ЛАБОРАТОРНАЯ РАБОТА №6

Многопоточное вычисление интегралов функций

по курсу
Объектно-ориентированное программирование

Группа 6204-010302D

Студент: С.0.Куропаткин.

Преподаватель: Борисов Дмитрий

Сергеевич.

Задание 1: Реализация метода интегрирования

В класс Functions добавлен метод для вычисления интеграла по методу трапеций:

```
public static double integrate(Function function, double left, double right,
double step) throws IllegalArgumentException,
InappropriateFunctionPointException {

    if (left >= right || step <= 0) {
        throw new IllegalArgumentException();
    }

    if (left < function.getLeftDomainBorder() || right >
function.getRightDomainBorder()) {
        throw new IllegalArgumentException();
    }

    double result = 0;
    double x = left;

    while (x + step < right) {
        result += (function.getFunctionValue(x) + function.getFunctionValue(x
+ step)) * step / 2;
        x += step;
    }

    // Последняя неполная трапеция
    if (x < right) {
        result += (function.getFunctionValue(x) +
function.getFunctionValue(right)) * (right - x) / 2;
    }

    return result;
}
```

Задание 2

Класс Task и последовательная версия

Создан пакет threads и класс Task:

```
package threads;

import functions.Function;

public class Task {

    private Function function;
    private double leftX, rightX, step;
    private int numTasks;

    public Task(int numTasks) {
        if (numTasks <= 0) {
            throw new IllegalArgumentException();
        }
        this.numTasks = numTasks;
    }

    public Function getFunction() {
        return function;
    }
```

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

    public double getLeftX() {
        return leftX;
    }

    public void setLeftX(double leftX) {
        this.leftX = leftX;
    }

    public double getRightX() {
        return rightX;
    }

    public void setRightX(double rightX) {
        if (rightX <= leftX) {
            throw new IllegalArgumentException();
        }
        this.rightX = rightX;
    }

    public double getStep() {
        return step;
    }

    public void setStep(double step) {
        if (step <= 0) {
            throw new IllegalArgumentException();
        }
        this.step = step;
    }

    public int getNumTasks() {
        return numTasks;
    }

    public void setNumTasks(int numTasks) {
        if (numTasks <= 0) {
            throw new IllegalArgumentException();
        }
        this.numTasks = numTasks;
    }
}
```

Задание 3

Простая многопоточная версия

Созданы классы SimpleGenerator и SimpleIntegrator:

```
package threads;

import functions.Function;
import functions.Functions;
import functions.basic.Log;

public class SimpleGenerator implements Runnable{
    private Task task;

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

```
    }

    public void run() {
        for (int i = 0; i < task.getNumTasks(); ++i) {
            double base = 1 + Math.random() * 9;
            Function function = Functions.power(new Log(base), 2);
            double leftX = Math.random() * 100;
            double rightX = Math.random() * 100 + 100;
            double step = Math.random() * 0.9 + 0.1;
            synchronized (task) {
                task.setFunction(function);
                task.setLeftX(leftX);
                task.setRightX(rightX);
                task.setStep(step);
                System.out.println("Source leftX = " + leftX + " rightX = " +
rightX + " step = " + step);
            }
        }
    }
}
```

```
package threads;

import functions.Function;
import functions.Functions;
import functions.InappropriateFunctionPointException;

public class SimpleIntegrator implements Runnable{
    private Task task;

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

    public void run(){
        for (int i = 0; i < task.getNumTasks(); ++i) {
            double leftX, rightX, step;
            Function function;
            if (task.getFunction() == null) {
                continue;
            }
            synchronized (task) {
                leftX = task.getLeftX();
                rightX = task.getRightX();
                step = task.getStep();
                function = task.getFunction();

                double result = 0;
                try {
                    result = Functions.integrate(function, leftX, rightX,
step);
                } catch (InappropriateFunctionPointException e) {
                    throw new RuntimeException(e);
                }
                System.out.println("Result leftX = " + leftX + " rightX = " +
rightX + " step = " + step + " Result: " + result);
            }
        }
    }
}
```

Задание 4

Усовершенствованная версия с семафором

Созданы классы Generator и Integrator:

```
package threads;

import functions.*;
import functions.basic.*;

import java.util.concurrent.Semaphore;

public class Generator extends Thread {
    private Task task;
    private Semaphore semaphore;

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

    @Override
    public void run() {
        for (int i = 0; i < task.getNumTasks(); ++i) {
            double base = 1 + Math.random() * 9;
            Function function = Functions.power(new Log(base), 2);
            double leftX = Math.random() * 100;
            double rightX = Math.random() * 100 + 100;
            double step = Math.random() * 0.9 + 0.1;

            try {
                semaphore.acquire();
                synchronized (task) {
                    task.setFunction(function);
                    task.setLeftX(leftX);
                    task.setRightX(rightX);
                    task.setStep(step);
                    System.out.println("Source leftX = " + leftX + " rightX =
" + rightX + " step = " + step);
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            } finally {
                semaphore.release();
            }
        }
    }
}
```

```
package threads;

import functions.Function;
import functions.Functions;
import functions.InappropriateFunctionPointException;

import java.util.concurrent.Semaphore;

public class Integrator extends Thread {
    private Task task;
    private Semaphore semaphore;

    public Integrator(Task task, Semaphore semaphore) {
```

```
        this.task = task;
        this.semaphore = semaphore;
    }

    @Override
    public void run() {
        for (int i = 0; i < task.getNumTasks(); ++i) {
            double leftX, rightX, step;
            Function function;

            try {
                semaphore.acquire();
                synchronized (task) {
                    leftX = task.getLeftX();
                    rightX = task.getRightX();
                    step = task.getStep();
                    function = task.getFunction();
                }

                double result = 0;
                try {
                    result = Functions.integrate(function, leftX, rightX,
step);
                } catch (InappropriateFunctionPointException e) {
                    throw new RuntimeException(e);
                }

                System.out.println("Result leftX = " + leftX + " rightX = " +
rightX + " step = " + step + " Result: " + result);
            } catch (InterruptedException e) {
                e.printStackTrace();
            } finally {
                semaphore.release();
            }
        }
    }
}
```

Тестирование написанных методов

```
import functions.*;
import functions.basic.*;
import functions.meta.*;
import threads.*;

import java.util.concurrent.Semaphore;

public class Main {
    public static void main(String[] args) throws
InappropriateFunctionPointException, InterruptedException {

        // Пример использования метода интегрирования для экспонентов на
отрезке от 0 до 1
        System.out.println("Задание 1:");
        Function expFunction = new Exp();

        double theoreticValue = Math.E - 1;
        double integralValue = Functions.integrate(expFunction, 0, 1,
0.000001);

        System.out.println("Теоретическое значение: " + theoreticValue);
        System.out.println("Значение полученное при помощи функции: " +
integralValue);
        System.out.println("Шаг = " + 0.000001 + "\n");
```

```
        // Пример использования метода nonThread
        System.out.println("Задание 2");
        nonThread();

        // Пример использования метода simpleThreads
        System.out.println("Задание 3");
        simpleThreads();

        // Пример использования метода complicatedThreads
        System.out.println("Задание 4");
        complicatedThreads();
    }

    public static void nonThread() throws InappropriateFunctionPointException
{
        Task t = new Task(100);
        for (int i = 0; i < t.getNumTasks(); i++) {
            t.setFunction(new Log(1 + (Math.random() * 9)));
            t.setLeftX(Math.random() * 100);
            t.setRightX(Math.random() * 100 + 100);
            t.setStep(Math.random() * 0.9 + 0.1);
            System.out.println("Source leftX = " + t.getLeftX() + " rightX =
" + t.getRightX() + " step = " + t.getStep());
            double res = Functions.integrate(t.getFunction(), t.getLeftX(),
t.getRightX(), t.getStep());
            System.out.println("Result leftX = " + t.getLeftX() + " rightX =
" + t.getRightX() + " step = " + t.getStep() + " integrate = " + res);
        }
    }

    public static void simpleThreads() {
        Task task = new Task(100);

        Thread generatorThread = new Thread(new SimpleGenerator(task));
        generatorThread.start();

        Thread integratorThread = new Thread(new SimpleIntegrator(task));
        integratorThread.start();
    }

    public static void complicatedThreads() throws InterruptedException {
        Task task = new Task(100);
        Semaphore semaphore = new Semaphore(1); // Используем семафор с одним
разрешением

        Thread generatorThread = new Generator(task, semaphore);
        generatorThread.start();

        Thread integratorThread = new Integrator(task, semaphore);
        integratorThread.start();

        Thread.sleep(500); // Ждем 50 миллисекунд

        generatorThread.interrupt();
        integratorThread.interrupt();
    }
}
```

Вывод Main:
Задание 1:
Теоретическое значение: 1.718281828459045
Значение полученное при помощи функции: 1.7182818284378987
Шаг = 1.0E-6

Задание 2
Source leftX = 32.00572422249254 rightX = 167.3059484076776 step = 0.3518467085266558
Result leftX = 32.00572422249254 rightX = 167.3059484076776 step = 0.3518467085266558 integrate = 319.3366591419455
Source leftX = 77.24056672242506 rightX = 124.34089535600702 step = 0.15926535517038368
Result leftX = 77.24056672242506 rightX = 124.34089535600702 step = 0.15926535517038368 integrate = 155.05789563083115
Source leftX = 20.343391957349343 rightX = 108.68983786857537 step = 0.3200654990717219
Result leftX = 20.343391957349343 rightX = 108.68983786857537 step = 0.3200654990717219 integrate = 179.07594857664753
Source leftX = 33.6039441736182 rightX = 124.48384293869901 step = 0.9527173551251192
Result leftX = 33.6039441736182 rightX = 124.48384293869901 step = 0.9527173551251192 integrate = 247.32573177688064
Source leftX = 20.650413489731434 rightX = 101.2207541376989 step = 0.23451052740872722
Result leftX = 20.650413489731434 rightX = 101.2207541376989 step = 0.23451052740872722 integrate = 179.4180443157909
Source leftX = 53.20521670311156 rightX = 104.48980126092297 step = 0.3448878168970466
Result leftX = 53.20521670311156 rightX = 104.48980126092297 step = 0.3448878168970466 integrate = 106.48132545269262
Source leftX = 83.71639880364079 rightX = 181.82745086873902 step = 0.7015084507596379
Result leftX = 83.71639880364079 rightX = 181.82745086873902 step = 0.7015084507596379 integrate = 566.6105154399896
Source leftX = 84.86343904349536 rightX = 130.41999265232334 step = 0.5060497042163014
Result leftX = 84.86343904349536 rightX = 130.41999265232334 step = 0.5060497042163014 integrate = 151.84704710687168

Source leftX = 89.52795600539405 rightX = 148.3606255979215 step = 0.3472145234544104
Result leftX = 89.52795600539405 rightX = 148.3606255979215 step = 0.3472145234544104 integrate = 251.67070061267492
Source leftX = 51.72298714925204 rightX = 138.60802116602306 step = 0.6299350811558729
Result leftX = 51.72298714925204 rightX = 138.60802116602306 step = 0.6299350811558729 integrate = 206.1987894170534
Задание 3
[Simple] Source leftX = 44.50782025970753 rightX = 170.78965255633307 step = 0.44675424903320504
[Simple] Result leftX = 44.50782025970753 rightX = 170.78965255633307 step = 0.44675424903320504 Result: 526.527148292715
[Simple] Source leftX = 8.498795040874796 rightX = 181.21473399994608 step = 0.30715992385325386
[Simple] Result leftX = 8.498795040874796 rightX = 181.21473399994608 step = 0.30715992385325386 Result: 1425.3068756325624
[Simple] Source leftX = 8.889246242467019 rightX = 149.7961501697877 step = 0.9285619809902113
[Simple] Result leftX = 8.889246242467019 rightX = 149.7961501697877 step = 0.9285619809902113 Result: 477128.774222821
[Simple] Source leftX = 11.333241442403075 rightX = 148.92832997322336 step = 0.7911365308456396
[Simple] Result leftX = 11.333241442403075 rightX = 148.92832997322336 step = 0.7911365308456396 Result: 565.130069185635
[Simple] Source leftX = 92.61408727974356 rightX = 193.84076326963265 step = 0.2854222077390412
[Simple] Result leftX = 92.61408727974356 rightX = 193.84076326963265 step = 0.2854222077390412 Result: 721.3503244854152
[Simple] Source leftX = 12.12405491433417 rightX = 119.29138618839286 step = 0.31731316672480814
[Simple] Result leftX = 12.12405491433417 rightX = 119.29138618839286 step = 0.31731316672480814 Result: 342.5646535862333
[Simple] Source leftX = 93.40189363860001 rightX = 116.78011460034841 step = 0.6870504074907096
[Simple] Result leftX = 93.40189363860001 rightX = 116.78011460034841 step = 0.6870504074907096 Result: 225.5455919675212
[Simple] Source leftX = 58.003805640491166 rightX = 191.69453588192314 step = 0.500734632049701

[Simple] Result leftX = 58.003805640491166 rightX = 191.69453588192314 step = 0.500734632049701 Result: 1275.624947205448
[Simple] Source leftX = 86.65465901024061 rightX = 121.7117020631239 step = 0.4555757562443822
[Simple] Result leftX = 86.65465901024061 rightX = 121.7117020631239 step = 0.4555757562443822 Result: 397.7900184260649
[Simple] Source leftX = 41.610274793072854 rightX = 129.15953149962695 step = 0.19276129078639895
[Simple] Result leftX = 41.610274793072854 rightX = 129.15953149962695 step = 0.19276129078639895 Result: 706.3633452366637
Задание 4
[Complicated] Source leftX = 48.06264077335381 rightX = 144.66073366098635 step = 0.8154231758985427
[Complicated] Result leftX = 48.06264077335381 rightX = 144.66073366098635 step = 0.8154231758985427 Result: 792.0028878810027
[Complicated] Source leftX = 14.92282841160213 rightX = 112.31886171887733 step = 0.2950385895119515
[Complicated] Result leftX = 14.92282841160213 rightX = 112.31886171887733 step = 0.2950385895119515 Result: 363.0226783716565
[Complicated] Source leftX = 65.78540045347862 rightX = 128.66148511016087 step = 0.24221989823874684
[Complicated] Result leftX = 65.78540045347862 rightX = 128.66148511016087 step = 0.24221989823874684 Result: 359.14632665949324
[Complicated] Source leftX = 15.819606224253846 rightX = 176.4031104484714 step = 0.9288840714172336
[Complicated] Result leftX = 15.819606224253846 rightX = 176.4031104484714 step = 0.9288840714172336 Result: 605.602768886072
[Complicated] Source leftX = 48.3989832223078 rightX = 161.0630932380345 step = 0.8879521289291368
[Complicated] Result leftX = 48.3989832223078 rightX = 161.0630932380345 step = 0.8879521289291368 Result: 4818.833964122993
[Complicated] Source leftX = 20.812067679381475 rightX = 153.83870498988975 step = 0.9420976399887516
[Complicated] Result leftX = 20.812067679381475 rightX = 153.83870498988975 step = 0.9420976399887516 Result: 1347.7602404357744
[Complicated] Source leftX = 50.049986485429635 rightX = 150.0660958619441 step = 0.3078543773934297
[Complicated] Result leftX = 50.049986485429635 rightX = 150.0660958619441 step = 0.3078543773934297 Result: 542.8110740315444

[Complicated] Source leftX = 89.66483878647186 rightX = 196.02374459180007 step = 0.9662624907143279
[Complicated] Result leftX = 89.66483878647186 rightX = 196.02374459180007 step = 0.9662624907143279 Result: 519.5327646230361
[Complicated] Source leftX = 33.614791576515614 rightX = 113.4886711232206 step = 0.4420617202149708
[Complicated] Result leftX = 33.614791576515614 rightX = 113.4886711232206 step = 0.4420617202149708 Result: 422.53144905197746
[Complicated] Source leftX = 62.992678591722026 rightX = 177.76747752654447 step = 0.9350639323189401
[Complicated] Result leftX = 62.992678591722026 rightX = 177.76747752654447 step = 0.9350639323189401 Result: 11296.755031489593