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

по дисциплине «Объектно ориентированное программирование»

на тему «Выбор и сохранение фракталов»

Выполнил

Студент группы БФИ1701

Неретин И.Г.

Проверила:

Мосеева М.С.

Москва 2020

1. **Цель работы**

Цель данной лабораторной работы – научиться работать с сохранением изображений, диалоговым меню сохранения файлов и дополнительными компонентами интерфейса фреймворка Swing.

1. **Задание**

Цель работы определила следующие задачи:

1. Разработать классы для отображения двух дополнительных фракталов – «Burning Ship» и «Tricorn»;
2. Реализовать процесс переключения фракталов через элемент combobox;
3. Создать кнопку сохранения и реализовать для нее метод, сохраняющий текущее изображение в виде файла.
4. **Порядок выполнения работы**

Относительно предыдущей работы, классы FractalGenerator и Mandelbrot были переработаны. Оба перестали быть статическими, а последний был вынесен в отдельный класс.

Листинг – BurningShip.java

|  |
| --- |
| import java.awt.geom.Rectangle2D;  */\*\*  \* Созданный класс для фрактала BurningShip  \* \*\*/* public class BurningShip extends FractalGenerator {  public static final int *MAX\_ITERATIONS* = 2000;  public void getInitialRange(Rectangle2D.Double rect){  rect.x = -2;  rect.y = -2.5;  rect.height = 2 - rect.x;  rect.width = 1.5 - rect.y;  }   //Возвращает количество итераций для точки (x, y), при которых очевидно, что точка не принадлежит  // набору. Возвращает -1, если точка находится во множестве Мандельброта  public int numIterations(double x, double y){  int iterations = 0;  Complex c = new Complex(x, y);  Complex z = new Complex(0, 0);  while (iterations < *MAX\_ITERATIONS*){  iterations++;  z = z.AbsImAndReParts().step2().sum(c);  if (z.isMoreThan(2)){  return iterations;  }  }  return -1;  }   @Override  public String toString(){  return "Burning Ship";  } } |

Листинг 2 – Complex.java

|  |
| --- |
| import static java.lang.Math.*abs*;  public class Complex {  public double x;  public double y;  Complex(){  x = 0;  y = 0;  }  Complex (double x, double y){  this.x = x;  this.y = y;  }  public Complex sum(Complex b){  Complex a = new Complex();  a.x = this.x + b.x;  a.y = this.y + b.y;  return a;  }  public Complex step2(){  double buf = x;  x = x\*x - y\*y;  y = 2\*buf\*y;  return this;  }  public Complex step2sopr(){  double buf = x;  x = x\*x - y\*y;  y = -2\*buf\*y;  return this;  }  public Complex AbsImAndReParts(){  x = *abs*(x);  y = *abs*(y);  return this;  }  public boolean isMoreThan(double a){  return (x\*x + y\*y) > a\*a;  } } |

Листинг 3 – FractalExplorer.java

|  |
| --- |
| import javax.swing.\*; import javax.swing.filechooser.FileFilter; import javax.swing.filechooser.FileNameExtensionFilter; import javax.swing.plaf.basic.BasicButtonListener; import javax.swing.plaf.basic.BasicOptionPaneUI; import java.awt.\*; import java.awt.color.ICC\_ProfileRGB; 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.io.File;  public class FractalExplorer {  private int screenSize;  private JImageDisplay display;  private FractalGenerator CurrentFractalGenerator;  private Rectangle2D.Double range;  private JFrame frame;  FractalExplorer(int size){  screenSize = size;  display = new JImageDisplay(screenSize, screenSize);  range = new Rectangle2D.Double();  CurrentFractalGenerator = new Mandelbrot();  CurrentFractalGenerator.getInitialRange(range);  }  public void createAndShowGUI(){  //кнопка сброса  Button reset = new Button("Reset");  reset.setSize(screenSize / 3, 50);  reset.addActionListener(new JButtonClick());  reset.setVisible(true);   //кнопка сохранения фрактала  Button saveImg = new Button("Save");  saveImg.setSize(screenSize / 3, 50);  saveImg.addActionListener(new SaveButtonClick());  saveImg.setVisible(true);   //создание фракталов  BurningShip BurningShipFractal = new BurningShip();  Tricorn TricornFractal = new Tricorn();   //комбобокс выбора фракталов  JComboBox<FractalGenerator> comboBox = new JComboBox<>();  comboBox.addItem(CurrentFractalGenerator);  comboBox.addItem(BurningShipFractal);  comboBox.addItem(TricornFractal);  comboBox.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  FractalGenerator newFractal = (FractalGenerator) comboBox.getSelectedItem();  if (newFractal != null){  CurrentFractalGenerator = newFractal;  newFractal.getInitialRange(range);  drawFractal();  }  }  });   //панель вывода компонентов интерфейса  JPanel panel = new JPanel();  GroupLayout layout = new GroupLayout(panel);  panel.setLayout(layout);   //настройка размещения элементов интерфейса на форме  layout.setHorizontalGroup(layout.createParallelGroup()  .addComponent(display)  .addComponent(comboBox)  .addGroup(layout.createSequentialGroup()  .addComponent(reset)  .addComponent(saveImg))   );  layout.setVerticalGroup(layout.createSequentialGroup()  .addComponent(comboBox)  .addComponent(display)  .addGroup(layout.createParallelGroup()  .addComponent(reset)  .addComponent(saveImg)  )  );   display.clearImage();   //создание и настройка окна  frame = new JFrame();  frame.setContentPane(panel);  frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  frame.setPreferredSize(new Dimension(screenSize, screenSize));  frame.pack();  frame.setVisible(true);  frame.setResizable(false);  frame.addMouseListener(new JMouseAdapter());  this.drawFractal();  }   private void drawFractal(){  int i, j;  double xCoord, yCoord;  int iterations;  float hue;  int rgbColor;  for (i = 0; i < display.getWidth(); i++){  for (j = 0; j < display.getHeight(); j++){  xCoord = CurrentFractalGenerator.getCoord(range.x, range.x + range.width, screenSize, i);  yCoord = CurrentFractalGenerator.getCoord(range.y, range.y + range.height, screenSize, j);  iterations = CurrentFractalGenerator.numIterations(xCoord, yCoord);  if (iterations == -1){  rgbColor = 0;  }  else{  hue = 0.7f + (float) iterations / 200f;  rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  }  display.drawPixel(i, j, rgbColor);  display.repaint();  }  }   }   private class JMouseAdapter extends java.awt.event.MouseAdapter {  @Override  public void mouseClicked(MouseEvent e) {  super.mouseClicked(e);  display.drawPixel(e.getX(), e.getY(), ICC\_ProfileRGB.*icSigGreenColorantTag*);  double xCoord = CurrentFractalGenerator.getCoord(range.x, range.x + range.width, screenSize, e.getX());;  double yCoord = CurrentFractalGenerator.getCoord(range.y, range.y + range.height, screenSize, e.getY());  FractalGenerator.*recenterAndZoomRange*(range, xCoord, yCoord, 0.5);  display.repaint();  drawFractal();  }  }   private class JButtonClick implements ActionListener {  @Override  public void actionPerformed(ActionEvent e) {  CurrentFractalGenerator.getInitialRange(range);  drawFractal();  }  }   private class SaveButtonClick implements ActionListener {  @Override  public void actionPerformed(ActionEvent e) {  JFileChooser saveDlg = new JFileChooser();  if (saveDlg.showSaveDialog(frame) == JFileChooser.*APPROVE\_OPTION*){  File file = saveDlg.getSelectedFile();  FileFilter filter = new FileNameExtensionFilter("PNG Images", "png");  saveDlg.setFileFilter(filter);  saveDlg.setAcceptAllFileFilterUsed(false);  try{  javax.imageio.ImageIO.*write*(display.getBufferedImage(), "png", file);  }  catch (Exception Exc){  JOptionPane.*showMessageDialog*(frame, "Saving image failed. " + Exc.getMessage(), "error", JOptionPane.*ERROR\_MESSAGE*);  }  }  }  } } |

Листинг 4 – FractalGenerator.java

|  |
| --- |
| import java.awt.geom.Rectangle2D;  */\*\*  \* Этот класс предоставляет общий интерфейс и операции для генераторов фракталов,   \* которые можно просматривать в Fractal Explorer.  \*/* public abstract class FractalGenerator {   */\*\*  \* Эта статическая вспомогательная функция принимает целочисленную координату   \* и преобразует ее в значение двойной точности, соответствующее определенному диапазону.   \* Он используется для преобразования координат пикселей в значения   \* двойной точности для вычисления фракталов и т. Д.  \*  \** ***@param*** *rangeMin минимальное значение диапазона с плавающей запятой  \** ***@param*** *rangeMax максимальное значение диапазона с плавающей запятой  \*  \** ***@param*** *size размер измерения, из которого берется пиксельная координата.  \* Например, это может быть ширина изображения или высота изображения.  \*  \** ***@param*** *coord координата, чтобы вычислить значение двойной точности  \* Координата должна находиться в диапазоне [0, размер].  \*/* public 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);  }   */\*\*  \* Устанавливает указанный прямоугольник, чтобы содержать начальный диапазон, подходящий  \* для генерируемого фрактала.  \*/* public void getInitialRange(Rectangle2D.Double range) {};  public int numIterations(double x, double y) { return 0; };   */\*\*  \* Обновляет текущий диапазон с центром в указанных координатах,   \* а также для увеличения или уменьшения с помощью указанного коэффициента масштабирования.  \*/* public static 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;  } } |

Листинг 5 – JImageDisplay.java

|  |
| --- |
| import java.awt.\*; import java.awt.image.BufferedImage;  import static java.awt.image.BufferedImage.*TYPE\_INT\_RGB*;  public class JImageDisplay extends javax.swing.JComponent{  private java.awt.image.BufferedImage bufferedImage;  JImageDisplay(int width, int height){  bufferedImage = new BufferedImage(width, height, *TYPE\_INT\_RGB*);  super.setPreferredSize(new Dimension(width, height));  }   @Override  protected void paintComponent(Graphics g) {  super.paintComponent(g);  g.drawImage (bufferedImage, 0, 0, bufferedImage.getWidth(), bufferedImage.getHeight(), null);  }   public void clearImage(){  int[] rgbData = new int[bufferedImage.getWidth()\*bufferedImage.getHeight()];  //создаем массив нулевых значений, чтобы окрасить все полотно в черный цвет  for (int i=0; i < rgbData.length; i++){  rgbData[i] = 0;  }  bufferedImage.setRGB(0, 0, bufferedImage.getWidth(),  bufferedImage.getHeight(), rgbData, 0, bufferedImage.getWidth());  }   public void drawPixel(int x, int y, int rgbColor){  bufferedImage.setRGB(x, y, rgbColor);  }   public java.awt.image.BufferedImage getBufferedImage(){  return bufferedImage;  } } |

Листинг 6 – Lab5Main.java

|  |
| --- |
| public class Lab5Main {  public static void main(String[] args){  FractalExplorer explorer = new FractalExplorer(500);  explorer.createAndShowGUI();  } } |

Листинг 7 – Mandelbrot.java

|  |
| --- |
| import java.awt.geom.Rectangle2D;  */\*\*  \* Созданный класс для фрактала Мандельброта  \* \*\*/* public class Mandelbrot extends FractalGenerator {  public static final int *MAX\_ITERATIONS* = 2000;  public void getInitialRange(Rectangle2D.Double rect){  rect.x = -2;  rect.y = -1.5;  rect.height = 3 - rect.x;  rect.width = 3 - rect.y;  }   //Возвращает количество итераций для точки (x, y), при которых очевидно, что точка не принадлежит  // набору. Возвращает -1, если точка находится во множестве Мандельброта  public int numIterations(double x, double y){  int iterations = 0;  Complex c = new Complex(x, y);  Complex z = new Complex(0, 0);  while (iterations < *MAX\_ITERATIONS*){  iterations++;  z = z.step2().sum(c);  if (z.isMoreThan(2)){  return iterations;  }  }  return -1;  }   @Override  public String toString(){  return "Mandelbrot";  } } |

Листинг 8 – Tricorn.java

|  |
| --- |
| import java.awt.geom.Rectangle2D;  */\*\*  \* Созданный класс для фрактала Tricorn  \* \*\*/* public class Tricorn extends FractalGenerator {  public static final int *MAX\_ITERATIONS* = 2000;  public void getInitialRange(Rectangle2D.Double rect){  rect.x = -2;  rect.y = -2;  rect.height = 2 - rect.x;  rect.width = 2 - rect.y;  }   //Возвращает количество итераций для точки (x, y), при которых очевидно, что точка не принадлежит  // набору. Возвращает -1, если точка находится во множестве Мандельброта  public int numIterations(double x, double y){  int iterations = 0;  Complex c = new Complex(x, y);  Complex z = new Complex(0, 0);  while (iterations < *MAX\_ITERATIONS*){  iterations++;  z = z.step2sopr().sum(c);  if (z.isMoreThan(2)){  return iterations;  }  }  return -1;  }   @Override  public String toString(){  return "Tricorn";  } } |

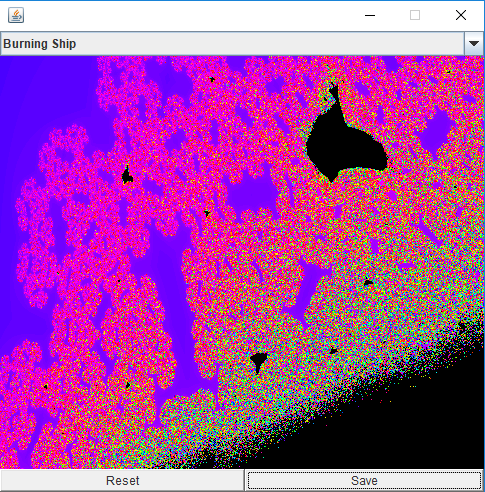


Рисунок – Фрактал «Burning Ship»

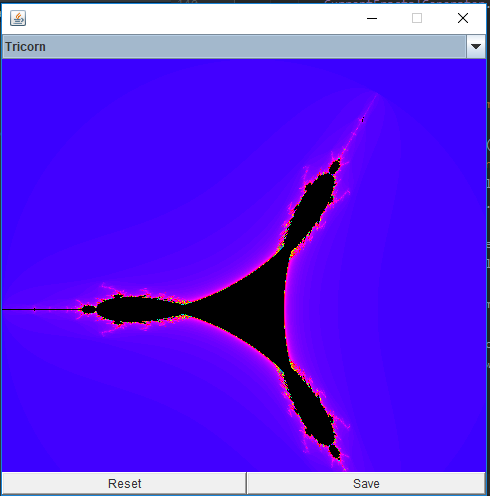


Рисунок – Фрактал «Tricorn»

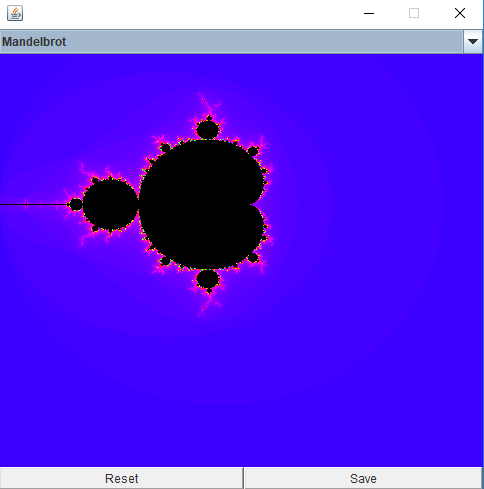


Рисунок – Фрактал «Мандельброт»

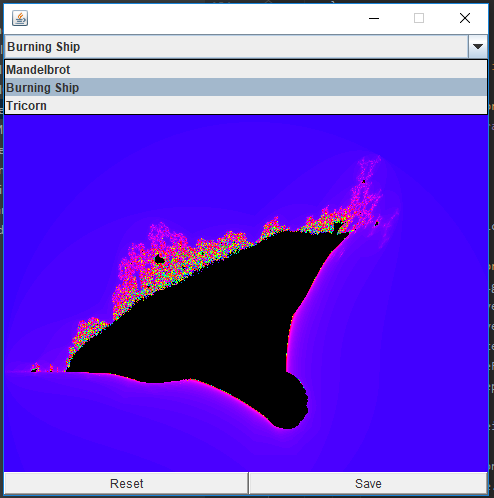


Рисунок – Меню выбора фрактала

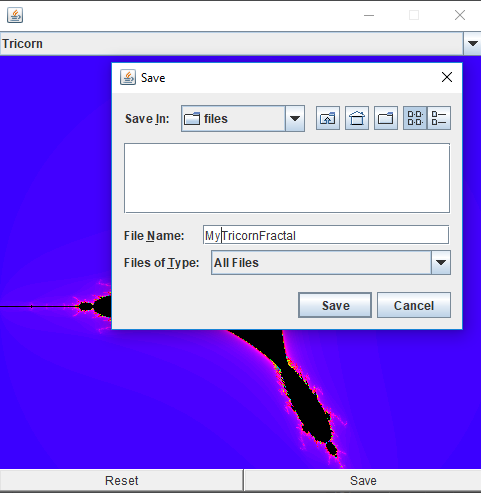


Рисунок – Диалоговое окно сохранения фрактала

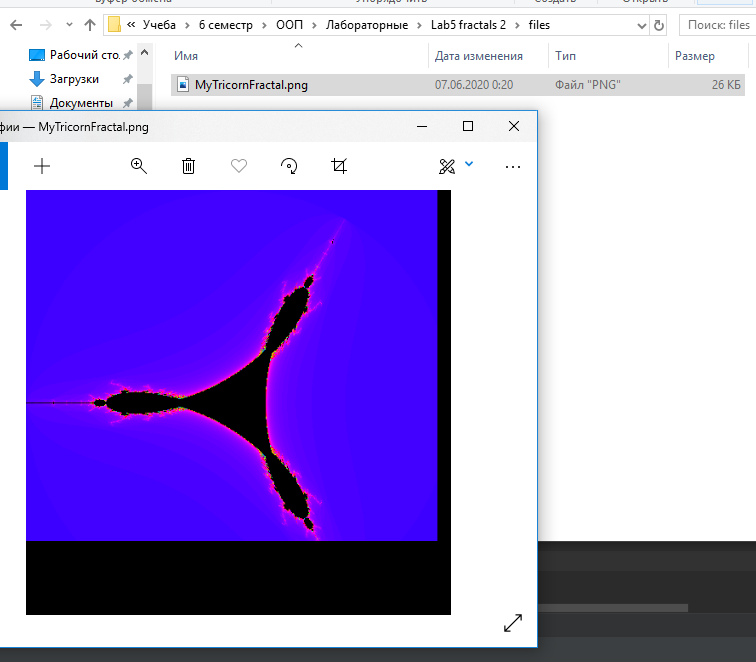


Рисунок – Сохраненный фрактал на диске

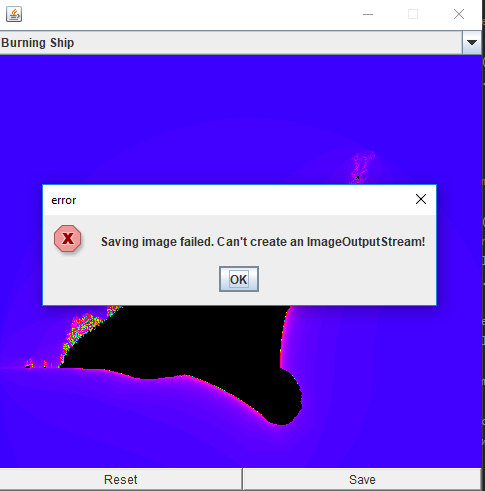


Рисунок – Обработка исключения при попытке сохранить фрактал с некорректным именем («”»)

1. **Заключение**

При помощи фреймворка Swing был разработан улучшенный интерфейс для программы, позволяющий переключать фракталы, а также сохранять их в виде изображений. Таким образом, цель данной лабораторной работы успешно достигнута.

1. **Литература**
2. Камаев В.А., Костерин В.В. Технологии программирования. М.: Высшая школа, 2006;
3. Жоголев Е.А.Технология программирования. – М.: Научный мир, 2004;
4. <https://www.google.ru/>