Круг меняет цвет при каждом нажатии кнопки

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SimpleGui3C implements ActionListener {

    JFrame frame;

    public static void main (String[] args) {
        SimpleGui3C gui = new SimpleGui3C();
        gui.go();
    }

    public void go() {
        frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JButton button = new JButton("Change colors");
        button.addActionListener(this);

        MyDrawPanel drawPanel = new MyDrawPanel();

        frame.getContentPane().add(BorderLayout.SOUTH, button);
        frame.getContentPane().add(BorderLayout.CENTER, drawPanel);
        frame.setSize(300,300);
        frame.setVisible(true);
    }

    public void actionPerformed(ActionEvent event) {
        frame.repaint();
    }
}
```

Пользовательская панель для рисования (экземпляр MyDrawPanel) находится в области CENTER фрейма.

Кнопка в области SOUTH фрейма.

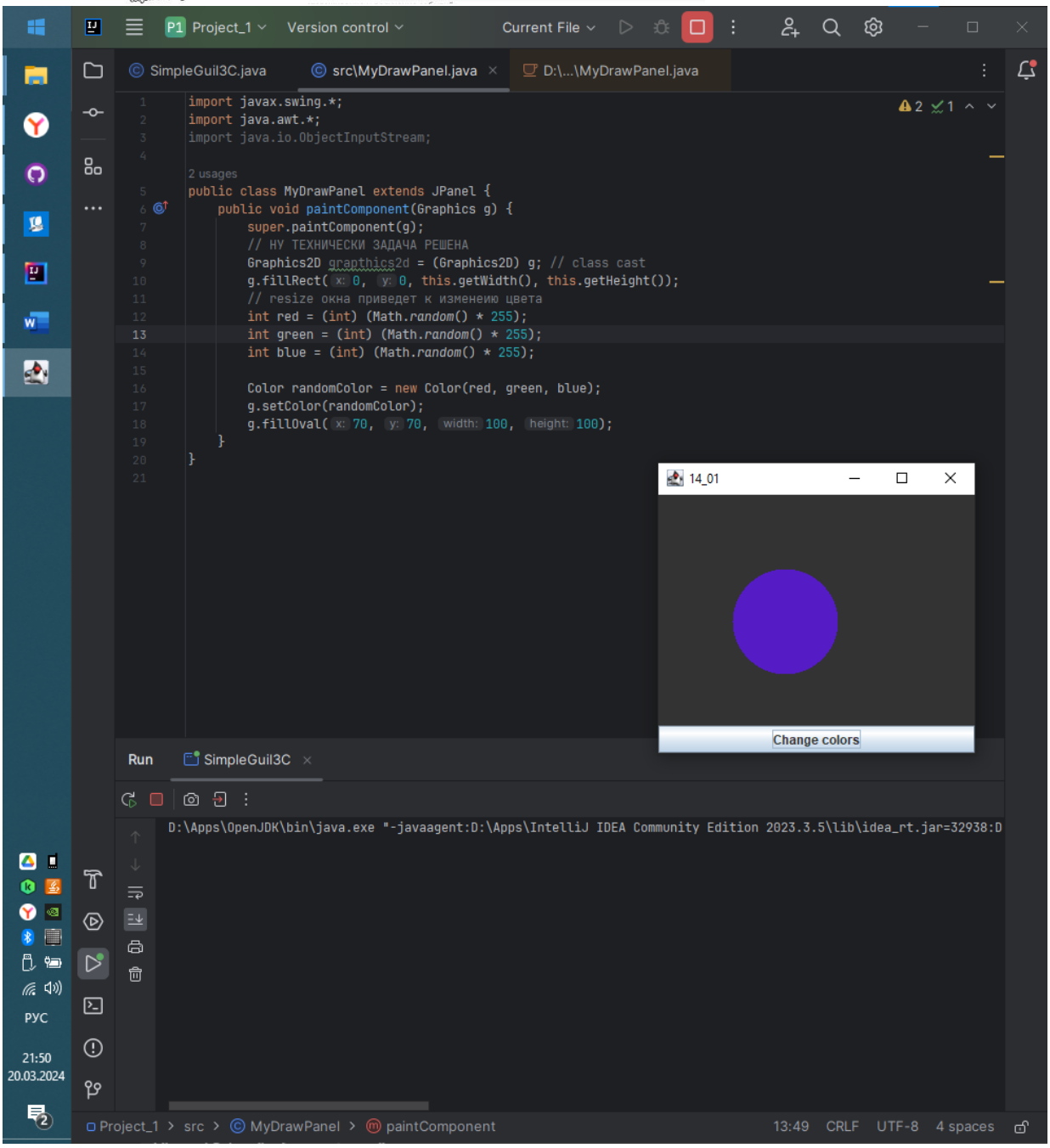
Добавляем слушатель (this) к кнопке.

Добавляем два виджета (кнопку и панель для рисования) в две области фрейма.

Когда пользователь нажимает кнопку, вызываем для фрейма метод repaint(). Это значит, что метод paintComponent() вызывается для каждого виджета во фрейме!

```
class MyDrawPanel extends JPanel {
    public void paintComponent(Graphics g) {
        // Код для заполнения овала произвольным цветом
        // Смотрите его на странице
    }
}
```

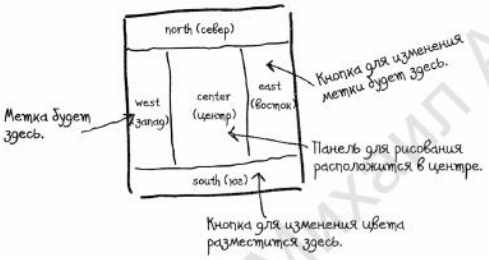
Метод панели для рисования paintComponent() вызывается при каждом нажатии кнопки.



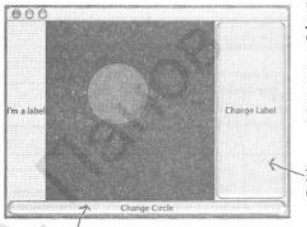
Попробуем сделать это с двумя кнопками

Кнопка в области south будет делать то же, что и сейчас, — вызывать перерисовку фрейма. Вторая кнопка (которую мы поместим в область east) будет менять содержимое метки, которая представляет собой текст на экране.

Теперь нам понадобятся четыре виджета



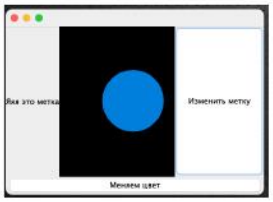
И нам нужно получить два события



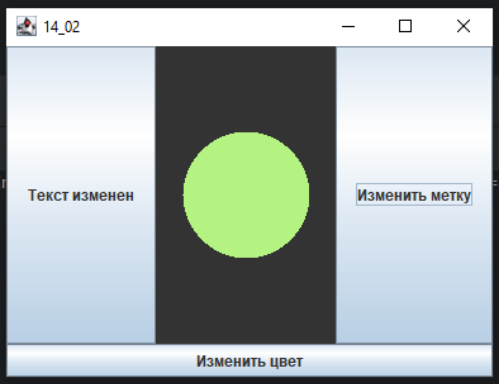
Эта кнопка меняет текст на противоположной стороне.

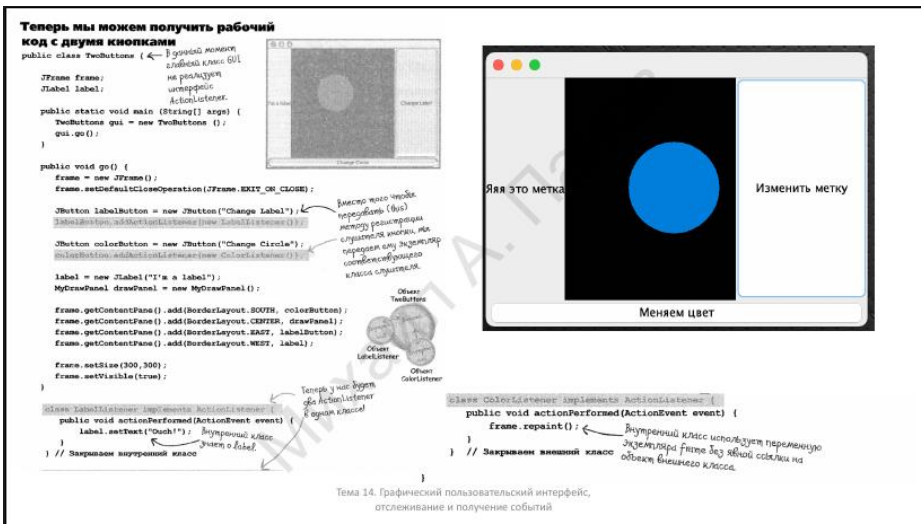
Эта кнопка изменяет цвет круга.

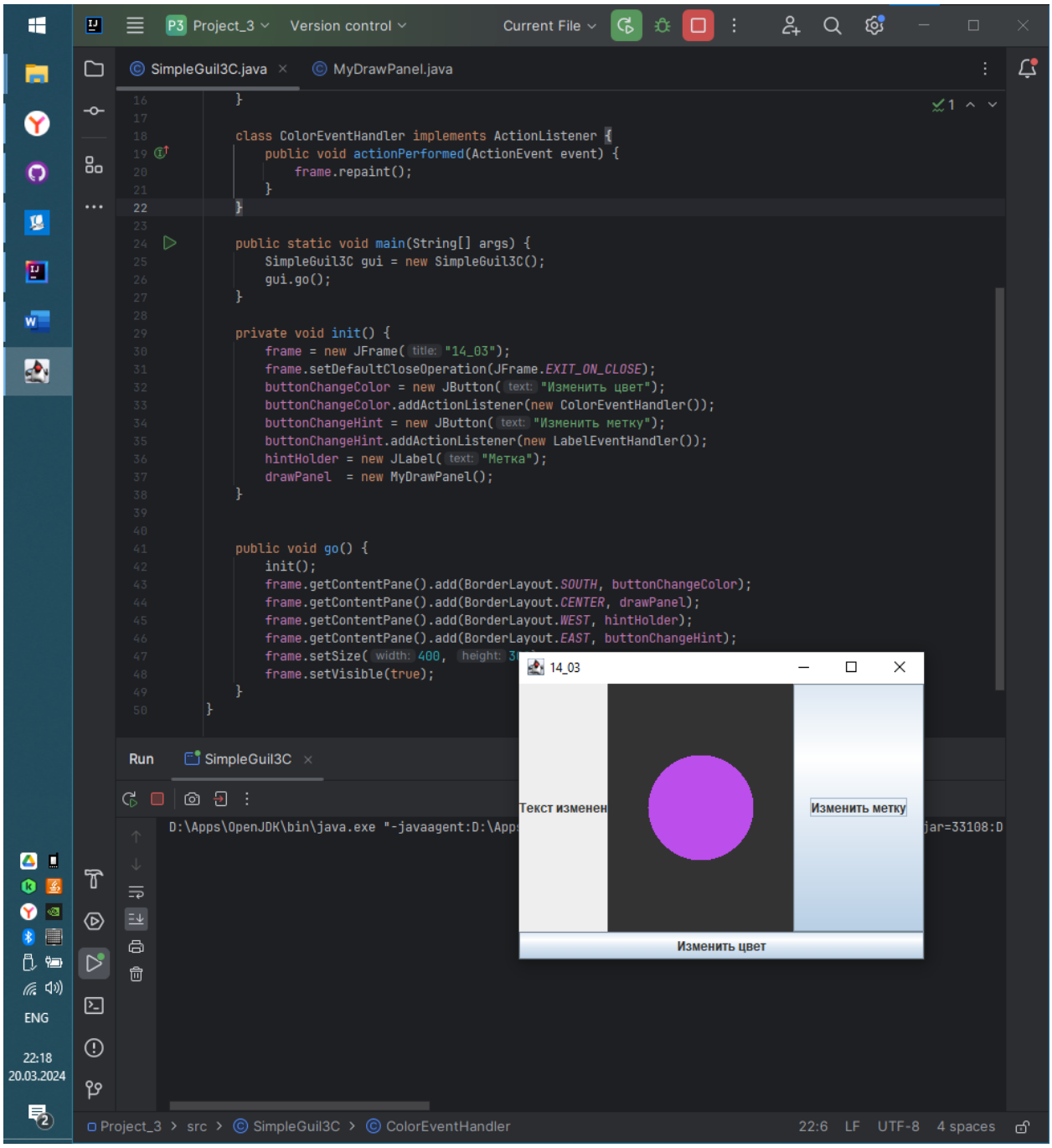
Ой! Разве так можно? Как вы получите *два* события, когда у вас есть только *один* метод actionPerformed()?




```java
    }

    public void go() {
        frame = new JFrame( title: "14_02");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        buttonChangeColor = new JButton( text: "Изменить цвет");
        buttonChangeColor.addActionListener( l: this);

        buttonChangeHint = new JButton( text: "Изменить метку");
        buttonChangeHint.addActionListener( l: this);

        hintHolder = new JButton( text: "Метка");

        MyDrawPanel drawPanel  = new MyDrawPanel();

        frame.getContentPane().add(BorderLayout.SOUTH, buttonChangeColor);
        frame.getContentPane().add(BorderLayout.CENTER, drawPanel);
        frame.getContentPane().add(BorderLayout.WEST, hintHolder);
        frame.getContentPane().add(BorderLayout.EAST, buttonChangeHint);
        frame.setSize( width: 400,  height: 300);
        frame.setVisible(true);
    }

    public void actionPerformed(ActionEvent event) {
        frame.repaint();
        if(event.getSource() == buttonChangeHint) {
            hintHolder.setText("Текст изменен");
        }
    }
}
```




Теперь мы можем получить рабочий код с двумя кнопками
Яяя это метка
Изменить метку
Меняем цвет
Тема 14. Графический пользовательский интерфейс, отслеживание и получение событий


Project_3
Version control
Current File
SimpleGuil3C.java
MyDrawPanel.java
class ColorEventHandler implements ActionListener {
public void actionPerformed(ActionEvent event) {
frame.repaint();
}
}
public static void main(String[] args) {
SimpleGuil3C gui = new SimpleGuil3C();
gui.go();
}
private void init() {
frame = new JFrame(title: "14_03");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
buttonChangeColor = new JButton(text: "Изменить цвет");
buttonChangeColor.addActionListener(new ColorEventHandler());
buttonChangeHint = new JButton(text: "Изменить метку");
buttonChangeHint.addActionListener(new LabelEventHandler());
hintHolder = new JLabel(text: "Метка");
drawPanel = new MyDrawPanel();
}
public void go() {
init();
frame.getContentPane().add(BorderLayout.SOUTH, buttonChangeColor);
frame.getContentPane().add(BorderLayout.CENTER, drawPanel);
frame.getContentPane().add(BorderLayout.WEST, hintHolder);
frame.getContentPane().add(BorderLayout.EAST, buttonChangeHint);
frame.setSize(width: 400, height: 3
frame.setVisible(true);
}
}
14_03
Текст изменен
Изменить метку
Изменить цвет
Run
SimpleGuil3C
D:\Apps\OpenJDK\bin\java.exe "-javaagent:D:\App
jar=33108:D
ENG
22:18
20.03.2024
Project_3 > src > SimpleGuil3C > ColorEventHandler
22:6 LF UTF-8 4 spaces

Используем внутренний класс для анимации

Мы рассказали, почему внутренние классы так удобны для слушателей событий, так как вам придется реализовывать один и тот же метод обработки событий несколько раз. Но теперь мы рассмотрим, насколько полезен внутренний класс при использовании в качестве подкласса, не наследуемого внешним классом. Другими словами, рассмотрим случай, когда внешний и внутренний классы находятся в различных иерархиях наследования.

Наша цель состоит в том, чтобы сделать простую анимацию, где круг преодолевает экран от верхнего левого угла к нижнему правому.

Как работает простая анимация

1. Рисуем объект по определенным координатам x и y:

```
g.fillOval(20,50,100,100);
```

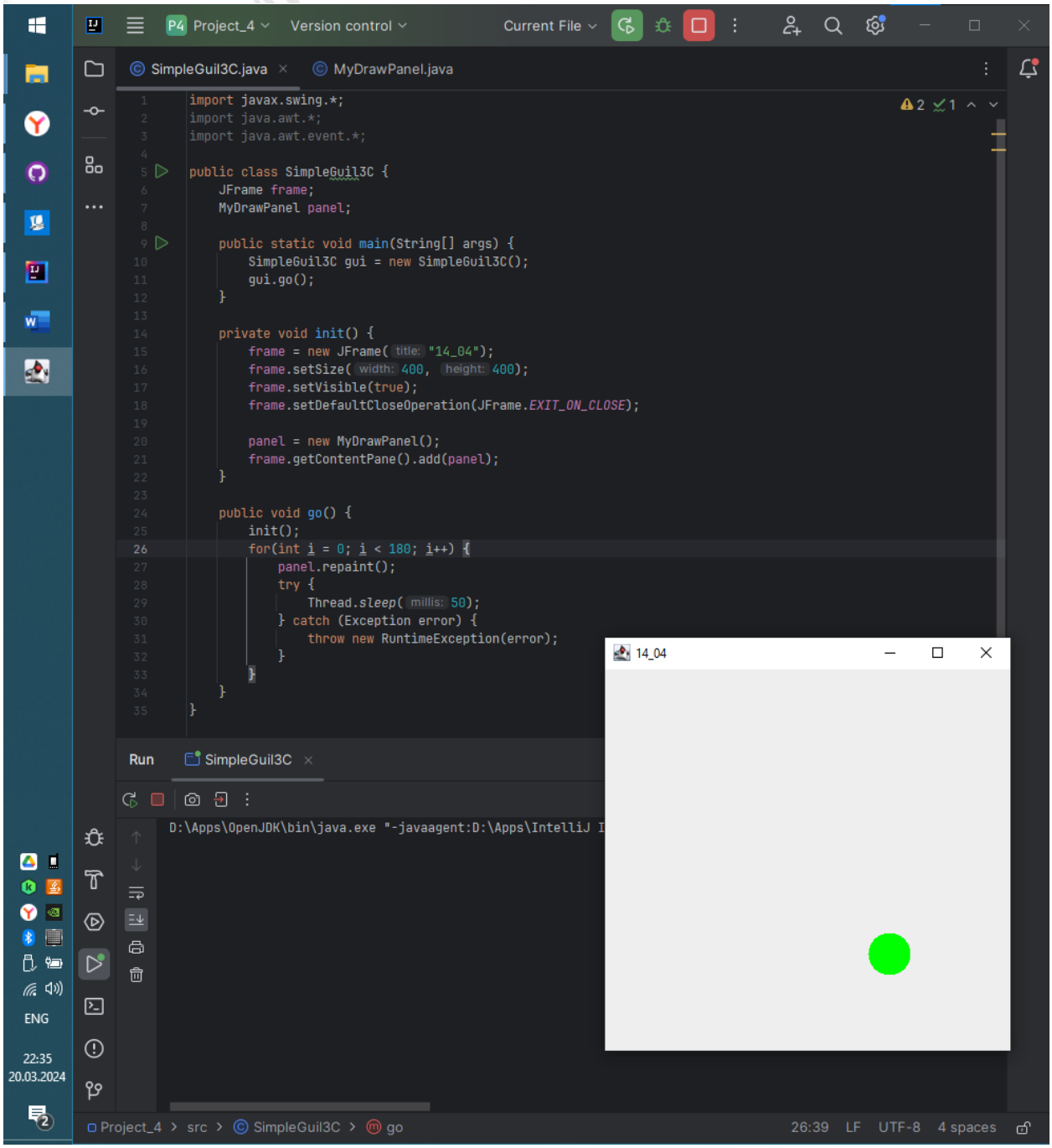
20 пикселов от левого края,
50 пикселов от верхнего края.

2. Рисуем объект по другим координатам x и y:

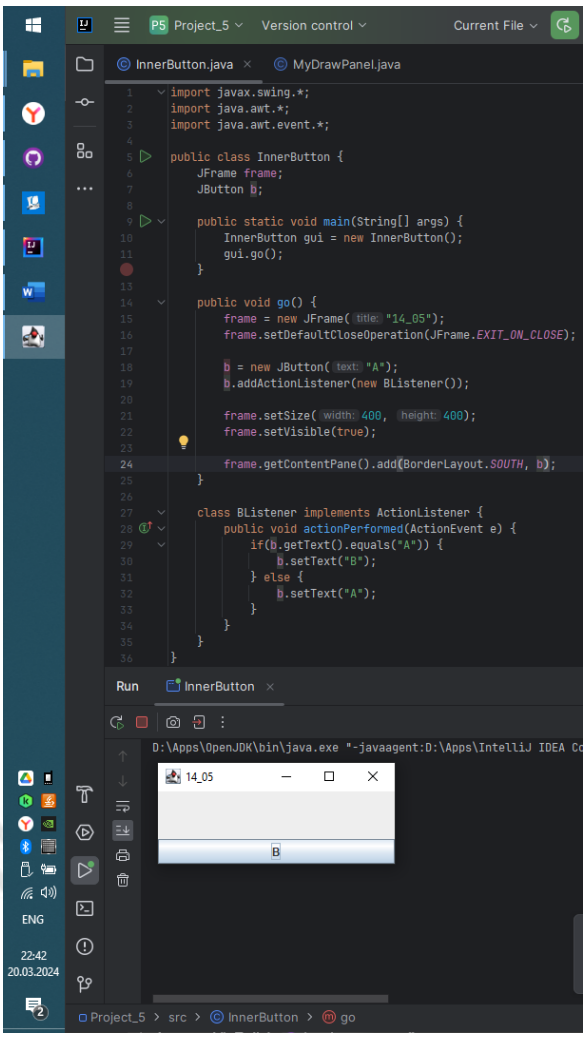
```
g.fillOval(25,55,100,100);
```

25 пикселов от левого края,
55 пикселов от верхнего края (объект переместился немного вниз и вправо).

3. Повторяем предыдущий шаг с изменением значений x и y столько раз, сколько должна длиться анимация.



```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

class InnerButton {

  JFrame frame;
  JButton b;

  public static void main(String [] args) {
    InnerButton gui = new InnerButton();
    gui.go();
  }

  public void go() {
    frame = new JFrame();
    frame.setDefaultCloseOperation(
                  JFrame.EXIT_ON_CLOSE);

    b = new JButton("A");
    b.addActionListener();

    frame.getContentPane().add(
                  BorderLayout.SOUTH, b);
    frame.setSize(200,100);
    frame.setVisible(true);
  }

  class BListener extends ActionListener {
    public void actionPerformed(ActionEvent e) {
      if (b.getText().equals("A")) {
        b.setText("B");
      } else {
        b.setText("A");
      }
    }
  }
}
```



```
import javax.swing.*;
import java.awt.*;
public class Animate {
    int x = 1;
    int y = 1;
    public static void main (String[] args) {
        Animate gui = new Animate ();
        gui.go();
    }
    public void go() {
        JFrame ________ = new JFrame();
        frame.setDefaultCloseOperation(
                        JFrame.EXIT_ON_CLOSE);
        _____________________;
        ________.getContentPane().add(drawP);
        _____________________;
        ________.setVisible(true);
        for (int i=0; i<124; _____________) {
        _____________________;
        _____________________;
        try {
            Thread.sleep(50);
        } catch(Exception ex) { }
        }
    }
    class MyDrawP extends JPanel {
            public void paintComponent (Graph::
                                    ________) {
        _____________________________;
        _____________________________;
        _____________________________;
        _____________________________;
    }
}
```

Головоломка у бассейна

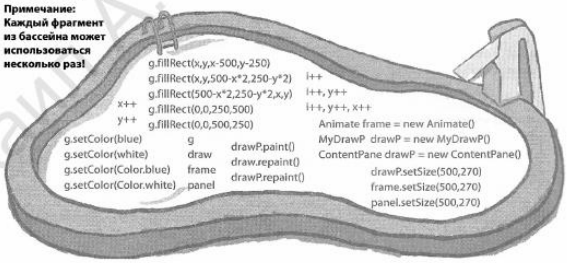
Ваша **задача** — взять фрагменты кода со дна бассейна и поместить их в пустые секции. Вы **можете** использовать один фрагмент несколько раз, и необязательно задействовать все фрагменты. Ваша **цель** — сделать класс, который скомпилируется, запустится и создаст приведенный ниже результат.

Результат

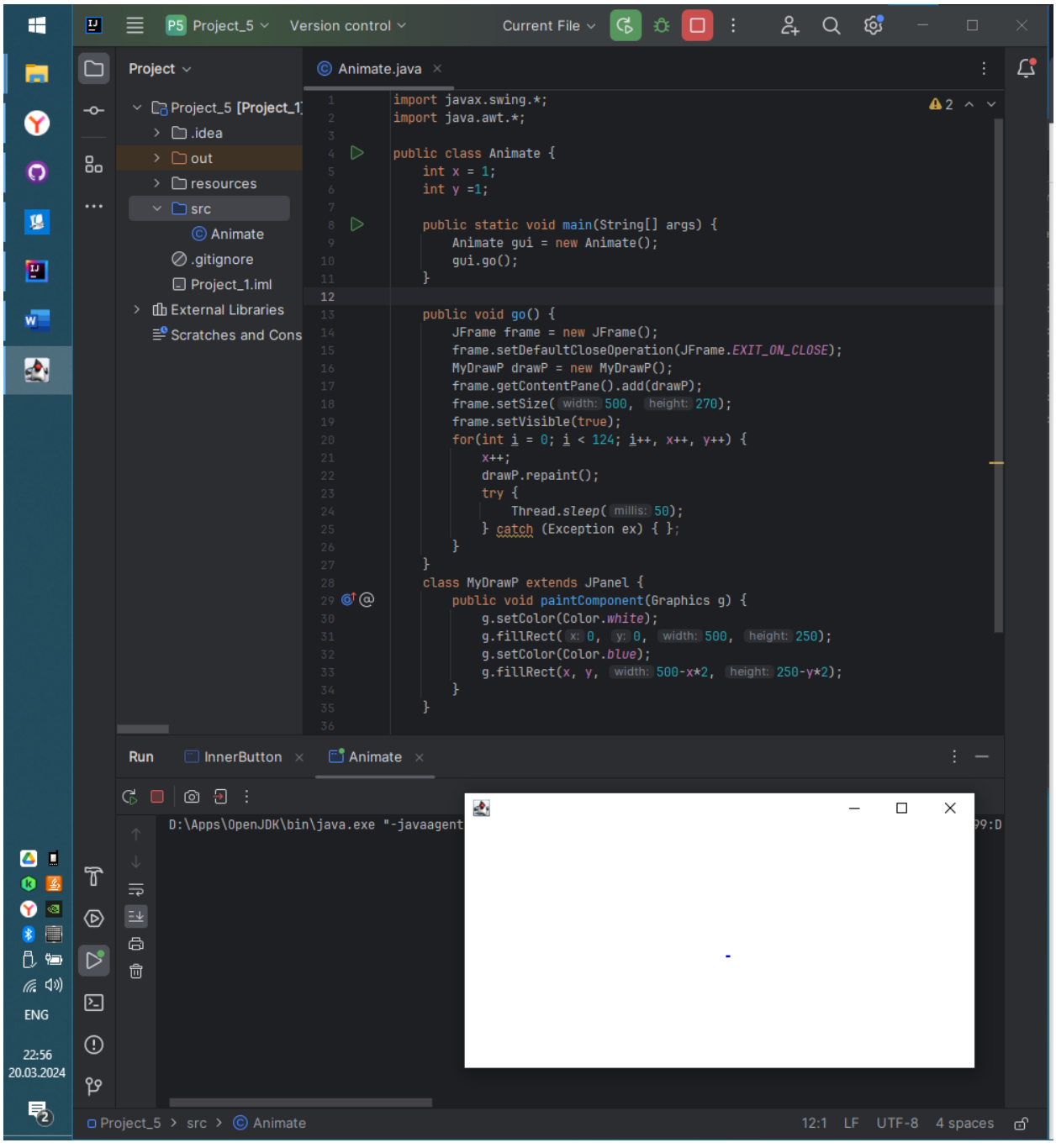
Удивительный сжимающийся синий прямоугольник.

Программа создаст синий прямоугольник, который будет сжиматься, пока не исчезнет на белом полотне.

Примечание: Каждый фрагмент из бассейна может использоваться несколько раз!



Головоломка у бассейн




```
import javax.swing.*;
import java.awt.*;

public class Animate {
    int x = 1;
    int y =1;

    public static void main(String[] args) {
        Animate gui = new Animate();
        gui.go();
    }

    public void go() {
        JFrame frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        MyDrawP drawP = new MyDrawP();
        frame.getContentPane().add(drawP);
        frame.setSize( width: 500,  height: 270);
        frame.setVisible(true);
        for(int i = 0; i < 124; i++, x++, y++) {
            x++;
            drawP.repaint();
            try {
                Thread.sleep( millis: 50);
            } catch (Exception ex) { };
        }
    }
    class MyDrawP extends JPanel {
        public void paintComponent(Graphics g) {
            g.setColor(Color.white);
            g.fillRect( x: 0,  y: 0,  width: 500,  height: 250);
            g.setColor(Color.blue);
            g.fillRect(x, y,  width: 500-x*2,  height: 250-y*2);
        }
    }

```