Федеральное агентство связи

Ордена трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и информационных технологий

Отчет по лабораторной работе № 5 «Выбор и сохранение фракталов»

по дисциплине «Объектно-ориентированное программирование»

Выполнил: студент группы БФИ1901

Кириллов Роман Сергеевич

Проверил: Мосева М.С

Москва, 2020

Задача:

• Создать 2 новые реализации FractalGenerator Первым будет фрактал tricorn, который должен находиться в файле Tricorn.java. Для этого нужно создать подкласс FractalGenerator и реализация будет почти идентична фракталу Мандельброта, кроме двух изменений. Вы даже можете скопировать исходный код фрактала Мандельберта и просто внести следующие изменения:

¬ Уравнение имеет вид zn = zn-1 2 + c. Единственное отличие только в том, что используется комплексное сопряжение zn-1 на каждой итерации.

¬ Начальный диапазон для трехцветного фрактала должен быть от (- 2, -2) до (2, 2). Второй фрактал, который необходимо реализовать - это фрактал «Burning Ship», который в реальности не похож на пылающий корабль. Данный фрактал имеет следующие свойства:

¬ Уравнение имеет вид zn = (|Re(zn-1)| + i |Im(zn-1)|)2 + c. Другими словами, вы берете абсолютное значение каждого компонента zn-1 на каждой итерации.

¬ Начальный диапазон для данного фрактала должен быть от (-2, -2.5) до (2, 1.5).

• Сombo-boxe в Swing может управлять коллекцией объектов, но объекты должны предоставлять метод toString(). Убедитесь, что в каждой реализации фракталов tcnm метод toString(), который возвращает имя, например «Mandelbrot», «Tricorn» и «Burning Ship».

• Настроить JComboBox в вашем пользовательском интерфейсе можно с использованием конструктора без параметров, а затем использовать метод addItem(Object) для того, чтобы добавить реализации вашего генератора фракталов. Как указывалось в предыдущем шаге, выпадающий список будет использовать метод toString () в ваших реализациях для отображения генераторов в выпадающем списке.

Следующая ваша задача - сохранение текущего изображения фрактала на диск. Java API предоставляет несколько инструментов для реализации данной задачи.

• Во-первых, вам нужно добавить кнопку «Save Image» в ваше окно. Для этого вы можете добавить обе кнопки «Save Image» и «Reset» в новую Jpanel, а затем разместить эту панель в SOUTH части окна. События от кнопки «Save Image» также должны обрабатываться реализацией ActionListener. Назначьте кнопкам «Save Image» и «Reset» свои значения команд (например, «save» и «reset») для того, чтобы обработчик событий мог отличить события от этих двух разных кнопок.

• В обработчике кнопки «Save Image» вам необходимо реализовать возможность указания пользователем, в какой файл он будет сохранять изображение. Это можно сделать с помощью класса javax.swing.JFileChooser. Указанный класс предоставляет метод showSaveDialog(), который открывает диалоговое окно «Save file», позволяя тем самым пользователю выбрать директорию для сохранения. Метод принимает графический компонент, который является родительским элементом для диалогового окна с выбором файла, что позволяет центрированию окна с выбором относительно его родителя. В качестве родителя используйте окно приложения. Как вы могли заметить, данный метод возвращает значение типа int, которое указывает результат операции выбора файла. Если метод возвращает значение JfileChooser.APPROVE\_OPTION, тогда можно продолжить операцию сохранения файлов, в противном случае, пользователь отменил операцию, поэтому закончите данную обработку события без сохранения. Если пользователь выбрал директорию для сохранения файла, вы можете ее узнать, используя метод getSelectedFile(), который возвращает объект типа File.

• Также необходимо настроить средство выбора файлов, чтобы сохранять изображения только в формате PNG, на данном этапе вы будете работать только с данным форматом. вы сможете это настроить с помощью javax.swing.filechooser.FileNameExtensionFilter, как это продемонстрировано ниже: JFileChooser chooser = new JfileChooser(); FileFilter filter = new FileNameExtensionFilter("PNG Images", "png"); chooser.setFileFilter(filter); chooser.setAcceptAllFileFilterUsed(false); Последняя строка гарантирует, что средство выбора не разрешит пользователю использование отличных от png форматов.

• Если пользователь успешно выбрал файл, следующим шагом является сохранения изображения фрактала на диск! Для данного рода задач Java включает в себя необходимую функциональность. Класс javax.imageio.ImageIO обеспечивает простые операции загрузки и сохранения изображения. Вы можете использовать метод write(RenderedImage im, String formatName, File output). Параметр formatName будет содержать значение «png”. Тип «RenderedImage» - это просто экземпляр BufferedImage из вашего компонента JimageDisplay. (Используйте для него тип доступа public) Метод write() может вызвать исключение, поэтому вам необходимо заключить этот вызов в блок try/catch и обработать возможную ошибку. Блок catch должен проинформировать пользователя об ошибке через диалоговое окно. Swing предоставляет класс javax.swing.JoptionPane для того, чтобы упростить процесс создания информационных диалоговых окон или окон, где нужно выбрать да/нет. Для этого вы можете использовать статический метод JoptionPane.showMessageDialog(Component parent, Object message, String title, int messageType), где messageType у вас будет JOptionPane.ERROR\_MESSAGE. В сообщении об ошибке вы можете использовать возвращаемое значение метода getMessage(), а заголовком окна может быть, например, «Cannot Save Image». Родительским компонентом будет окно для того, чтобы диалоговое окно с сообщением об ошибке выводилось относительно центра окна. После того, как вы закончите реализацию этих функций, запустите. Теперь вы сможете исследовать различные фракталы, а также вы сможете сохранять их на диск. Вы также можете проверить приложение на вывод сообщений об ошибках, попробуйте сохранить изображение в файл, который уже существует, но доступен только для чтения. Или вы можете попробовать сохранить файл с именем, которое является каталогом в целевой папке.

Ход работы:

Класс Mandelbrot

import java.awt.geom.Rectangle2D;  
public class Mandelbrot extends FractalGenerator {  
 public static final int *MAX\_ITERATIONS* = 2000;  
 public void getInitialRange(Rectangle2D.Double range) {  
 range.x = -2;  
 range.y = -1.5;  
 range.width = 3;  
 range.height = 3;  
 }  
 public int numIterations(double x, double y) {  
 double r = 0, i = 0, r1, i1;  
 int k = 0;  
 while (k < *MAX\_ITERATIONS* && (r \* r + i \* i) < 4) {  
 r1 = r \* r - i \* i + x;  
 i1 = 2 \* r \* i + y;  
 r = r1;  
 i = i1;  
 k++;  
 }  
 if (k == *MAX\_ITERATIONS*)  
 return -1;  
 return k;  
 }  
 @Override  
 public String toString() {  
 return "Mandelbrot";  
 }  
}

Класс JImageDisplay

import javax.swing.JComponent;  
import java.awt.Color;  
import java.awt.Dimension;  
import java.awt.Graphics;  
import java.awt.Graphics2D;  
import java.awt.image.BufferedImage;  
public class JImageDisplay extends JComponent {  
 private BufferedImage m;  
 public BufferedImage im() {  
 return m;  
 }  
 public JImageDisplay(int w, int h){  
 m = new BufferedImage(w, h, BufferedImage.*TYPE\_INT\_RGB*);  
 Dimension ty = new Dimension(w, h);  
 super.setPreferredSize(ty);  
 }  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 g.drawImage(m,0,0,m.getWidth(), m.getHeight(), null);  
 }  
 public void clearImage() {  
 Graphics2D itr = m.createGraphics();  
 itr.setColor(Color.*BLACK*);  
 itr.fillRect(0, 0, m.getWidth(), m.getHeight());  
 }  
 public void drawPixel (int x, int y, int rgbColor){  
 m.setRGB(x, y, rgbColor);  
 }  
}

Класс FractalGenerator

import java.awt.geom.Rectangle2D;  
public abstract class FractalGenerator {  
 */\*\*  
 \* This static helper function takes an integer coordinate and converts  
 it  
 \* into a double-precision value corresponding to a specific range. It  
 is  
 \* used to convert pixel coordinates into double-precision values for  
 \* computing fractals, etc.  
 \*  
 \** ***@param*** *rangeMin the minimum value of the floating-point range  
 \** ***@param*** *rangeMax the maximum value of the floating-point range  
 \*  
 \** ***@param*** *size the size of the dimension that the pixel coordinate is  
 from.  
 \* For example, this might be the image width, or the image  
 height.  
 \*  
 \** ***@param*** *coord the coordinate to compute the double-precision value for.  
 \* The coordinate should fall in the range [0, size].  
 \*/* public static double getCoord(double rangeMin, double rangeMax,  
 int size, int coord) {  
 assert size > 0;  
 assert coord >= 0 && coord < size;  
 double range = rangeMax - rangeMin;  
 return rangeMin + (range \* (double) coord / (double) size);  
 }  
 */\*\*  
 \* Sets the specified rectangle to contain the initial range suitable for  
 \* the fractal being generated.  
 \*/* public abstract void getInitialRange(Rectangle2D.Double range);  
 */\*\*  
 \* Updates the current range to be centered at the specified coordinates,  
 \* and to be zoomed in or out by the specified scaling factor.  
 \*/* public void recenterAndZoomRange(Rectangle2D.Double range,  
 double centerX, double centerY, double  
 scale) {  
 double newWidth = range.width \* scale;  
 double newHeight = range.height \* scale;  
 range.x = centerX - newWidth / 2;  
 range.y = centerY - newHeight / 2;  
 range.width = newWidth;  
 range.height = newHeight;  
 }  
 */\*\*  
 \* Given a coordinate <em>x</em> + <em>iy</em> in the complex plane,  
 \* computes and returns the number of iterations before the fractal  
 \* function escapes the bounding area for that point. A point that  
 \* doesn't escape before the iteration limit is reached is indicated  
 \* with a result of -1.  
 \*/* public abstract int numIterations(double x, double y);  
}

FractalExplorer

import javax.imageio.ImageIO;  
import javax.swing.\*;  
import javax.swing.filechooser.FileNameExtensionFilter;  
import java.awt.BorderLayout;  
import java.awt.Color;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import java.awt.geom.Rectangle2D;  
import java.awt.image.BufferedImage;  
import java.io.File;  
import javax.swing.JFileChooser;  
public class FractalExplorer {  
 private int si;  
 private JImageDisplay JDisplay;  
 private FractalGenerator gen;  
 private Rectangle2D.Double d;  
 public FractalExplorer(int size) {  
 si = size;  
 gen = new Mandelbrot();  
 d = new Rectangle2D.Double();  
 gen.getInitialRange(d);  
 JDisplay = new JImageDisplay(si, si);  
 }  
 public void createAndShowGUI() {  
 JDisplay.setLayout(new BorderLayout());  
 JFrame frame = new JFrame("Fractal Explorer");  
 JButton reset = new JButton("Reset Display");  
 reset rez = new reset();  
 reset.addActionListener((ActionListener) rez);  
 JPanel south = new JPanel();  
 FractalGenerator mand = new Mandelbrot();  
 FractalGenerator tric = new Tricorn();  
 FractalGenerator burn = new BurningShip();  
 JPanel north = new JPanel();  
 JComboBox comboBox = new JComboBox();  
 comboBox.addItem(mand);  
 comboBox.addItem(tric);  
 comboBox.addItem(burn);  
 Chooser fractals= new Chooser();  
 comboBox.addActionListener(fractals);  
 JLabel label = new JLabel("Fractal:");  
 north.add(label);  
 north.add(comboBox);  
 JButton save = new JButton("Save");  
 Saver save1 = new Saver();  
 save.addActionListener(save1);  
 south.add(save);  
 south.add(reset);  
 kliker klik = new kliker();  
 JDisplay.addMouseListener(klik);  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.add(north, BorderLayout.*NORTH*);  
 frame.add(south, BorderLayout.*SOUTH*);  
 frame.add(JDisplay, BorderLayout.*CENTER*);  
 frame.pack();  
 frame.setVisible(true);  
 frame.setResizable(false);  
 }  
 private void drawFractal() {  
 double x1, y1;  
 for (int x = 0; x < si; x++) {  
 for (int y = 0; y < si; y++) {  
 x1 = FractalGenerator.*getCoord*(d.x, d.x + d.width, si, x);  
 y1 = FractalGenerator.*getCoord*(d.y, d.y + d.height, si, y);  
 int k = gen.numIterations(x1, y1);  
 if (k == -1) {  
 JDisplay.drawPixel(x, y, 0);  
 } else {  
 float hue = 0.7f + (float) k / 200f;  
 int rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  
 JDisplay.drawPixel(x, y, rgbColor);  
 }  
 }  
 }  
 JDisplay.repaint();  
 }  
 private class Saver implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 if (e.getActionCommand().equals("Save")) {  
 JFileChooser chooser = new JFileChooser();  
 FileNameExtensionFilter filter = new  
 FileNameExtensionFilter("PNG Images", "png");  
 chooser.setFileFilter(filter);  
 chooser.setAcceptAllFileFilterUsed(false);  
 if (chooser.showSaveDialog(JDisplay) ==  
 JFileChooser.*APPROVE\_OPTION*) {  
 java.io.File file = chooser.getSelectedFile();  
 String image = file.getPath();  
 file = new File(image + ".png");  
 try {  
 BufferedImage displayImage = JDisplay.im();  
 ImageIO.*write*(displayImage, "png", file);  
 } catch (Exception exception) {  
 JOptionPane.*showMessageDialog*(JDisplay,  
 exception.getMessage(), "ERROR", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 else return;  
 }  
 }  
 }  
 private class Chooser implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 Object source = e.getSource();  
 if (source instanceof JComboBox) {  
 JComboBox comboBox = (JComboBox) source;  
 gen = (FractalGenerator) comboBox.getSelectedItem();  
 assert gen != null;  
 gen.getInitialRange(d);  
 drawFractal();  
 }  
 }  
 }  
 private class reset implements ActionListener  
 {  
 public void actionPerformed(ActionEvent e)  
 {  
 gen.getInitialRange(d);  
 drawFractal();  
 }  
 }  
 private class kliker extends MouseAdapter  
 {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 int x = e.getX();  
 int y = e.getY();  
 double x1, y1;  
 x1 = FractalGenerator.*getCoord*(d.x, d.x + d.width, si, x);  
 y1 = FractalGenerator.*getCoord*(d.y, d.y + d.height, si, y);  
 gen.recenterAndZoomRange(d, x1, y1, 0.5);  
 drawFractal();  
 }  
 }  
 public static void main(String[] args)  
 {  
 FractalExplorer displayExplorer = new FractalExplorer(700);  
 displayExplorer.createAndShowGUI();  
 displayExplorer.drawFractal();  
 }  
}

Класс BurningShip

import java.awt.geom.Rectangle2D;  
public class BurningShip extends FractalGenerator{  
 public static final int *MAX\_ITERATIONS* = 2000;  
 public void getInitialRange(Rectangle2D.Double range) {  
 range.x = -2;  
 range.y = -2.5;  
 range.width = 4;  
 range.height = 4;  
 }  
 public int numIterations(double x, double y) {  
 double r = 0, i = 0, r1, i1;  
 int k = 0;  
 while (k < *MAX\_ITERATIONS* && (r \* r + i \* i) < 4) {  
 r1 = (r \* r) - (i \* i) + x;  
 i1 = 2 \* Math.*abs*(r\*i) + y;  
 r = r1;  
 i = i1;  
 k++;  
 }  
 if (k == *MAX\_ITERATIONS*)  
 return -1;  
 return k;  
 }  
 @Override  
 public String toString() {  
 return "Burning Ship";  
 }  
}

Класс Tricorn

import java.awt.geom.Rectangle2D;  
public class Tricorn extends FractalGenerator{  
 public static final int *MAX\_ITERATIONS* = 2000;  
 public void getInitialRange(Rectangle2D.Double range) {  
 range.x = -2;  
 range.y = -2;  
 range.width = 4;  
 range.height = 4;  
 }  
 public int numIterations(double x, double y) {  
 double r = 0, i = 0, r1, i1;  
 int k = 0;  
 while (k < *MAX\_ITERATIONS* && (r \* r + i \* i) < 4) {  
 r1 = r \* r - i \* i + x;  
 i1 = - 2 \* r \* i + y;  
 r = r1;  
 i = i1;  
 k++;  
 }  
 if (k == *MAX\_ITERATIONS*)  
 return -1;  
 return k;  
 }  
 @Override  
 public String toString() {  
 return "Tricorn";  
 }  
}

Результат работы:







