177776, значение интеграла =  299,566575
---------
Задание: левая граница = 14,501836, правая граница = 120,803039, шаг = 0,895042
Результат: левая граница = 14,501836, правая граница = 120,803039, шаг = 0,895042, значение интеграла =  228,024648
---------
Задание: левая граница = 67,261067, правая граница = 146,748755, шаг = 0,511797
Результат: левая граница = 67,261067, правая граница = 146,748755, шаг = 0,511797, значение интеграла =  247,356202
---------
Задание: левая граница = 59,236942, правая граница = 146,189951, шаг = 0,435081
Результат: левая граница = 59,236942, правая граница = 146,189951, шаг = 0,435081, значение интеграла =  205,478896
---------
Задание: левая граница = 17,075930, правая граница = 132,786611, шаг = 0,011813
Результат: левая граница = 17,075930, правая граница = 132,786611, шаг = 0,011813, значение интеграла =  294,144411
---------
Задание: левая граница = 30,158534, правая граница = 177,266015, шаг = 0,875631
Результат: левая граница = 30,158534, правая граница = 177,266015, шаг = 0,875631, значение интеграла =  301,071476
---------
Задание: левая граница = 58,899917, правая граница = 147,381480, шаг = 0,534871
Результат: левая граница = 58,899917, правая граница = 147,381480, шаг = 0,534871, значение интеграла =  195,192410
---------
Задание: левая граница = 88,321177, правая граница = 194,109899, шаг = 0,224674
Результат: левая граница = 88,321177, правая граница = 194,109899, шаг = 0,224674, значение интеграла =  264,016127
---------
Задание: левая граница = 52,675562, правая граница = 199,252281, шаг = 0,148732
Результат: левая граница = 52,675562, правая граница = 199,252281, шаг = 0,148732, значение интеграла =  1243,797788
---------
Задание: левая граница = 6,230325, правая граница = 127,824124, шаг = 0,369981
Результат: левая граница = 6,230325, правая граница = 127,824124, шаг = 0,369981, значение интеграла =  240,423859
---------
Задание: левая граница = 17,746331, правая граница = 134,970086, шаг = 0,774594
Результат: левая граница = 17,746331, правая граница = 134,970086, шаг = 0,774594, значение интеграла =  269,317164
---------
Задание: левая граница = 21,763947, правая граница = 161,610169, шаг = 0,729008
Результат: левая граница = 21,763947, правая граница = 161,610169, шаг = 0,729008, значение интеграла =  5492,473833
---------
Задание: левая граница = 68,495138, правая граница = 115,622171, шаг = 0,284453
Результат: левая граница = 68,495138, правая граница = 115,622171, шаг = 0,284453, значение интеграла =  145,941166
---------
Задание: левая граница = 75,938105, правая граница = 176,087908, шаг = 0,017059
Результат: левая граница = 75,938105, правая граница = 176,087908, шаг = 0,017059, значение интеграла =  421,490732
---------
Задание: левая граница = 2,772183, правая граница = 176,978294, шаг = 0,932033
Результат: левая граница = 2,772183, правая граница = 176,978294, шаг = 0,932033, значение интеграла =  425,017785
---------
Конец последовательной версии(nonThread)
```

Задание №3

В пакете threads создал два следующих класса.

Класс SimpleGenerator реализовывает интерфейс Runnable, получает в конструкторе и сохраняет в своё поле ссылку на объект типа Task, а в методе run() в цикле формирует задачи и заносит в полученный объект задания, а также выводит сообщение в консоль.

```java
package threads;

import functions.Function;
import functions.basic.*;
import java.util.Random;

public class SimpleGenerator implements Runnable {
    private final Task task;

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

    @Override
    public void run(){
        Random random = new Random();

        for (int i = 0; i < task.getTaskCount(); i++) {
            try {
                double base = 1 + random.nextDouble() * 9;
                Function logFunction = new Log(base);

                double left = random.nextDouble() * 100;
                double right = 100 + random.nextDouble() * 100;
                double step = random.nextDouble();

                synchronized (task){
                    task.setFunction(logFunction);
                    task.setLeftBorder(left);
                    task.setRightBorder(right);
                    task.setStep(step);
                    System.out.printf(format: "(SimpleGenerator) Задание: левая граница = %.6f, правая граница = %.6f, шаг = %.6f%n", left, right, step);
                }
                Thread.sleep(millis: 1);

            } catch (IllegalArgumentException e) {
                System.out.println("Ошибка при генерации" + e.getMessage());
            } catch (InterruptedException e) {
                e.getMessage();
            }
        }
        System.out.println(x: "Конец simpleThreads");
    }
}
```

Добавил задержку в 1 миллисекунду для нормального вывода в консоль (получил это значение экспериментально с приоритетами в MIN на генератор и MAX в интеграторе).

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

```java
package threads;

import functions.Functions;

public class SimpleIntegrator implements Runnable {
    private final Task task;

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

    @Override
    public void run(){
        for (int i = 0; i < task.getTaskCount(); ++i){
            try {
                double left;
                double right;
                double step;
                double result;

                synchronized (task) {
                    while (!task.isTaskReadyForIntegration()) {
                        task.wait();
                    }
                    left = task.getLeftBorder();
                    right = task.getRightBorder();
                    step = task.getStep();
                    result = Functions.integrate(task.getFunction(), left, right, step);
                    System.out.printf(format: "(SimpleIntegrator) Результат: левая граница = %.6f, правая граница = %.6f, шаг = %.6f, значение интеграла =  %.6f",
                    left, right, step, result);
                    System.out.printf(format: "\n---------\n");
                    task.resetTaskReadiness();
                }

            } catch(InterruptedException e){
                System.out.println("При интегрировании произошла ошибка" + e.getMessage());
            } catch (Exception e) {
                System.out.println("В SimpleIntegrator ошибка: " + e.getMessage());
            }
        }
    }
```

Здесь использую флаг

while (!task.isTaskReadyForIntegration()){

task.wait;

}

(по сути пока “не” false, simpleIntegrator ждет)

который заставляет ждать simpleIntegrator момента, когда он получит задание для решения, после он его получает, решает и возвращает значение false для taskReadyForIntegration.

В main() создал метод для проверки работы классов.

```java
import functions.*;
import functions.basic.*;
import java.io.*;
import java.util.Random;
import java.util.concurrent.Semaphore;
import threads.*;

public class Main {

    Run | Debug | Run main | Debug main
    public static void main(String[] args) throws IOException, FunctionPointIndexOutOfBoundsException, InappropriateFunctionP
        System.out.println(x: "\t\t Нахождение приближенного решения интеграла");
        Intergral();

        //System.out.println("\n\t\t Генерирование и решение Task-ов последовательно\n");
        //nonThread();

        System.out.println(x: "\n\t\t Генерирование и решение Task-ов с помощью SimpleGenerator и SimpleIntegrator\n");
        simpleThreads();

        //System.out.println("\n\t\t Генерирование и решение Task-ов с помощью Generator и Integrator\n");
        //complicatedThreads();
    }
```

```java
    public static void simpleThreads() throws InterruptedException{
        Task task = new Task();
        task.setTaskCount(taskCount: 100);

        Thread simpleGenerator = new Thread(new SimpleGenerator(task));
        Thread simpleIntegrator = new Thread(new SimpleIntegrator(task));

        simpleGenerator.setPriority(Thread.MIN_PRIORITY);
        simpleIntegrator.setPriority(Thread.MAX_PRIORITY);

        simpleIntegrator.start();
        simpleGenerator.start();
    }
```

```
                Генерирование и решение Task-ов с помощью SimpleGenerator и SimpleIntegrator

(SimpleGenerator) Задание: левая граница = 47,719964, правая граница = 131,716647, шаг = 0,580488
(SimpleIntegrator) Результат: левая граница = 47,719964, правая граница = 131,716647, шаг = 0,580488, значение интеграла =  360,896491
---------
(SimpleGenerator) Задание: левая граница = 1,564663, правая граница = 134,217850, шаг = 0,280390
(SimpleIntegrator) Результат: левая граница = 1,564663, правая граница = 134,217850, шаг = 0,280390, значение интеграла =  285,162619
---------
(SimpleGenerator) Задание: левая граница = 72,159064, правая граница = 196,810371, шаг = 0,432858
(SimpleIntegrator) Результат: левая граница = 72,159064, правая граница = 196,810371, шаг = 0,432858, значение интеграла =  1781,796560
---------
(SimpleGenerator) Задание: левая граница = 95,171899, правая граница = 140,851634, шаг = 0,569451
(SimpleIntegrator) Результат: левая граница = 95,171899, правая граница = 140,851634, шаг = 0,569451, значение интеграла =  215,307613
---------
(SimpleGenerator) Задание: левая граница = 58,166768, правая граница = 161,530529, шаг = 0,554786
(SimpleIntegrator) Результат: левая граница = 58,166768, правая граница = 161,530529, шаг = 0,554786, значение интеграла =  234,911308
---------
(SimpleGenerator) Задание: левая граница = 48,385794, правая граница = 190,081614, шаг = 0,282748
(SimpleIntegrator) Результат: левая граница = 48,385794, правая граница = 190,081614, шаг = 0,282748, значение интеграла =  922,812594
---------
(SimpleGenerator) Задание: левая граница = 23,091215, правая граница = 172,872598, шаг = 0,045202
(SimpleIntegrator) Результат: левая граница = 23,091215, правая граница = 172,872598, шаг = 0,045202, значение интеграла =  293,953810
---------
(SimpleGenerator) Задание: левая граница = 3,395692, правая граница = 147,910116, шаг = 0,098613
(SimpleIntegrator) Результат: левая граница = 3,395692, правая граница = 147,910116, шаг = 0,098613, значение интеграла =  1562,708769
---------
(SimpleGenerator) Задание: левая граница = 9,016007, правая граница = 182,219031, шаг = 0,509669
(SimpleIntegrator) Результат: левая граница = 9,016007, правая граница = 182,219031, шаг = 0,509669, значение интеграла =  484,295576
---------
(SimpleGenerator) Задание: левая граница = 71,013780, правая граница = 144,585577, шаг = 0,447900
(SimpleIntegrator) Результат: левая граница = 71,013780, правая граница = 144,585577, шаг = 0,447900, значение интеграла =  334,363636
---------
(SimpleGenerator) Задание: левая граница = 82,340685, правая граница = 127,314907, шаг = 0,711471
(SimpleIntegrator) Результат: левая граница = 82,340685, правая граница = 127,314907, шаг = 0,711471, значение интеграла =  110,279627
---------
(SimpleGenerator) Задание: левая граница = 71,884591, правая граница = 197,938663, шаг = 0,421696
(SimpleIntegrator) Результат: левая граница = 71,884591, правая граница = 197,938663, шаг = 0,421696, значение интеграла =  381,095031
---------
(SimpleGenerator) Задание: левая граница = 16,716931, правая граница = 157,218434, шаг = 0,502557
(SimpleIntegrator) Результат: левая граница = 16,716931, правая граница = 157,218434, шаг = 0,502557, значение интеграла =  264,371812
---------
(SimpleGenerator) Задание: левая граница = 39,370541, правая граница = 103,326063, шаг = 0,773338
(SimpleIntegrator) Результат: левая граница = 39,370541, правая граница = 103,326063, шаг = 0,773338, значение интеграла =  205,576742
---------
(SimpleGenerator) Задание: левая граница = 72,231087, правая граница = 153,741937, шаг = 0,225732
(SimpleIntegrator) Результат: левая граница = 72,231087, правая граница = 153,741937, шаг = 0,225732, значение интеграла =  437,037425
---------
```

....

```
---------
(SimpleGenerator) Задание: левая граница = 48,011035, правая граница = 101,003931, шаг = 0,401854
(SimpleIntegrator) Результат: левая граница = 48,011035, правая граница = 101,003931, шаг = 0,401854, значение интеграла =  108,902138
---------
(SimpleGenerator) Задание: левая граница = 56,075854, правая граница = 196,085857, шаг = 0,913988
(SimpleIntegrator) Результат: левая граница = 56,075854, правая граница = 196,085857, шаг = 0,913988, значение интеграла =  369,538948
---------
(SimpleGenerator) Задание: левая граница = 4,127185, правая граница = 168,911307, шаг = 0,946385
(SimpleIntegrator) Результат: левая граница = 4,127185, правая граница = 168,911307, шаг = 0,946385, значение интеграла =  21639,206155
---------
(SimpleGenerator) Задание: левая граница = 12,481738, правая граница = 146,918629, шаг = 0,787401
(SimpleIntegrator) Результат: левая граница = 12,481738, правая граница = 146,918629, шаг = 0,787401, значение интеграла =  332,082637
---------
(SimpleGenerator) Задание: левая граница = 63,877257, правая граница = 174,269079, шаг = 0,271610
(SimpleIntegrator) Результат: левая граница = 63,877257, правая граница = 174,269079, шаг = 0,271610, значение интеграла =  268,709667
---------
(SimpleGenerator) Задание: левая граница = 85,811583, правая граница = 164,168696, шаг = 0,845675
(SimpleIntegrator) Результат: левая граница = 85,811583, правая граница = 164,168696, шаг = 0,845675, значение интеграла =  307,567076
---------
(SimpleGenerator) Задание: левая граница = 13,079701, правая граница = 178,865175, шаг = 0,171921
(SimpleIntegrator) Результат: левая граница = 13,079701, правая граница = 178,865175, шаг = 0,171921, значение интеграла =  322,521901
---------
(SimpleGenerator) Задание: левая граница = 57,562631, правая граница = 135,784625, шаг = 0,360980
(SimpleIntegrator) Результат: левая граница = 57,562631, правая граница = 135,784625, шаг = 0,360980, значение интеграла =  154,798164
---------
(SimpleGenerator) Задание: левая граница = 77,263386, правая граница = 107,120905, шаг = 0,714843
(SimpleIntegrator) Результат: левая граница = 77,263386, правая граница = 107,120905, шаг = 0,714843, значение интеграла =  84,742118
---------
Конец simpleThreads
```

(Пробовал изменять количество Task-ов и несколько раз запускал программу, ошибок не возникало)

Задание №4

В пакете threads создал два следующих класса.

Класс Generator драсширяет класс Thread, получает в конструкторе и сохраняет в свои поля ссылки на объект типа Task и на объект семафора, а в методе run() выполняет те же действия, что и в предыдущей версии генерирующего класса.

```java
package threads;

import functions.Function;
import functions.basic.*;
import java.util.Random;
import java.util.concurrent.Semaphore;

public class Generator extends Thread{
    private final Task task;
    private final Semaphore dataReady;
    private final Semaphore dataProcessed;

    public Generator(Task task, Semaphore dataReady, Semaphore dataProcessed){
        this.task = task;
        this.dataReady = dataReady;
        this.dataProcessed = dataProcessed;
    }

    @Override
    public void run(){
        Random random = new Random();
        try {
            for (int i = 0; i < task.getTaskCount(); i++) {

                double base = 1 + random.nextDouble() * 9;
                Function logFunction = new Log(base);

                double left = random.nextDouble() * 100;
                double right = 100 + random.nextDouble() * 100;
                double step = random.nextDouble();

                dataProcessed.acquire();
                task.setFunction(logFunction);
                task.setLeftBorder(left);
                task.setRightBorder(right);
                task.setStep(step);

                System.out.printf(format: "(Generator %d)Задание: левая граница = %.6f, правая граница = %.6f, шаг = %.6f%n", i, left, right, step);
                dataReady.release();
            }
        } catch (InterruptedException e) {
            System.out.println(x: "Генератор был прерван");
        }
    }
}
```

Класс Integrator расширяет класс Thread, получает в конструкторе и сохраняет в свои поля ссылки на объект типа Task и на объект семафора, а в методе run() выполняет те же действия, что и в предыдущей версии интегрирующего класса.

```java
package threads;

import functions.Functions;
import java.util.concurrent.Semaphore;

public class Integrator extends Thread {
    private final Task task;
    private final Semaphore dataReady;
    private final Semaphore dataProcessed;

    public Integrator(Task task, Semaphore dataReady, Semaphore dataProcessed){
        this.task = task;
        this.dataReady = dataReady;
        this.dataProcessed = dataProcessed;
    }

    @Override
    public void run(){

        for (int i = 0; i < task.getTaskCount(); ++i){
            try {

                double left;
                double right;
                double step;
                double result;

                dataReady.acquire();
                    left = task.getLeftBorder();
                    right = task.getRightBorder();
                    step = task.getStep();

                    result = Functions.integrate(task.getFunction(), left, right, step);
                    System.out.printf(format: "(Integrator %d) Результат: левая граница = %.6f, правая граница = %.6f, шаг = %.6f, значение интеграла =  %.6f",
                    i, left, right, step, result);
                System.out.printf(format: "\n---------\n");
                    dataProcessed.release();

            } catch (InterruptedException e) {
                System.out.println(x: "Integrator прерван");
            }
        }
    }
}
```

В main() создал метод для проверки Generator и Integrator.

```java
import functions.*;
import functions.basic.*;
import java.io.*;
import java.util.Random;
import java.util.concurrent.Semaphore;
import threads.*;

public class Main {

    Run | Debug | Run main | Debug main
    public static void main(String[] args) throws IOException, FunctionPointIndexOutOfBoundsException, InappropriateFunctionPointException, Class
        System.out.println(x: "\t\t Нахождение приближенного решения интеграла");
        Intergral();

        //System.out.println("\n\t\t Генерирование и решение Task-ов последовательно\n");
        //nonThread();

        //System.out.println("\n\t\t Генерирование и решение Task-ов с помощью SimpleGenerator и SimpleIntegrator\n");
        //simpleThreads();

        System.out.println(x: "\n\t\t Генерирование и решение Task-ов с помощью Generator и Integrator\n");
        complicatedThreads();
    }
```

```java
    public static void complicatedThreads() throws InterruptedException{
        Task task = new Task();
        task.setTaskCount(taskCount: 100);
        Semaphore dataReady = new Semaphore(permits: 0);
        Semaphore dataProcessed = new Semaphore(permits: 1);

        Generator generator = new Generator(task, dataReady, dataProcessed);
        Integrator integrator = new Integrator(task, dataReady, dataProcessed);

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

        try {

        Thread.sleep(millis: 50);

        generator.interrupt();
        integrator.interrupt();

        generator.join();
        integrator.join();

        } catch (InterruptedException e) {
            System.out.println(x: "Main был остановлен");
        }
    }
}
```

Если закомментировать блок с искусственной остановкой, то классы корректно генерируют и решают task-и.

```
                Генерирование и решение Task-ов с помощью Generator и Integrator

(Generator 0)Задание: левая граница = 7,080244, правая граница = 169,514294, шаг = 0,729099
(Integrator 0) Результат: левая граница = 7,080244, правая граница = 169,514294, шаг = 0,729099, значение интеграла =  323,942902
---------
(Generator 1)Задание: левая граница = 17,815153, правая граница = 170,047800, шаг = 0,959071
(Integrator 1) Результат: левая граница = 17,815153, правая граница = 170,047800, шаг = 0,959071, значение интеграла =  308,665464
---------
(Generator 2)Задание: левая граница = 50,006745, правая граница = 166,130369, шаг = 0,940621
(Integrator 2) Результат: левая граница = 50,006745, правая граница = 166,130369, шаг = 0,940621, значение интеграла =  921,837196
---------
(Generator 3)Задание: левая граница = 80,858767, правая граница = 108,588000, шаг = 0,148110
(Integrator 3) Результат: левая граница = 80,858767, правая граница = 108,588000, шаг = 0,148110, значение интеграла =  70,312702
---------
(Generator 4)Задание: левая граница = 23,294317, правая граница = 151,863883, шаг = 0,553778
(Integrator 4) Результат: левая граница = 23,294317, правая граница = 151,863883, шаг = 0,553778, значение интеграла =  735,404513
---------
```

…

```
(Generator 94)Задание: левая граница = 89,511100, правая граница = 130,849328, шаг = 0,847992
(Integrator 94) Результат: левая граница = 89,511100, правая граница = 130,849328, шаг = 0,847992, значение интеграла =  211,259901
---------
(Generator 95)Задание: левая граница = 69,582317, правая граница = 130,463782, шаг = 0,436821
(Integrator 95) Результат: левая граница = 69,582317, правая граница = 130,463782, шаг = 0,436821, значение интеграла =  202,664892
---------
(Generator 96)Задание: левая граница = 48,499446, правая граница = 129,797519, шаг = 0,947350
(Integrator 96) Результат: левая граница = 48,499446, правая граница = 129,797519, шаг = 0,947350, значение интеграла =  280,863705
---------
(Generator 97)Задание: левая граница = 46,872735, правая граница = 197,459399, шаг = 0,193041
(Integrator 97) Результат: левая граница = 46,872735, правая граница = 197,459399, шаг = 0,193041, значение интеграла =  311,289143
---------
(Generator 98)Задание: левая граница = 33,245869, правая граница = 118,705283, шаг = 0,354614
(Integrator 98) Результат: левая граница = 33,245869, правая граница = 118,705283, шаг = 0,354614, значение интеграла =  185,783315
---------
(Generator 99)Задание: левая граница = 21,347112, правая граница = 188,455165, шаг = 0,574010
(Integrator 99) Результат: левая граница = 21,347112, правая граница = 188,455165, шаг = 0,574010, значение интеграла =  453,423646
---------
```

Для обработки корректного прерывания потоков, в Generator и Integrator есть блоки try-catch, которые позволяют поймать исключение InterruptedException об остановке.