**Федеральное государственное общеобразовательное учреждение высшего образования**

**Московский государственный технический университет**

**им. Н. Э. Баумана**

Факультет "Информатика и системы управления"

Кафедра "Информационные системы и телекоммуникации"

**Отчет по лабораторной работе №6**

**"Многозадачность в программе FractalExplorer"**

Студент: Задубин А.А.

Группа: ИУ3-42Б

Преподаватель: Руденкова Ю.С.

Задание:

Надо было создать дочерний класс SwingWorker и дать ему имя FractalWorker. Это класс должен был быть внутренним классом FractalExplorer - private class FractalWorker extends SwingWorker.

Класс FractalWorker отвечает за вычисление цветов точек одной строки фрактала, и для этого ему потребовалось два поля: целая координата y вычисляемой строки, и массив целых чисел (int) для хранения вычисленных RGB значений каждой точки в строке. Конструктор через аргумент получает координату y и сохранять ее.

Метод done() вызывается когда фоновая задача завершена, и этот метод вызывается в контексте потока диспетчера событий Swing.

После завершения прорисовки строки, надо было попросить Swing перерисовать часть экрана, которая была изменена. Так как мы изменили только одну строку воспользуемся вариантом JComponent.repaint() который позволяет задать границы области перерисовки. Метод имеет небольшую странность – у него есть неиспользуемый параметр типа long в начале списка аргументов, и мы просто подставляем ему значение 0. В остальном, просто указываем строку, которую надо перерисовать – позицию (0, y) и размер (displaySize, 1).

Создали функцию void enableUI(boolean val) которая блокирует или разрешает работу кнопок и выпадающего списка в вашем пользовательском интерфейсе в зависимости от значения аргумента. Используем метод setEnabled(boolean) Swing компонента для блокировки/разрешения работы. Необходимо менять состояние кнопки сохраняющей изображение, кнопки сбрасывающей изображение в начальное состояние и выпадающего списка.

В методе done() фонового потока, уменьшаем "количество отавшийся строк" на единицу как завершающий шаг операции. Затем, если количество оставшихся строк становится равным 0, вызываем enableUI(true).

**Код программы:**

BurningShip.java

**public** **class** **BurningShip** **extends** FractalGenerator {

**public** **static** **final** **int** MAX\_ITERATIONS = **500**;

**@Override**

**public** **void** **getInitialRange**(Double range) {}

**@Override**

**public** **int** **numIterations**(**double** x, **double** y) {}

**@Override**

**public** String **toString**() {}

}

FractalExplorer.java

**public** **class** **FractalExplorer** {

**private** **int** side;

**private** **int** strCount;

**private** JButton b;

**private** JButton b2;

**private** JComboBox<FractalGenerator> combo;

**private** JImageDisplay jpgImg;

**private** FractalGenerator geni = **new** BurningShip();

**private** Rectangle2D.Double vsblRange;

**public** **FractalExplorer**(**int** x) {}

**void** **enableUI**(**boolean** val) {}

**class** **ActiveLisner** **implements** ActionListener {

**@Override**

**public** **void** **actionPerformed**(ActionEvent e) {}

**class** **MouseClass** **extends** MouseAdapter {}

**private** **void** **drawFractal**() {}

**private** **void** **createAndShowGUI**() {}

**private** **class** **FractalWorker** **extends** SwingWorker<Object, Object> {

**@Override**

**protected** Object **doInBackground**() **throws** Exception {}

**@Override**

**protected** **void** **done**() {}

}

**public** **static** **void** **main**(String[] args) {}

}

FractalGenerator.java

**public** **abstract** **class** **FractalGenerator** {

**public** **static** **double** **getCoord**(**double** rangeMin, **double** rangeMax,

**int** size, **int** coord) {}

**public** **abstract** **void** **getInitialRange**(Rectangle2D.Double range);

**public** **void** **recenterAndZoomRange**(Rectangle2D.Double range,

**double** centerX, **double** centerY, **double** scale) {}

**public** **abstract** **int** **numIterations**(**double** x, **double** y);

}

JImageDisplay.java

**public** **class** **JImageDisplay** **extends** JComponent {

**private** BufferedImage img;

**public** **JImageDisplay**(**int** width, **int** height) {}

**public** BufferedImage **getImg**() {}

**@Override**

**protected** **void** **paintComponent**(Graphics g) {}

**public** **void** **clearImage**() {}

**public** **void** **drawPixel**(**int** x, **int** y, **int** rgbColor) {}

}

Mandelbrot.java

**public** **class** **Mandelbrot** **extends** FractalGenerator {

**@Override**

**public** **void** **getInitialRange**(Double range) {}

**@Override**

**public** **int** **numIterations**(**double** x, **double** y) {}

**@Override**

**public** String **toString**() {}

}

Tricorn.java

**public** **class** **Tricorn** **extends** FractalGenerator {

**public** **static** **final** **int** MAX\_ITERATIONS = **1000**;

**@Override**

**public** **void** **getInitialRange**(Double range) {}

**@Override**

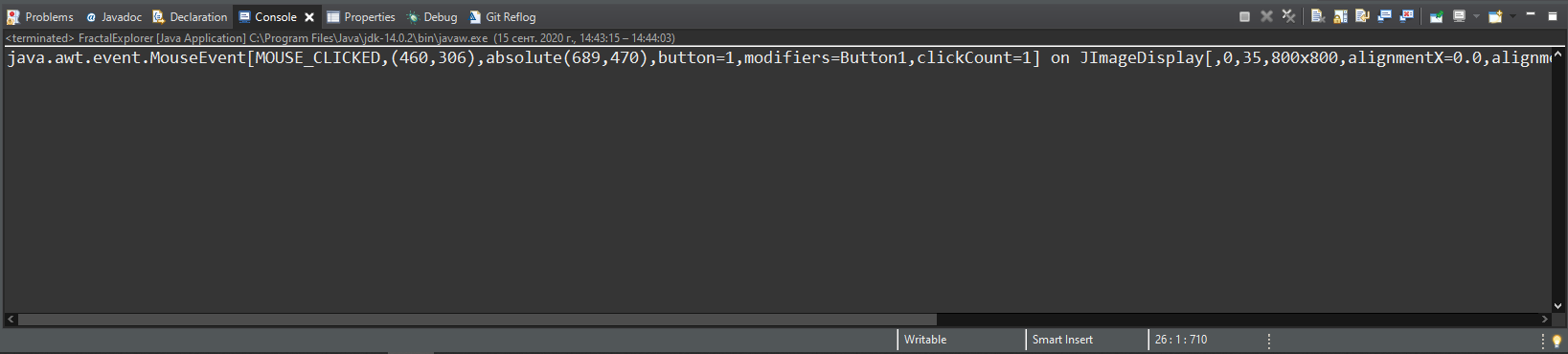
**public** **int** **numIterations**(**double** x, **double** y) {}

**@Override**

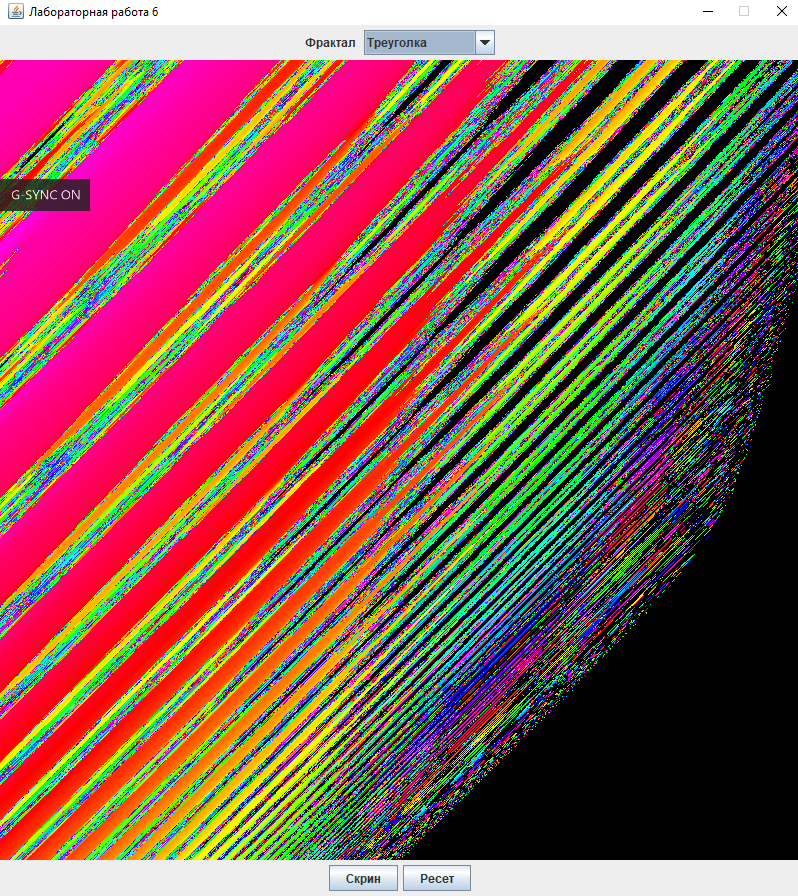
**public** String **toString**() {}

}

**Вывод в консоль:**

****

**Результат работы программы:**



**Вывод**

В ходе лабораторной работы были рассмотрены возможности использования потоков языком Java. Мы модифицировали старую программу, ускорив ее работу и запретив отклик на события во время прорисовки фрактала.