ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ

Ордена Трудового Красного Знамени

федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский Технический Университет Связи и Информатики» (МТУСИ)

Кафедра математической кибернетики и информационных технологий

Лабораторная работа №6

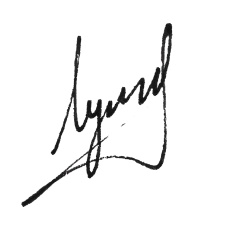
«Многоядерный исследователь/генератор фракталов»

Выполнила:

Студентка 2 курса

Группы БСТ1601

Луканина Елена



Проверил:

Городничев М. Г.

Москва, 2018

Оглавление

[Цель лабораторной работы: 3](#_Toc514185411)

[Задачи: 3](#_Toc514185412)

[Анализ предметной области и выбор инструментария: 3](#_Toc514185413)

[Ход решения задач: 3](#_Toc514185414)

[Вывод: 4](#_Toc514185415)

# Цель лабораторной работы:

Закончить в этой работе с генератором фракталов с еще одной функцией – возможностью нарисовать фрактал с многопоточными фоновыми процессами. Обеспечить независание пользовательского интерфейса во время рисования нового фрактала.

# Задачи:

Внести изменения, используя встроенную поддержку Swing для фоновых потоков. Разбить вычисление на несколько независимых частей для выполнения его несколькими ядрами.

# Анализ предметной области и выбор инструментария:

Я использовала пакет разработчика приложений Java JDK, командная строка и NotePad++.

# Ход решения задач:

• FractalExplorer.java - позволяет исследовать различные части фрактала, создавая и показывая GUI Swing и обрабатывая события, вызванные различным взаимодействием с пользователем.

Я создала подкласс SwingWorker под названием FractalWorker, который является внутренним классом FractalExplorer.

Я создала класс FractalWorker, который отвечает за вычисление значений цвета для одного строки/ряда row фрактала. Конструктор берет y-координату в качестве аргумента и сохраняет ее.

Метод doInBackground () вызывается в фоновом потоке и отвечает за выполнение долгосрочной задачи. Я взяла часть кода из ранней функции «draw fractal» и поместила ее в этот метод. Сохраняем каждое значение RGB в соответствующий элемент целочисленного массива.

Выделяю массив целых чисел в начале этого метода, а затем сохраняю цвет каждого пикселя в этот массив.

Метод done () вызывается, когда фоновая задача завершена, и этот метод вызывается из ядра обработки событий Swing.

Функция void enableUI (boolean val) включает или отключает кнопки и выпадающий список интерфейса на основе указанного значения

В методе done () уменьшаю значение «rows rows» на 1 в качестве последнего шага этой операции. Затем, если после уменьшения оставшихся строк не осталось, вызываю enableUI (true).

Изменяю имплементацию приемника-мышки (mouselistener), чтобы немедленно возвращаться, если значение «rows rows» не равно нулю.

• FractalGenerator.java – предоставленный исходный файл, из которого происходят наши фрактальные генераторы.

• JImageDisplay.java – графический виджет, позволяющий отображать фракталы.

• Mandelbrot.java – производит итерации для построения фрактала. Высчитывает число итераций для данного фрактала: на протяжении 2000 итераций или пока z1 < 4, считает переменную, которая и является числом нужных итераций.

•Tricorn.java – практически идентичная имплементация фракталу Мандельброта, за исключением равенства zn = zn-1 2 + c. Единственное отличие в том, что берется сложное сопряжение zn-1. Каждая итерация начинается с изначальной области определения фрактала tricorn и является от (-2, -2) до (2,2).

•BurningShip – второй фрактал, который имплементирован, это «горящий корабль», который так назван, потому что очень похож на горящий корабль. Вот детали: Равенство zn = (|Re(zn-1)| + i |Im(zn-1)|)2 + c. Другими словами, берется абсолютное значение каждого компонента zn-1. Каждая итерация начинается с изначальной области определения данного фрактала, т.е. от (-2, -2.5) до (2,1.5).

# Вывод:

Эта лабораторная работа позволила изменить уже имеющийся код так, чтобы программа использовала несколько ядер для того, чтобы вычислить фрактал, что позволяет избежать ситуации, когда весь дисплей зависает, в то время как фрактал оттягивается.