

### Ваш первый графический интерфейс: кнопка во фрейме

```
import javax.swing.*;

public class SimpleGui {
    public static void main (String[] args) {
        JFrame frame = new JFrame();
        JButton button = new JButton("click me");

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.getContentPane().add(button);

        frame.setSize(300, 300);

        frame.setVisible(true);
    }
}
```

*Импортируем пакет Swing.*

*Создаем фрейм и кнопку.*

*Передаем конструктору кнопки текст, который будет на ней отображаться.*


*Эта строка завершит работу программы при закрытии окна (если вы не добавите ее, то приложение будет висеть на экране вечно).*

*Добавляем кнопку на панель фрейма.*

*Присваиваем фрейму размер (в пикселях).*

*И наконец, делаем фрейм видимым (если вы пропустите этот шаг, то выполнив данный код, ничего не увидите).*

**Посмотрим, что произойдет при запуске:**



*Ова! Это действительно большая кнопка. Она заполняет все доступное пространство фрейма. После вы узнаете, как управлять положением (и размером) кнопки во фрейме.*

**Но когда я нажимаю ее, ничего не происходит...**

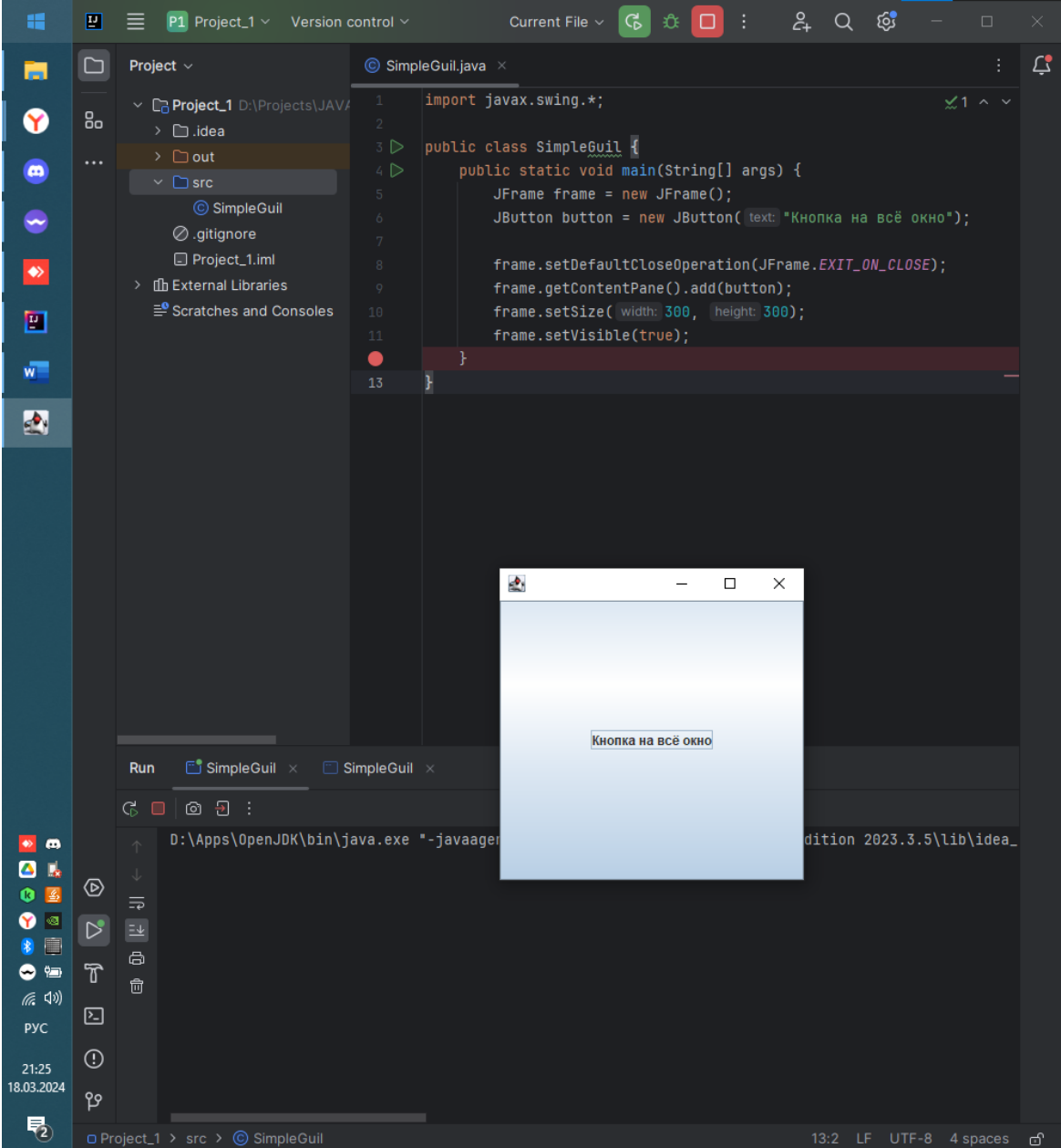
*Это не совсем верно. Когда вы нажимаете кнопку, она меняет свой вид — как бы вдавливаясь (визуально меняется в зависимости от платформы, но кнопка всегда делает хоть что-то, чтобы показать, что она нажата).*

*На самом деле вопрос звучит так: «Как мне заставить кнопку делать что-либо в тот момент, когда пользователь нажимает ее?»*

**Понадобятся две вещи.**

- ① **Метод**, вызываемый при нажатии (действие, которое должно произойти в результате нажатия кнопки).
- ② **Способ узнать**, когда запустить данный метод. Другими словами, это способ распознать, когда пользователь нажимает кнопку.

Тема 13. Графический пользовательский интерфейс, отслеживание и получение событий

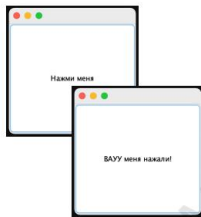


The IDE interface shows the project structure on the left, the code editor in the center, and the Run window at the bottom. The code in SimpleGui.java is as follows:

```
1 import javax.swing.*;
2
3 public class SimpleGui {
4     public static void main (String[] args) {
5         JFrame frame = new JFrame();
6         JButton button = new JButton("Кнопка на всё окно");
7
8         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
9         frame.getContentPane().add(button);
10        frame.setSize(300, 300);
11        frame.setVisible(true);
12    }
13 }
```

The Run window shows the command: `D:\Apps\OpenJDK\bin\java.exe -javaagent:idea-2023.3.5\lib\idea_2023.3.5\jre\bin\java.exe`. The output window shows the text: "Кнопка на всё окно".

- 1 Реализуем интерфейс `ActionListener`.
- 2 Регистрируем кнопку (сообщаем ей, что хотим отслеживать ее события).
- 3 Определяем метод обработки события (реализуем метод `actionPerformed()` из интерфейса `ActionListener`).



Тема 13. Графический пользовательский интерфейс,  
отслеживание и получение событий

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

Новый оператор импорта для пакета в котором  
хранятся ActionListener и ActionEvent.

```
public class SimpleGUIB implements ActionListener {
    JButton button;

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

```
public void go() {
    JFrame frame = new JFrame();
    button = new JButton("click me");
```

② `button.addActionListener(this);`

```
frame.getContentPane().add(button);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(300,300);
frame.setVisible(true);
```

```

3 public void actionPerformed(ActionEvent e) {
    button.setText("I've been clicked!");
}

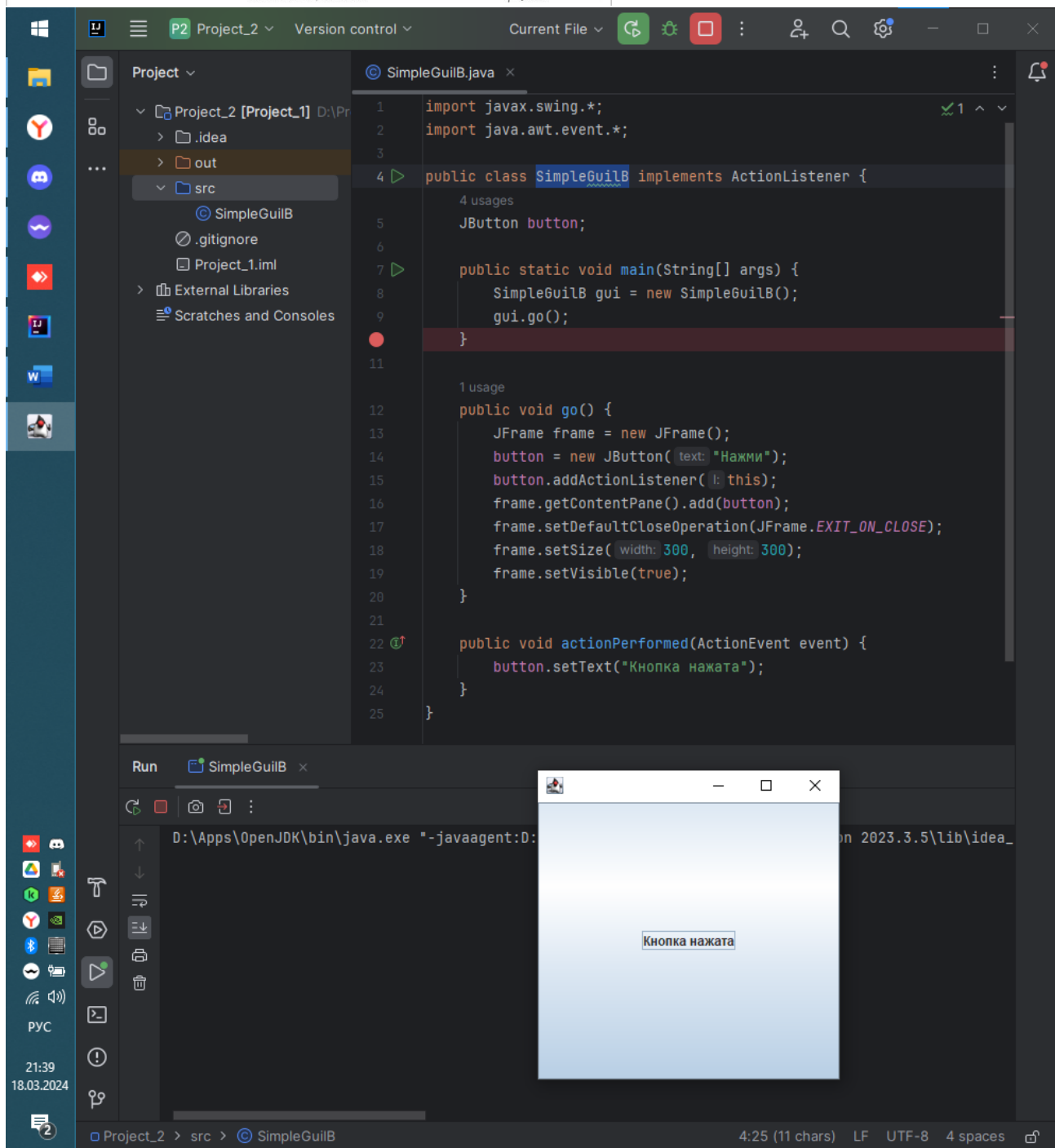
```

Listener ( *Результатом взаимодействия. Ког. вызовутся:  
Этот класс SimpleListener реализует  
интерфейс ActionListener.  
Класс будет наследовать от класса  
AbstractListener, который реализует  
интерфейс ActionListener.* )

Регистрируем нашу заинтересованность в кнопке. Под говорит кнопке: «Добавь меня к своему списку слушателей». Передаваемый аргумент ДОЛЖЕН быть объектом класса `Runnable` или `ActionListener`!

pearl333-  
me.EXIT\_ON\_CLOSE);  
Pearl333ем memop actionPerformed()  
интерфейса ActionListener. Это  
фактически memop обработчик  
событий!

Кнопка вызывает этот метод, чтобы извести о наступлении события. Она отправляет объект ActionEvent как аргумент, но он нам не нужен. Достаточно знать, что событие произошло.



## Создайте личный виджет для рисования

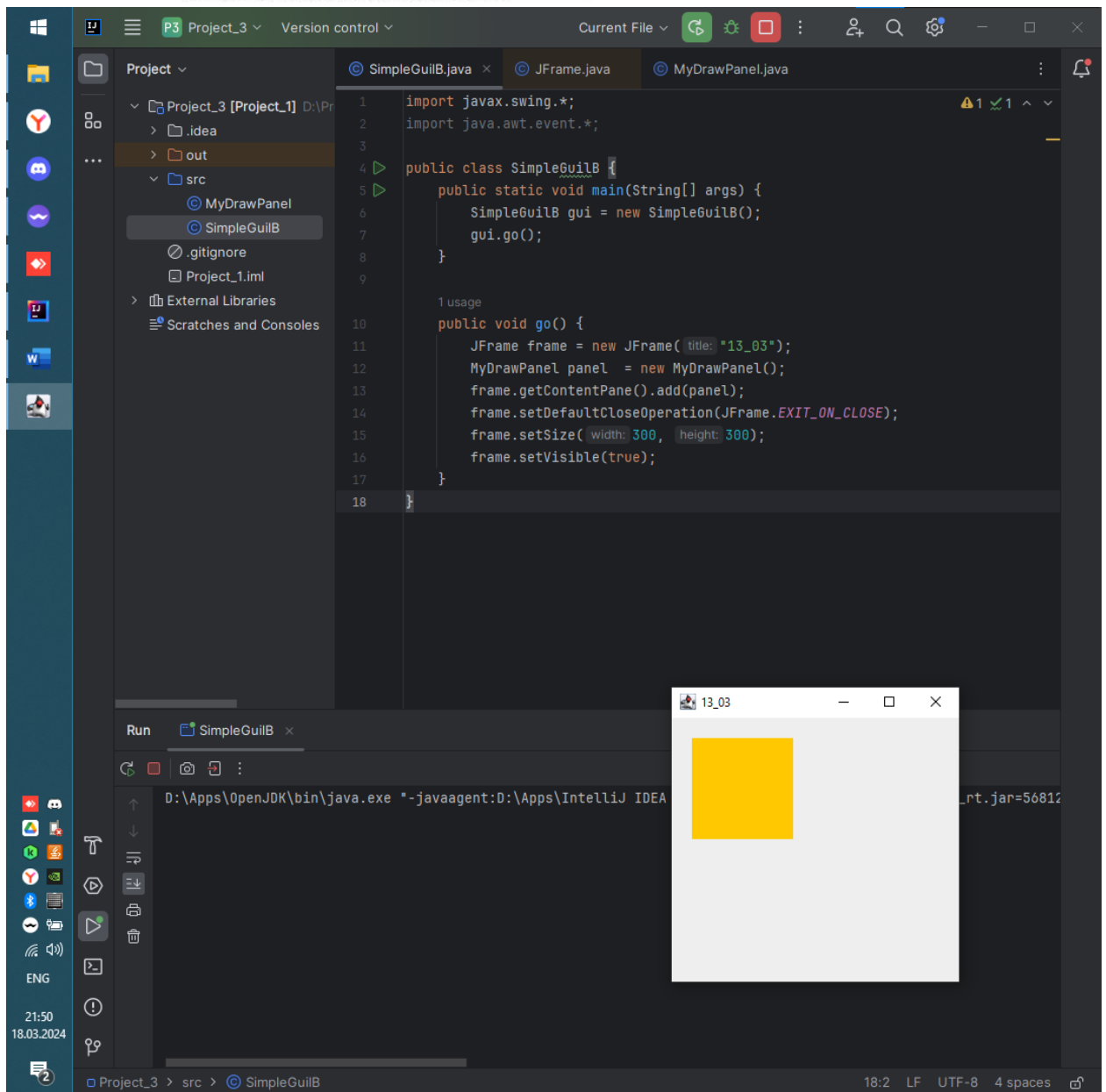
Если вы хотите добавить собственную графику, лучше всего создать личный виджет для рисования. Его нужно поместить во фрейм, как кнопку или любой другой виджет, и, когда вы выведете его на экран, увидите свои изображения. Вы можете даже заставить их двигаться с помощью анимации или менять цвета на экране каждый раз, когда нажимаете кнопку.

Это проще простого.

**Создайте подкласс JPanel и замените один метод под названием `paintComponent()`.**

Весь ваш графический код располагается внутри метода `paintComponent()`. Считайте, что этот метод вызывается системой, чтобы сказать: «Эй, виджет, пришло время нарисовать себя». Если вы хотите нарисовать круг, то метод `paintComponent()` будет содержать код для рисования круга. Когда фрейм, содержащий вашу панель для отрисовки, будет показан на экране, будет вызван метод `paintComponent()` и появится круг. Если пользователь свернет окно, то JVM будет знать, что фрейму нужно «восстановиться» при разворачивании, и снова вызовет метод `paintComponent()`. В любое время, когда JVM посчитает, что экран нужно обновить, будет вызван ваш метод `paintComponent()`.

Помните еще кое-что: *вы никогда не будете вызывать этот метод сами!* Аргумент для него (объект `Graphics`) — это фактически холст для рисования, который наложили на *настоящий* экран. Вы не сможете добиться этого самостоятельно: он должен быть передан вам системой. Однако позже вы увидите, что можете запрашивать систему на обновление дисплея (`repaint()`), что в итоге приведет к вызову метода `paintComponent()`.



## Что интересного можно сделать в paintComponent()

Рассмотрим несколько вещей, которые можно сделать в paintComponent(). Хотя интереснее будет, если вы сами начнете экспериментировать. Поиграйте с числами и поищите в API класс Graphics (позже вы увидите, что ваши возможности далеко не ограничиваются этим классом).

### Изображаем JPEG

```
public void paintComponent(Graphics g) {  
    Image image = new ImageIcon("catzilla.jpg").getImage();  
    g.drawImage(image, 3, 4, this);  
}
```

Имя вашего файла указывается здесь.

Координаты x и y для верхнего левого угла картиншки. Когда говорит: «Отступить 3 пиксела от левого края панели и 4 пиксела от верхнего». Эти числа всегда относятся к виджету (в данном случае к вашему наследнику JPanel), а не ко всему фрейму.



```
1 import javax.swing.*;
2 import java.awt.event.*;
3
4 public class SimpleGuiB {
5     public static void main(String[] args) {
6         SimpleGuiB gui = new SimpleGuiB();
7         gui.go();
8     }
9
10    1 usage
11    public void go() {
12        JFrame frame = new JFrame( title: "13_04");
13        MyDrawPanel panel = new MyDrawPanel();
14        frame.getContentPane().add(panel);
15        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16        frame.setSize( width: 300, height: 300);
17        frame.setVisible(true);
18    }
19 }
```

Run SimpleGuiB

D:\Apps\OpenJDK\bin\java.exe "-javaagent:D:\Ap...

13\_04

11:41 LF UTF-8 4 spaces

## Рисуем на черном фоне круг произвольного цвета

```
public void paintComponent(Graphics g) {
```

```
    g.fillRect(0,0,this.getWidth(), this.getHeight());
```

```
    int red = (int) (Math.random() * 255);
    int green = (int) (Math.random() * 255);
    int blue = (int) (Math.random() * 255);
```

```
    Color randomColor = new Color(red, green, blue);
    g.setColor(randomColor);
    g.fillOval(70,70,100,100);
}
```

Начинаем рисование с 70 пикселей слева и 70 пикселей сверху, а также задаем ширину и высоту по 100 пикселей.

Закрасим всю панель черным (цвет по умолчанию).

Два первых аргумента определяют координаты верхнего левого угла по отношению к панели, где начнется рисование. Здесь 0,0 означает: «Начни с 0 пикселей от левого края и 0 пикселей от верхнего». Два других аргумента говорят: «Сделай ширину прямоугольника, как у панели (this.getWidth()), и высоту такую же, как у панели (this.getHeight())».

Вы можете задать цвет тремя целыми числами, представляющими RGB-значения.



```
1 import javax.swing.*;
2 import java.awt.event.*;
3
4 public class SimpleGuiB {
5     public static void main(String[] args) {
6         SimpleGuiB gui = new SimpleGuiB();
7         gui.go();
8     }
9
10    1 usage
11    public void go() {
12        JFrame frame = new JFrame("13_05");
13        MyDrawPanel panel = new MyDrawPanel();
14        frame.getContentPane().add(panel);
15        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16        frame.setSize( width: 300, height: 300);
17        frame.setVisible(true);
18    }
19 }
```

Run SimpleGuiB

D:\Apps\OpenJDK\bin\java.exe -javaagent:D:\Apps\IntelliJ IDEA Community Edition 2023.3.5\lib\idea\_rt.jar=32107:D:\Apps\OpenJDK\bin\java.exe

Project\_5 > src > SimpleGuiB > go

Жизнь слишком коротка, чтобы тратить ее на рисование одноцветных кругов, когда есть плавный градиент



Это настоящий объект Graphics2D, замаскированный под простой объект Graphics.

Указываем его, чтобы иметь возможность вызвать нечто такое, что есть у Graphics2D, но отсутствует у Graphics.

GradientPaint gradient = new GradientPaint(70, 70, Color.blue, 150, 150, Color.orange);

Начальная точка      Начальный цвет      Конечная точка      Конечный цвет

Здесь мы назначаем для виртуальной кисти градиент вместо сплошного цвета.

Метод fillOval() в данном случае позволяет закрасить овал тем, что находится на кисти — градиентом.

```

public void paintComponent(Graphics g) {
    Graphics2D g2d = (Graphics2D) g;
    GradientPaint gradient = new GradientPaint(70, 70, Color.blue, 150, 150, Color.orange);
    g2d.setPaint(gradient);
    g2d.fillOval(70, 70, 100, 100);
}

```

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows 'Project\_6 [Project\_1]' and 'External Libraries'.
- Code Editor:** Displays the code for `SimpleGuiB.java`. The `go()` method is highlighted, showing the creation of a `JFrame` and a `MyDrawPanel`.
- Run Console:** Shows the command `D:\Apps\OpenJDK\bin\java.exe -javaagent:D:\Apps\In...` and the output `jar=32138:0`.
- Application Window:** A window titled '13\_05' is open, displaying a circle with a smooth gradient from blue to yellow.