Лабораторная работа №1

Многопоточные оконные приложения

**Задание 1.**

Написать приложение, которое в одном окне несколькими потоками (каждый поток своим цветом и на своем месте) рисует график функции согласно варианта

Один из способов организации приложения:

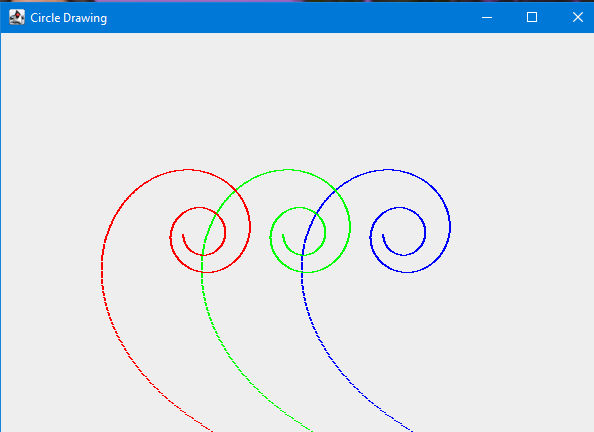
1. Добавить на форму холст (Canvas), передать в класс Picture контекст холста и производить изображение на холсте.
2. Класс Picture сделать, реализующей интерфейс Runnable.
3. Создание изображения переместить из функции paintComponent() в функцию Run().
4. Для запуска потоков создать функцию Start().  
     
   public void start(){  
    new Thread(this).start();  
   }

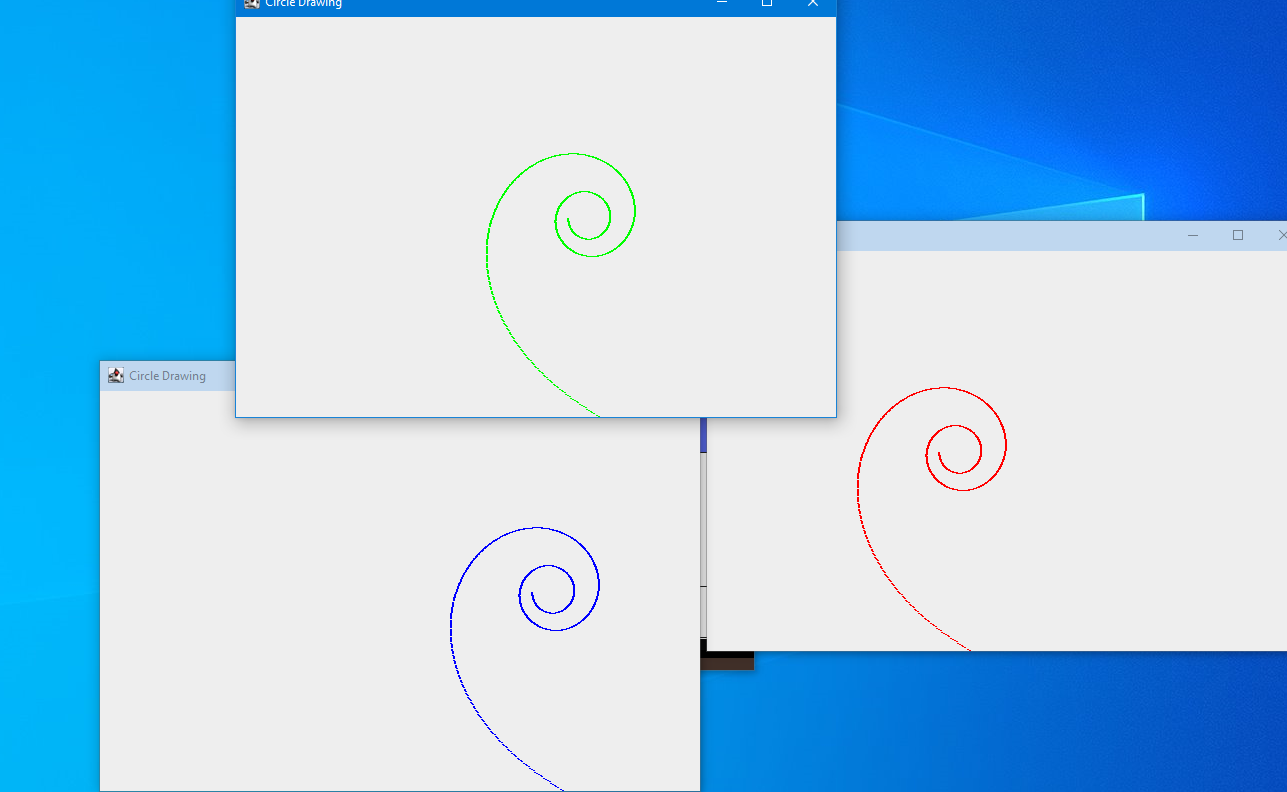
**Задание 2.**

Модифицировать приложение так, чтобы каждый поток создавал новое окноы

**Варианты:**

r= a/φ





import javax.swing.\*;  
import java.awt.\*;  
import java.util.concurrent.ExecutorService;  
import java.util.concurrent.Executors;  
  
import static javax.swing.JFrame.*EXIT\_ON\_CLOSE*;  
class FunDrawing extends JPanel {  
 final int R = 50;  
 final int WIDTH = 600;  
 final int HEIGHT = 400;  
  
  
  
 public FunDrawing() {  
 setPreferredSize(new Dimension(WIDTH, HEIGHT));  
 }  
  
  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 drawFunction(g, Color.*RED*, -2\*R);  
 drawFunction(g, Color.*GREEN*, 0);  
 drawFunction(g, Color.*BLUE*, 2\*R);  
 }  
  
  
 private void drawFunction(Graphics g, Color color, int offsetX) {  
 g.setColor(color);  
 int a=300;  
 for (double phi = 0; phi < 5 \* Math.*PI*; phi += 0.01) {  
 double radius = a / phi;  
 int x = (int) (radius \* Math.*cos*(phi)) + WIDTH / 2 + offsetX;  
 int y = (int) (radius \* Math.*sin*(phi)) + HEIGHT / 2;  
 g.fillOval(x, y, 2, 2);  
 try {  
 Thread.*sleep*(2);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
  
  
 public static void main(String[] args) {  
  
 int c = 2;  
  
 switch (c){  
 case 1:  
 JFrame frame = new JFrame("Circle Drawing");  
 FunDrawing plotter = new FunDrawing();  
 frame.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 frame.add(plotter);  
 frame.pack();  
 frame.setLocationRelativeTo(null);  
 frame.setVisible(true);  
  
 ExecutorService executorService = Executors.*newFixedThreadPool*(3);  
  
 executorService.submit(() -> {  
 plotter.drawFunction(plotter.getGraphics(), Color.*RED*, -plotter.R \* 2);  
 });  
  
 executorService.submit(() -> {  
 plotter.drawFunction(plotter.getGraphics(), Color.*GREEN*, 0);  
 });  
  
 executorService.submit(() -> {  
 plotter.drawFunction(plotter.getGraphics(), Color.*BLUE*, plotter.R \* 2);  
 });  
  
 executorService.shutdown();  
 break;  
  
 case 2:  
 ExecutorService executorService1 = Executors.*newFixedThreadPool*(3);  
  
 executorService1.submit(() -> {  
 JFrame frame1 = new JFrame("Circle Drawing");  
 FunDrawing plotter1 = new FunDrawing();  
 frame1.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 frame1.add(plotter1);  
 frame1.pack();  
 frame1.setLocationRelativeTo(null);  
 frame1.setVisible(true);  
 plotter1.drawFunction(plotter1.getGraphics(), Color.*RED*, -plotter1.R);  
 });  
  
 executorService1.submit(() -> {  
 JFrame frame1 = new JFrame("Circle Drawing");  
 FunDrawing circlePlotter1 = new FunDrawing();  
 frame1.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 frame1.add(circlePlotter1);  
 frame1.pack();  
 frame1.setLocationRelativeTo(null);  
 frame1.setVisible(true);  
 circlePlotter1.drawFunction(circlePlotter1.getGraphics(), Color.*GREEN*, circlePlotter1.R);  
 });  
  
 executorService1.submit(() -> {  
 JFrame frame1 = new JFrame("Circle Drawing");  
 FunDrawing circlePlotter1 = new FunDrawing();  
 frame1.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 frame1.add(circlePlotter1);  
 frame1.pack();  
 frame1.setLocationRelativeTo(null);  
 frame1.setVisible(true);  
 circlePlotter1.drawFunction(circlePlotter1.getGraphics(), Color.*BLUE*, circlePlotter1.R \* 3);  
 });  
  
 executorService1.shutdown();  
 break;  
 default:  
 System.*out*.println("Invalid option. Please enter 1, 2, or 3.");  
 }  
 }  
 };