Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»

Институт информатики и кибернетики

Отчет по лабораторной работе №6

Дисциплина: «ООП»

Тема «Многопоточное приложение»

Выполнил: Пантелеев Ю.В.

Группа: 6201-120303

Самара, 2025

Задание на лабораторную работу

Задание 1

Я добавил в Functions метод, возвращающий значение интеграла функции, вычисленное с помощью численного метода.

Метод integrate:

```
public static double integrate (Function function, double leftX, double rightX, double step){ 7 usages  new *
    if (function.getLeftDomainBorder() - leftX > -1e-10||function.getRightDomainBorder() - rightX < 1e-10){
        throw new IllegalArgumentException("Интервал интегрирования выходит за границы области определения функции");
    }
    if (leftX - rightX > - 1e-10){
        throw new IllegalArgumentException("Левая граница диапазона больше или равна правой");
    }

    if(step<=0){
        throw new IllegalArgumentException("Шаг дискретизации должен быть больше нуля");
    }
    double integral = 0.0;
    double current = leftX;

    while (current < rightX) {
        double next = Math.min(current + step, rightX);
        double f_current = function.getFunctionValue(current);
        double f_next = function.getFunctionValue(next);

        double trapezoidArea = (f_current + f_next) * (next - current) / 2.0;
        integral += trapezoidArea;

        current = next;
    }

    return integral;
}
```

Далее в классе main я проверил этот метод.

Проверка:

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

public class Main {  Юрий *
    public static void main(String[] args) throws Exception {  Юрий *

        System.out.println("1. Проверка метода integrate: ");
        System.out.printf("Значение интегралла экспоненты от 0 до 1: %.8f%n",Math.exp(1)-1);
        double integrate = Functions.integrate(new Exp(), leftX: 0, rightX: 1, step: 0.1);
        System.out.printf("Значение интегралла экспоненты от 0 до 1: %.8f для шага 0.1; погрешность - %.8f%n",integrate, integrate - (Math.exp(1)-1));
        integrate = Functions.integrate(new Exp(), leftX: 0, rightX: 1, step: 0.01);
        System.out.printf("Значение интегралла экспоненты от 0 до 1: %.8f для шага 0.01; погрешность - %.8f%n",integrate, integrate - (Math.exp(1)-1));
        integrate = Functions.integrate(new Exp(), leftX: 0, rightX: 1, step: 0.001);
        System.out.printf("Значение интегралла экспоненты от 0 до 1: %.8f для шага 0.001; погрешность - %.8f%n",integrate, integrate - (Math.exp(1)-1));
        integrate = Functions.integrate(new Exp(), leftX: 0, rightX: 1, step: 0.0001);
        System.out.printf("Значение интегралла экспоненты от 0 до 1: %.8f для шага 0.0001; погрешность - %.8f%n",integrate, integrate - (Math.exp(1)-1));
```

Результат:

```
1. Проверка метода integrate:
Значение интегралла экспоненты от 0 до 1: 1,71828183
Значение интегралла экспоненты от 0 до 1: 1,71971349 для шага 0.1; погрешность - 0,00143166
Значение интегралла экспоненты от 0 до 1: 1,71829615 для шага 0.01; погрешность - 0,00001432
Значение интегралла экспоненты от 0 до 1: 1,71828197 для шага 0.001; погрешность - 0,00000014
Значение интегралла экспоненты от 0 до 1: 1,71828183 для шага 0.0001; погрешность - 0,00000000
```

Задание 2

Я создал пакет threads, в котором создал класс Task, объект которого должен хранить ссылку на объект интегрируемой функции.

Task:

```java
package threads;

import functions.*;

public class Task {  14 usages
    private Function function;  2 usages
    private double leftBorder;  2 usages
    private double rightBorder;  2 usages
    private double discretizationStep;  2 usages
    private int tasksCount;  4 usages

    public Task(int tasksCount){  3 usages
        this.tasksCount=tasksCount;
    }

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

    public void setLeftBorder(double left) {  2 usages
        this.leftBorder=left;
    }

    public void setRightBorder(double right) {  2 usag
        this.rightBorder=right;
    }

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

    public void setTasksCount(int i) {  no usages
        this.tasksCount=tasksCount;
    }

    public Function getFunction() {
        return  function;
    }

    public int getTasksCount(){  4 usages
        return tasksCount;
    }
```

```
    public double getLeftBorder() {  2 usages
        return leftBorder;
    }

    public double getRightBorder() {  2 usages
        return rightBorder;
    }

    public double getDiscretizationStep() {
        return discretizationStep;
    }


}
```

Далее я проверил работу этого класса.

Проверка:

```
public static void nonThread(){  1 usage  new *
    Task task = new Task( tasksCount: 100);
    for(int i =0; i < 100; i++){
        double base = 1 + Math.random()*9;
        task.setLeftBorder(Math.random() * 100);
        task.setRightBorder(100 + Math.random() * 100);
        task.setDiscretizationStep(Math.random());

        System.out.printf("Source %.4f %.4f %.4f%n", task.getLeftBorder(), task.getRightBorder(), task.getDiscretizationStep());

        try {
            Log logFunc = new Log(base);
            double result = Functions.integrate(logFunc, task.getLeftBorder(), task.getRightBorder(), task.getDiscretizationStep());

            System.out.printf("Result %.4f %.4f %.4f %.6f%n", task.getLeftBorder(), task.getRightBorder(), task.getDiscretizationStep(), result);
        } catch (Exception e) {
            System.out.println("Ошибка при интегрировании: " + e.getMessage());
        }
    }
}
```

Результат:
(см. Приложение)

Задание 3

Я создал в пакете threads два класса:

SimpleGenerator, в котором должны формироваться задачи и заноситься в полученный объект задания, а также выводиться сообщения в консоль:

```
package threads;

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

public class SimpleGenerator implements Runnable {  1 usage  new *
    private Task task;  7 usages

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

    @Override  new *
    public void run() {
        int tasksCount = task.getTasksCount();

        for (int i = 0; i < tasksCount; i++) {
            double base = 1 + Math.random() * 9;
            double left = Math.random() * 100;
            double right = 100 + Math.random() * 100;
            double step = Math.random();
            Log logFunction = new Log(base);

            synchronized (task){
                task.setFunction(logFunction);
                task.setLeftBorder(left);
                task.setRightBorder(right);
                task.setDiscretizationStep(step);
            }

            System.out.printf("Generator: Source %.4f %.4f %.4f%n", left, right, step);

            try {
                Thread.sleep( millis: 10);
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                break;
            }
        }
        System.out.println("Generator finished.");
    }
}
```

SimpleIntegrator, в котором должны решаться задачи, данные для которых берутся из полученного объекта задания, а также выводиться сообщения в консоль:

```
package threads;

import functions.*;

public class SimpleIntegrator implements Runnable{  1 usage  new *
    private Task task;  9 usages

    public SimpleIntegrator(Task task){  1 usage  new *
        this.task = task;
    }

    @Override  new *
    public void run() {
        int tasksCount = task.getTasksCount();

        for (int i = 0; i < tasksCount; i++) {
            synchronized (task) {
                if (task.getFunction() == null) {
                }
                else {
                    double left = task.getLeftBorder();
                    double right = task.getRightBorder();
                    double step = task.getDiscretizationStep();

                    try {
                        double result = Functions.integrate(task.getFunction(), left, right, step);

                        System.out.printf("Integrator: Result %.4f %.4f %.4f %.6f%n", left, right, step, result);

                    } catch (IllegalArgumentException e) {
                        System.out.printf("Integrator: ERROR - %s%n", e.getMessage());
                    }

                    task.setFunction(null);
                }
            }
                try {
                    Thread.sleep( millis: 15);
                } catch (InterruptedException e) {
                    System.out.println("Integrator interrupted");
                    Thread.currentThread().interrupt();
                    break;
                        break;
                    }


            }
            System.out.println("Integrator finished.");
        }
}
```

Далее я проверил работу этих классов.

Проверка:

```
public static void simpleThreads() {  1 usage  new *
    Task task = new Task( tasksCount: 100);

    Thread generator = new Thread(new SimpleGenerator(task));
    Thread integrator = new Thread(new SimpleIntegrator(task));

    generator.setPriority(Thread.NORM_PRIORITY);
    integrator.setPriority(Thread.NORM_PRIORITY);

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

    try {
        generator.join();
        integrator.join();
    } catch (InterruptedException e) {
        System.out.println("Основной поток прерван при ожидании");
    }
}
```

Результат:
(см. Приложение)

Задание 4

Я создал в пакете threads три класса:

Generator, в котором должны формироваться задачи и заноситься в полученный объект задания, а также выводиться сообщения в консоль:

```
package threads;

import functions.basic.Log;

public class Generator extends Thread {  2 usages
    private Task task;  6 usages
    private Semaphore semaphore;  3 usages

    public Generator(Task task, Semaphore semaphore) {  1 usage
        this.task = task;
        this.semaphore = semaphore;
        this.setName("Generator-Thread");
    }

    @Override
    public void run() {
        int tasksCount = task.getTasksCount();

        try {
            for (int i = 0; i < tasksCount; i++) {
                if (Thread.interrupted()) {
                    System.out.println(getName() + ": Прерван перед созданием задания");
                    throw new InterruptedException();
                }

                double base = 1 + Math.random() * 9;
                double left = Math.random() * 100;
                double right = 100 + Math.random() * 100;
                double step = Math.random();

                Log logFunction = new Log(base);

                semaphore.beginWrite();

                try {
                    task.setFunction(logFunction);
                    task.setLeftBorder(left);
                    task.setRightBorder(right);
                    task.setDiscretizationStep(step);

                    System.out.printf("%s[%d]: Source %.4f %.4f %.4f%n",
                            getName(), i, left, right, step);
```

```
                }
                finally {
                    semaphore.endWrite();
                }

                try {
                    Thread.sleep( millis: 10);
                } catch (InterruptedException e) {
                    System.out.println(getName() + ": Прерван во время сна");
                    throw e;
                }
            }

            System.out.println(getName() + ": Завершил работу нормально");

        } catch (InterruptedException e) {
            System.out.println(getName() + ": Прерван с исключением");
            Thread.currentThread().interrupt();
        }
    }
}
```

Integrator, в котором должны решаться задачи, данные для которых берутся из полученного объекта задания, а также выводиться сообщения в консоль:

```
package threads;

import functions.Functions;

public class Integrator extends Thread {  2 usages
    private Task task;  8 usages
    private Semaphore semaphore;  3 usages

    public Integrator(Task task, Semaphore semaphore) {  1 usage
        this.task = task;
        this.semaphore = semaphore;
        this.setName("Integrator-Thread");
    }

    @Override
    public void run() {
        int tasksCount = task.getTasksCount();

        try {
            for (int i = 0; i < tasksCount; i++) {
                if (Thread.interrupted()) {
                    System.out.println(getName() + ": Прерван перед обработкой");
                    throw new InterruptedException();
                }

                semaphore.beginRead();

                boolean taskProcessed = false;

                try {
                    if (task.getFunction() != null) {
                        double left = task.getLeftBorder();
                        double right = task.getRightBorder();
                        double step = task.getDiscretizationStep();

                        try {
                            double result = Functions.integrate(task.getFunction(), left, right, step);

                            System.out.printf("%s[%d]: Result %.4f %.4f %.4f %.6f%n", getName(), i, left, right, step, result);

                        } catch (IllegalArgumentException e) {
                            System.out.printf("%s[%d]: ERROR - %s%n", getName(), i, e.getMessage());
```

```
                    }

                    // Очищаем задание
                    task.setFunction(null);
                    taskProcessed = true;
                }

            } finally {
                semaphore.endRead();
            }

            if (taskProcessed) {
                try {
                    Thread.sleep( millis: 15);
                } catch (InterruptedException e) {
                    System.out.println(getName() + ": Прерван после обработки");
                    throw e;
                }
            } else {
                try {
                    Thread.sleep( millis: 5);
                } catch (InterruptedException e) {
                    System.out.println(getName() + ": Прерван во время ожидания");
                    throw e;
                }
            }
        }

        System.out.println(getName() + ": Завершил работу.");

    } catch (InterruptedException e) {
        System.out.println(getName() + ": Прерван с исключением.");
        Thread.currentThread().interrupt(); // Восстанавливаем флаг
    }
  }
}
```

Semaphore, одноместный семафор, различающий операции чтения и записи в защищаемый объект:

```
package threads;

public class Semaphore {  6 usages  new *
    private int readers = 0;  3 usages
    private int writers = 0;  4 usages
    private int writeRequests = 0;  3 usages

    public synchronized void beginRead() throws InterruptedException {
        while (writers > 0 || writeRequests > 0) {
            wait();
        }
        readers++;
    }

    public synchronized void endRead() {  1 usage  new *
        readers--;
    }

    public synchronized void beginWrite() throws InterruptedException {
        writeRequests++;

        while (readers > 0 || writers > 0) {
            wait();
        }

        writeRequests--;
        writers++;
    }

    public synchronized void endWrite() {  1 usage  new *
        writers--;
        notifyAll();
    }
}
```

Далее я проверил работу этих классов.

Проверка:

```java
public static void complicatedThreads() {  1 usage  new *
    Task task = new Task( tasksCount: 100);

    Semaphore semaphore = new Semaphore();

    Generator generator = new Generator(task, semaphore);
    Integrator integrator = new Integrator(task, semaphore);
    generator.setName("Generator");
    integrator.setName("Integrator");

    System.out.println("Запускаем потоки...\n");
    long startTime = System.currentTimeMillis();

    generator.start();
    integrator.start();
    try {
        Thread.sleep( millis: 50);
        System.out.println("Основной поток: прошло 50 мс, прерываем рабочие потоки");
        generator.interrupt();
        integrator.interrupt();

    } catch (InterruptedException e) {
        System.out.println("Основной поток прерван");
    }

    try {
        generator.join( millis: 100);
        integrator.join( millis: 100);

        if (generator.isAlive()) {
            System.out.println("Generator всё ещё жив, принудительно останавливаем...");
        }
        if (integrator.isAlive()) {
            System.out.println("Integrator всё ещё жив, принудительно останавливаем...");
        }

    } catch (InterruptedException e) {
        System.out.println("Ошибка при ожидании завершения потоков");
    }

    long endTime = System.currentTimeMillis();
    System.out.println("РЕЗУЛЬТАТЫ:");
    System.out.printf("Общее время работы: %d мс%n", endTime - startTime);
    System.out.println("Состояние потоков:");
    System.out.println("  Generator: " + (generator.isAlive() ? "жив" : "завершён"));
    System.out.println("  Integrator: " + (integrator.isAlive() ? "жив" : "завершён"));
}
}
```

Результат:

```
4. Проверка классов Semaphore, Generator и Integrator:
Запускаем потоки...

Generator[0]: Source 52,1891 108,0982 0,5648
Integrator[0]: Result 52,1891 108,0982 0,5648 130,291646
Generator[1]: Source 61,7161 136,3682 0,5408
Integrator[1]: Result 61,7161 136,3682 0,5408 186,257218
Generator[2]: Source 36,7362 125,2159 0,6719
Integrator[2]: Result 36,7362 125,2159 0,6719 2734,248457
Generator[3]: Source 76,2921 133,0764 0,3936
Generator[4]: Source 53,4337 127,5103 0,6184
Integrator[3]: Result 53,4337 127,5103 0,6184 193,637158
Основной поток: прошло 50 мс, прерываем рабочие потоки
Generator: Прерван во время сна
Integrator: Прерван после обработки
Generator: Прерван с исключением
Integrator: Прерван с исключением.
РЕЗУЛЬТАТЫ:
Общее время работы: 52 мс
Состояние потоков:
  Generator: завершён
  Integrator: завершён
```

Приложение

Результат 2:

2. Проверка класса Task:

Source 34,5535 177,9206 0,0962

Result 34,5535 177,9206 0,0962 918,531279

Source 81,7445 139,0312 0,3624

Result 81,7445 139,0312 0,3624 149,320333

Source 4,7569 194,1684 0,7258

Result 4,7569 194,1684 0,7258 457,403808

Source 2,9375 132,6074 0,8372

Result 2,9375 132,6074 0,8372 243,677484

Source 34,8518 192,0091 0,6156

Result 34,8518 192,0091 0,6156 2463,496203

Source 92,0823 148,5645 0,2788

Result 92,0823 148,5645 0,2788 211,926117

Source 55,1144 173,1665 0,0173

Result 55,1144 173,1665 0,0173 1735,180789

Source 72,3793 198,5109 0,9116

Result 72,3793 198,5109 0,9116 414,839304

Source 29,3989 109,3106 0,8405

Result 29,3989 109,3106 0,8405 198,900441

Source 3,9836 163,2205 0,8091

Result 3,9836 163,2205 0,8091 583,546621

Source 35,5622 173,6030 0,3369

Result 35,5622 173,6030 0,3369 419,150546

Source 71,9768 136,9251 0,2904

Result 71,9768 136,9251 0,2904 148,481580

Source 95,4551 167,3651 0,8185

Result 95,4551 167,3651 0,8185 475,521537

Source 70,9756 109,1128 0,8536

Result 70,9756 109,1128 0,8536 97,323401

Source 4,8709 132,8865 0,9802

Result 4,8709 132,8865 0,9802 309,023511

Source 49,2678 187,3486 0,3396

Result 49,2678 187,3486 0,3396 300,408111

Source 38,8417 168,3860 0,6392

Result 38,8417 168,3860 0,6392 540,603270

Source 0,3669 189,6323 0,2652

Result 0,3669 189,6323 0,2652 464,688055

Source 32,6734 186,1449 0,9600

Result 32,6734 186,1449 0,9600 312,428685

Source 14,8748 159,8087 0,5184

Result 14,8748 159,8087 0,5184 439,297369

Source 64,1098 179,8243 0,4807

Result 64,1098 179,8243 0,4807 392,189251

Source 30,5452 184,5476 0,8575

Result 30,5452 184,5476 0,8575 849,105368

Source 87,0165 191,9851 0,5070

Result 87,0165 191,9851 0,5070 602,361952

Source 43,4356 115,4067 0,9041

Result 43,4356 115,4067 0,9041 167,253349

Source 91,9751 197,8243 0,6903

Result 91,9751 197,8243 0,6903 287,871824

Source 68,0414 124,4953 0,5729

Result 68,0414 124,4953 0,5729 475,218028

Source 94,1787 146,1438 0,8563

Result 94,1787 146,1438 0,8563 117,799130

Source 59,3000 147,2532 0,1282

Result 59,3000 147,2532 0,1282 179,924263

Source 74,0184 123,5688 0,4008

Result 74,0184 123,5688 0,4008 104,774476

Source 34,3583 190,8606 0,3105

Result 34,3583 190,8606 0,3105 315,971791

Source 20,7531 133,0283 0,8186

Result 20,7531 133,0283 0,8186 864,953531

Source 56,9914 111,7948 0,2219

Result 56,9914 111,7948 0,2219 263,433069

Source 54,6511 102,5163 0,1821

Result 54,6511 102,5163 0,1821 96,087399

Source 72,5658 191,0493 0,9146

Result 72,5658 191,0493 0,9146 336,234510

Source 27,9963 165,1483 0,6366

Result 27,9963 165,1483 0,6366 385,082590

Source 6,4350 160,5308 0,8717

Result 6,4350 160,5308 0,8717 1202,777928

Source 30,3473 176,5275 0,2275

Result 30,3473 176,5275 0,2275 299,022241

Source 63,6612 159,1498 0,8230

Result 63,6612 159,1498 0,8230 298,437651

Source 48,4428 160,4387 0,6269

Result 48,4428 160,4387 0,6269 248,468704

Source 92,9507 122,1208 0,5862

Result 92,9507 122,1208 0,5862 86,101055

Source 56,1407 198,8831 0,5755

Result 56,1407 198,8831 0,5755 385,982579

Source 2,6955 131,9863 0,8760

Result 2,6955 131,9863 0,8760 335,547334

Source 34,7848 107,4720 0,1064

Result 34,7848 107,4720 0,1064 192,987496

Source 42,0119 184,8828 0,2716

Result 42,0119 184,8828 0,2716 314,943243

Source 49,3871 162,1924 0,8668

Result 49,3871 162,1924 0,8668 1586,757982

Source 84,6649 150,5616 0,7846

Result 84,6649 150,5616 0,7846 220,624815

Source 31,0180 168,5626 0,1046

Result 31,0180 168,5626 0,1046 363,587832

Source 78,7707 129,6821 0,7182

Result 78,7707 129,6821 0,7182 103,888626

Source 40,7937 189,1919 0,1360

Result 40,7937 189,1919 0,1360 368,436627

Source 50,1681 136,8920 0,7598

Result 50,1681 136,8920 0,7598 184,101459

Source 15,8530 170,7605 0,3604

Result 15,8530 170,7605 0,3604 306,509497

Source 72,3473 163,2107 0,4973

Result 72,3473 163,2107 0,4973 628,459135

Source 22,7124 136,4351 0,1391

Result 22,7124 136,4351 0,1391 264,288322

Source 40,8296 196,2314 0,8018

Result 40,8296 196,2314 0,8018 1157,857747

Source 68,3918 134,7364 0,1046

Result 68,3918 134,7364 0,1046 140,062815

Source 14,8861 101,6572 0,7673

Result 14,8861 101,6572 0,7673 185,458576

Source 44,6997 160,8726 0,7158

Result 44,6997 160,8726 0,7158 538,179339

Source 14,4879 194,6871 0,1489

Result 14,4879 194,6871 0,1489 376,874361

Source 6,0705 191,5572 0,1272

Result 6,0705 191,5572 0,1272 952,465308

Source 85,3772 154,7677 0,4911

Result 85,3772 154,7677 0,4911 178,968245

Source 6,3422 199,3231 0,6868

Result 6,3422 199,3231 0,6868 564,573314

Source 70,1503 162,2256 0,3873

Result 70,1503 162,2256 0,3873 204,957187

Source 44,8676 137,3523 0,6301

Result 44,8676 137,3523 0,6301 223,398773

Source 48,2434 160,2970 0,5626

Result 48,2434 160,2970 0,5626 497,502305

Source 97,3474 110,8344 0,7865

Result 97,3474 110,8344 0,7865 45,238892

Source 32,7119 171,0691 0,7203

Result 32,7119 171,0691 0,7203 369,748024

Source 31,7775 175,8701 0,3644

Result 31,7775 175,8701 0,3644 368,223348

Source 36,6927 118,8503 0,3442

Result 36,6927 118,8503 0,3442 218,182890

Source 34,9001 131,9361 0,1310

Result 34,9001 131,9361 0,1310 202,574385

Source 47,4419 139,4829 0,8981

Result 47,4419 139,4829 0,8981 211,990903

Source 99,5499 198,9256 0,5540

Result 99,5499 198,9256 0,5540 283,207997

Source 70,8367 179,1161 0,1137

Result 70,8367 179,1161 0,1137 244,762716

Source 48,2850 194,7818 0,7208

Result 48,2850 194,7818 0,7208 665,679144

Source 34,2341 110,9531 0,3080

Result 34,2341 110,9531 0,3080 147,330011

Source 93,6949 154,7383 0,1028

Result 93,6949 154,7383 0,1028 143,103384

Source 60,5747 123,4892 0,9223

Result 60,5747 123,4892 0,9223 149,751581

Source 60,7742 158,1828 0,4792

Result 60,7742 158,1828 0,4792 212,529075

Source 51,6242 158,7553 0,7240

Result 51,6242 158,7553 0,7240 1372,705690

Source 21,7539 154,1606 0,5888

Result 21,7539 154,1606 0,5888 253,612390

Source 43,2426 104,0844 0,3617

Result 43,2426 104,0844 0,3617 116,263553

Source 18,7807 101,5316 0,3251

Result 18,7807 101,5316 0,3251 223,875258

Source 20,5714 165,9136 0,2171

Result 20,5714 165,9136 0,2171 345,150447

Source 74,8351 164,7978 0,5415

Result 74,8351 164,7978 0,5415 669,329242

Source 6,0291 179,1411 0,7422

Result 6,0291 179,1411 0,7422 580,441381

Source 18,2776 131,6169 0,7822

Result 18,2776 131,6169 0,7822 229,730870

Source 58,0072 130,5502 0,7259

Result 58,0072 130,5502 0,7259 576,157071

Source 71,4161 143,0171 0,5063

Result 71,4161 143,0171 0,5063 322,748312

Source 43,1554 199,8184 0,1802

Result 43,1554 199,8184 0,1802 1778,196985

Source 78,4825 152,3286 0,6216

Result 78,4825 152,3286 0,6216 214,022944

Source 70,1483 146,6260 0,7868

Result 70,1483 146,6260 0,7868 299,628238

Source 73,4111 153,8486 0,3926

Result 73,4111 153,8486 0,3926 316,651832

Source 97,8224 197,3803 0,0969

Result 97,8224 197,3803 0,0969 283,402527

Source 12,8082 147,2537 0,0415

Result 12,8082 147,2537 0,0415 272,506601

Source 55,5762 110,0712 0,9247

Result 55,5762 110,0712 0,9247 153,665013

Source 24,5041 128,7766 0,6075

Result 24,5041 128,7766 0,6075 363,117243

Source 38,9222 120,7726 0,6260

Result 38,9222 120,7726 0,6260 193,261523

Source 47,0784 125,1626 0,8804

Result 47,0784 125,1626 0,8804 154,164851

Source 68,0821 100,3074 0,2040

Result 68,0821 100,3074 0,2040 85,018094

Source 60,7477 111,4209 0,4756

Result 60,7477 111,4209 0,4756 256,220203

Source 22,5162 174,5895 0,4325

Result 22,5162 174,5895 0,4325 544,571696

Результат 3:

3. Проверка классов SimpleGenerator и SimpleIntegrator:

Generator: Source 35,0092 178,3865 0,3666

Integrator: Result 35,0092 178,3865 0,3666 292,492781

Generator: Source 89,2770 187,1597 0,4394

Integrator: Result 89,2770 187,1597 0,4394 482,029855

Generator: Source 43,4272 141,1093 0,4420

Integrator: Result 43,4272 141,1093 0,4420 2647,949700

Generator: Source 88,0780 107,4311 0,4408

Generator: Source 20,9566 104,0846 0,9020

Integrator: Result 20,9566 104,0846 0,9020 275,059736

Generator: Source 60,9682 113,1590 0,5671

Integrator: Result 60,9682 113,1590 0,5671 1447,790919

Generator: Source 64,1910 124,3411 0,5625

Generator: Source 39,5028 100,6516 0,9736

Integrator: Result 39,5028 100,6516 0,9736 146,804355

Generator: Source 99,9289 138,4029 0,3819

Integrator: Result 99,9289 138,4029 0,3819 114,405453

Generator: Source 27,3009 101,2364 0,6758

Generator: Source 72,4460 144,1399 0,7777

Integrator: Result 72,4460 144,1399 0,7777 213,567787

Generator: Source 60,8813 107,4985 0,8011

Integrator: Result 60,8813 107,4985 0,8011 160,638218

Generator: Source 24,8657 153,5229 0,5493

Generator: Source 36,6654 137,8065 0,1988

Integrator: Result 36,6654 137,8065 0,1988 798,437216

Generator: Source 75,0188 157,2987 0,4726

Integrator: Result 75,0188 157,2987 0,4726 243,569985

Generator: Source 17,0403 192,6413 0,5995

Generator: Source 11,1813 167,6013 0,8844

Integrator: Result 11,1813 167,6013 0,8844 1032,520716

Generator: Source 53,3702 181,2428 0,7649

Integrator: Result 53,3702 181,2428 0,7649 263,224419

Generator: Source 58,0829 129,5143 0,8211

Generator: Source 88,8485 122,0920 0,3203

Integrator: Result 88,8485 122,0920 0,3203 103,463435

Generator: Source 88,5750 117,6001 0,5298

Integrator: Result 88,5750 117,6001 0,5298 68,287510

Generator: Source 29,0806 145,7422 0,2104

Generator: Source 81,3507 158,4127 0,4166

Integrator: Result 81,3507 158,4127 0,4166 183,828893

Generator: Source 93,2877 153,5158 0,9700

Integrator: Result 93,2877 153,5158 0,9700 141,001767

Generator: Source 50,1058 145,0074 0,7503

Generator: Source 84,0229 110,5821 0,2512

Integrator: Result 84,0229 110,5821 0,2512 77,497806

Generator: Source 94,5322 134,4658 0,6864

Integrator: Result 94,5322 134,4658 0,6864 83,038788

Generator: Source 38,2899 176,9323 0,2989

Integrator: Result 38,2899 176,9323 0,2989 958,180871

Generator: Source 48,5341 144,4490 0,4281

Generator: Source 47,9543 104,7661 0,3146

Integrator: Result 47,9543 104,7661 0,3146 242,543621

Generator: Source 26,0583 189,7976 0,5921

Integrator: Result 26,0583 189,7976 0,5921 1403,318171

Generator: Source 62,3399 155,8071 0,9016

Generator: Source 89,9276 125,1904 0,0930

Integrator: Result 89,9276 125,1904 0,0930 79,625791

Generator: Source 10,8983 117,0614 0,0316

Integrator: Result 10,8983 117,0614 0,0316 192,702906

Generator: Source 94,0703 184,8818 0,1153

Generator: Source 8,8440 118,5350 0,2258

Integrator: Result 8,8440 118,5350 0,2258 472,069561

Generator: Source 74,5338 135,5294 0,9879

Integrator: Result 74,5338 135,5294 0,9879 135,909592

Generator: Source 33,6457 181,4271 0,8893

Generator: Source 89,6752 183,1301 0,0278

Integrator: Result 89,6752 183,1301 0,0278 395,197725

Generator: Source 85,2632 150,3127 0,9932

Integrator: Result 85,2632 150,3127 0,9932 187,427162

Generator: Source 34,9179 100,0795 0,6696

Generator: Source 68,0216 187,8323 0,5435

Integrator: Result 68,0216 187,8323 0,5435 251,463509

Generator: Source 82,4658 163,3701 0,4473

Integrator: Result 82,4658 163,3701 0,4473 284,137751

Generator: Source 57,0133 135,4026 0,5291

Generator: Source 21,3658 119,9440 0,6442

Integrator: Result 21,3658 119,9440 0,6442 227,345166

Generator: Source 36,9177 168,4651 0,2911

Integrator: Result 36,9177 168,4651 0,2911 291,499353

Generator: Source 98,3075 120,2155 0,6784

Generator: Source 26,1300 199,7874 0,5026

Integrator: Result 26,1300 199,7874 0,5026 549,684713

Generator: Source 84,5537 185,7803 0,5386

Integrator: Result 84,5537 185,7803 0,5386 314,966524

Generator: Source 24,0899 109,3328 0,2033

Generator: Source 19,3896 102,3251 0,8386

Integrator: Result 19,3896 102,3251 0,8386 183,209535

Generator: Source 44,6994 104,2150 0,5773

Integrator: Result 44,6994 104,2150 0,5773 113,650521

Generator: Source 5,4827 153,7553 0,3058

Generator: Source 23,7223 137,2517 0,4184

Integrator: Result 23,7223 137,2517 0,4184 338,084322

Generator: Source 56,5435 138,5541 0,7694

Integrator: Result 56,5435 138,5541 0,7694 243,840138

Generator: Source 13,0545 137,6344 0,4717

Generator: Source 22,6786 134,3912 0,9668

Integrator: Result 22,6786 134,3912 0,9668 627,860697

Generator: Source 99,7564 132,2286 0,0950

Integrator: Result 99,7564 132,2286 0,0950 206,349649

Generator: Source 43,8252 101,9945 0,7325

Generator: Source 12,7376 161,0216 0,9301

Integrator: Result 12,7376 161,0216 0,9301 308,008910

Generator: Source 6,6586 144,3018 0,3870

Integrator: Result 6,6586 144,3018 0,3870 1159,301073

Generator: Source 4,9856 115,2320 0,5255

Generator: Source 5,3643 161,9074 0,3917

Integrator: Result 5,3643 161,9074 0,3917 332,537660

Generator: Source 24,2275 151,5970 0,1032

Integrator: Result 24,2275 151,5970 0,1032 353,831358

Generator: Source 31,6624 170,8198 0,5871

Generator: Source 9,9162 161,3545 0,9090

Integrator: Result 9,9162 161,3545 0,9090 333,282797

Generator: Source 44,4785 173,5454 0,5789

Integrator: Result 44,4785 173,5454 0,5789 359,024288

Generator: Source 79,4328 197,4326 0,8394

Generator: Source 94,6493 199,5145 0,0904

Integrator: Result 94,6493 199,5145 0,0904 1536,231223

Generator: Source 91,3931 188,3657 0,2856

Integrator: Result 91,3931 188,3657 0,2856 283,371290

Generator: Source 5,0536 155,8650 0,3464

Generator: Source 98,9514 192,4500 0,3860

Integrator: Result 98,9514 192,4500 0,3860 204,206291

Generator: Source 31,5613 154,5996 0,8048

Integrator: Result 31,5613 154,5996 0,8048 250,366853

Generator: Source 30,9915 170,6556 0,1297

Generator: Source 55,1887 107,1310 0,8037

Integrator: Result 55,1887 107,1310 0,8037 379,776292

Generator: Source 64,0943 181,8099 0,0004

Integrator: Result 64,0943 181,8099 0,0004 2087,158306

Generator: Source 80,8517 102,2330 0,9390

Generator: Source 73,1745 128,5689 0,1043

Integrator: Result 73,1745 128,5689 0,1043 168,789874

Generator: Source 43,1081 108,4103 0,2395

Integrator: Result 43,1081 108,4103 0,2395 162,467011

Generator: Source 73,6760 107,6909 0,9717

Generator: Source 44,2785 166,7057 0,4557

Integrator: Result 44,2785 166,7057 0,4557 555,411307

Generator: Source 93,2161 143,1293 0,4749

Integrator: Result 93,2161 143,1293 0,4749 550,633762

Generator: Source 75,3376 164,9570 0,0291

Generator: Source 35,5706 105,4594 0,5525

Integrator: Result 35,5706 105,4594 0,5525 1012,540477

Generator: Source 94,9649 109,5224 0,5580

Integrator: Result 94,9649 109,5224 0,5580 29,711536

Generator: Source 24,8894 146,1552 0,6835

Generator: Source 67,1475 135,4876 0,8266

Integrator: Result 67,1475 135,4876 0,8266 177,098051

Generator: Source 43,0329 100,2312 0,3003

Integrator: Result 43,0329 100,2312 0,3003 126,560308

Generator: Source 82,7239 121,3719 0,7128

Generator: Source 97,9212 135,7658 0,5836

Integrator: Result 97,9212 135,7658 0,5836 108,529621

Generator: Source 15,2603 199,3476 0,5428

Integrator: Result 15,2603 199,3476 0,5428 796,804222

Generator: Source 85,7854 153,8216 0,3301

Generator: Source 86,6497 176,9892 0,7076

Integrator: Result 86,6497 176,9892 0,7076 287,952563

Generator: Source 35,2118 167,2773 0,8343

Integrator: Result 35,2118 167,2773 0,8343 325,365537

Generator: Source 42,8247 111,5679 0,6077

Generator: Source 45,8711 112,0883 0,9760

Integrator: Result 45,8711 112,0883 0,9760 299,742257

Generator: Source 64,9743 102,9068 0,2439

Integrator: Result 64,9743 102,9068 0,2439 84,579585

Generator: Source 59,8204 169,4904 0,9550

Generator: Source 66,6551 195,8124 0,0183

Integrator: Result 66,6551 195,8124 0,0183 483,361888

Generator: Source 28,2719 102,6633 0,3639

Integrator: Result 28,2719 102,6633 0,3639 231,010336

Generator finished.

Integrator finished.