

# Основы программирования (Java)

ФИСТ 1 курс

Власенко

Олег

Федосович

**Лекция 3**

**Методы. Рекурсия**

# Вычисление факториала

```
public class Main1 {  
  
    public static long fuct(int n) {  
        long res = 1;  
        for(int i = 1; i <= n; i++) {  
            res *= i;  
        }  
        return res;  
    }  
  
    public static void main(String[] args) {  
  
        int n = 4;  
        long f = fuct(n);  
  
        System.out.println(n + "! = " + f);  
  
    }  
}
```

# Вычисление факториала - рекурсия

```
public class Main1 {  
  
    private static long fuct2(int n) {  
  
        if (n == 0) {  
            return 1;  
        }  
  
        long res = fuct2(n - 1) * n;  
        return res;  
    }  
  
    public static void main(String[] args) {  
        int n = 4;  
        long f = fuct2(n);  
        System.out.println(n + "! = " + f);  
    }  
}
```

# Трассировка

- 1) Трассировка итерационной реализации
- 2) Трассировка рекурсивной реализации
- 3) Ручная трассировка итерации
- 4) Ручная трассировка рекурсии

# Рекурсия – загадка 1

Что будет выведено?

```
public class Main2 {  
  
    public static void main(String[] args) {  
        rec1(3);  
    }  
  
    public static void rec1(int n) {  
        System.out.print(" " + n);  
        if (n > 1) {  
            rec1(n - 1);  
        }  
    }  
}
```

# Рекурсия – загадка 2

Что будет выведено?

```
public class Main2 {  
  
    public static void main(String[] args) {  
        rec1(3);  
    }  
  
    public static void rec1(int n) {  
        if (n > 1) {  
            rec1(n - 1);  
        }  
        System.out.print(" " + n);  
    }  
}
```

# Рекурсия – загадка 3

Что будет выведено?

```
public class Main2 {  
  
    public static void main(String[] args) {  
        rec1(3);  
    }  
  
    public static void rec1(int n) {  
        System.out.print(" " + n);  
        if (n > 1) {  
            rec1(n - 1);  
        }  
        System.out.print(" " + n);  
    }  
}
```

# Рекурсия – загадка 4

Что будет выведено?

```
public class Main2 {  
  
    public static void main(String[] args) {  
        rec2(4);  
    }  
  
    public static void rec2(int n) {  
  
        if (n >= 1) {  
            System.out.print(" " + n);  
            rec2(n - 1);  
        }  
  
    }  
}
```



# Рекурсия – загадка 5

Что будет выведено?

```
public class Main2 {  
  
    public static void main(String[] args) {  
        rec2(3);  
    }  
  
    public static void rec2(int n) {  
  
        if (n >= 1) {  
            System.out.print(" " + n);  
            rec2(n - 1);  
            rec2(n - 1);  
        }  
  
    }  
}
```

# Рекурсия – загадка 6

Что будет выведено?

```
public class Main2 {  
  
    public static void main(String[] args) {  
        rec2(3);  
    }  
  
    public static void rec2(int n) {  
  
        if (n >= 1) {  
            System.out.print(" " + n);  
            rec2(n - 1);  
            rec2(n - 1);  
            System.out.print(" " + n);  
        }  
    }  
}
```

# Рекурсия – загадка 7 (ЕГЭ 2017 Демо)

Задача 11: Чему равна сумма напечатанных на экране чисел при выполнении вызова  $F(10)$ ?

```
public class Main2 {  
  
    public static void main(String[] args) {  
        F(10);  
    }  
  
    static void F(int n) {  
        if (n > 2) {  
            System.out.printf("%d\n", n);  
            F(n - 3);  
            F(n - 4);  
        }  
    }  
}
```

# Рекурсия – загадка 8 (ЕГЭ 2015 Демо)

Задача 11: Чему равна сумма всех чисел, напечатанных на экране при выполнении вызова `F(1)`?

```
public class Main2 {  
  
    public static void main(String[] args) {  
        F(1);  
    }  
  
    static void F(int n)  
    {  
        System.out.printf("%d\n", n);  
        if (n < 5) {  
            F(n + 1);  
            F(n + 3);  
        }  
    }  
}
```

# Рекурсия – загадка 9 (ЕГЭ 2016 Демо)

Задача 11: записаны две рекурсивные функции : **F** и **G**.  
Сколько символов «звёздочка» будет напечатано на экране при выполнении вызова F(11)?

```
public class Main2 {
    public static void main(String[] args) {
        F(11);
    }

    static void F(int n) {
        if (n > 0)
            G(n - 1);
    }

    static void G(int n) {
        System.out.printf("*");
        if (n > 1)
            F(n - 3);
    }
}
```

# Вычисление чисел Фибоначчи

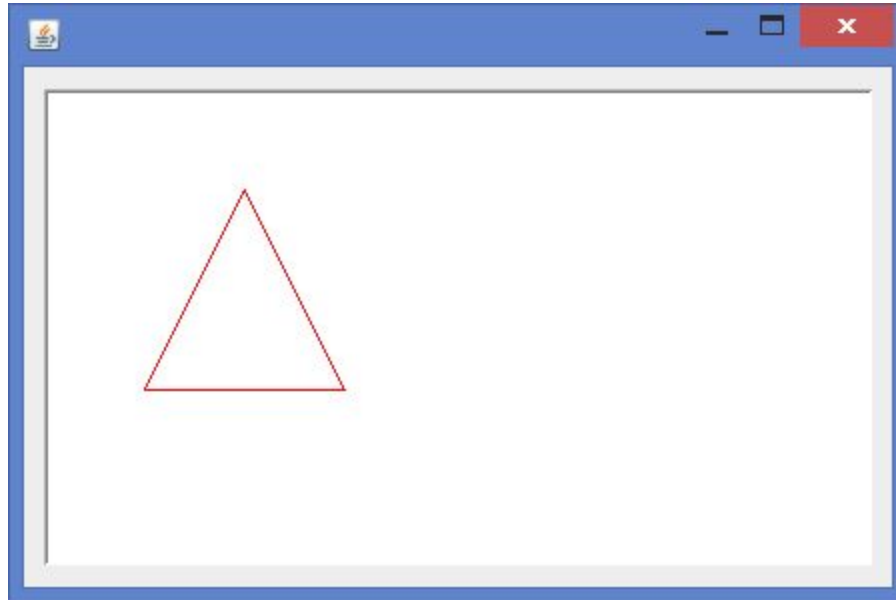
```
public class Main3 {
    public static void main(String[] args) {
        System.out.println(fib(6));
    }
    public static int fib(int n) {
        if (n <= 2) {
            return 1;
        } else {
            int s = 0;
            int f1 = 1;
            int f2 = 1;
            for(int i = 2; i < n; i++) {
                s = f1 + f2;
                f1 = f2;
                f2 = s;
            }
            return s;
        }
    }
}
```

# Вычисление чисел Фибоначчи (рекурсия)

```
public class Main3 {  
  
    public static void main(String[] args) {  
        System.out.println(fib2(6));  
  
    }  
  
    public static int fib2(int n) {  
  
        if (n <= 2) {  
            return 1;  
        } else {  
            return fib2(n - 1) + fib2(n - 2);  
        }  
  
    }  
  
}
```

# Метод для рисования треугольника

Создать приложение, в котором на панели рисуется 1 треугольник



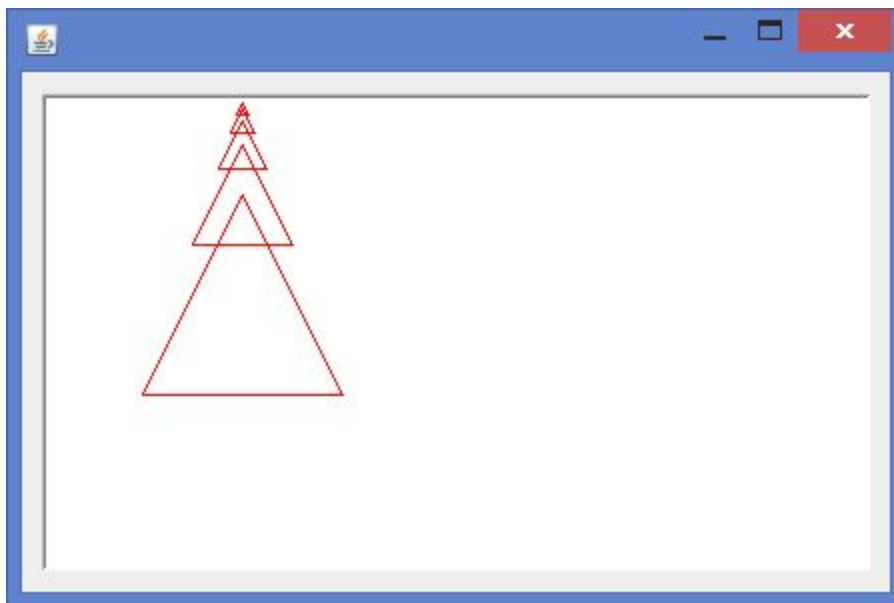


# Метод для рисования треугольника (код)

```
public class MyPanel3 extends JPanel {  
  
    private void drawTriangle(Graphics g,  
        int cx, int cy, int size) {  
        int x1 = cx;  
        int y1 = cy - size;  
        int x2 = cx - size;  
        int y2 = cy + size;  
        int x3 = cx + size;  
        int y3 = cy + size;  
        g.setColor(Color.RED);  
        g.drawLine(x1, y1, x2, y2);  
        g.drawLine(x2, y2, x3, y3);  
        g.drawLine(x3, y3, x1, y1);  
    }  
  
    @Override  
    public void paint(Graphics g) {  
        super.paint(g);  
        drawTriangle(g, 100, 100, 50);  
    }  
}
```

# Рекурсивный метод для рисования треугольника

Создать приложение, в котором на панели рисуется  
рекурсивный треугольник



# Код рекурсивного метода

```
public class MyPanel3 extends JPanel {
    ...

    private void triangleRec(Graphics g,
        int cx, int cy, int size) {

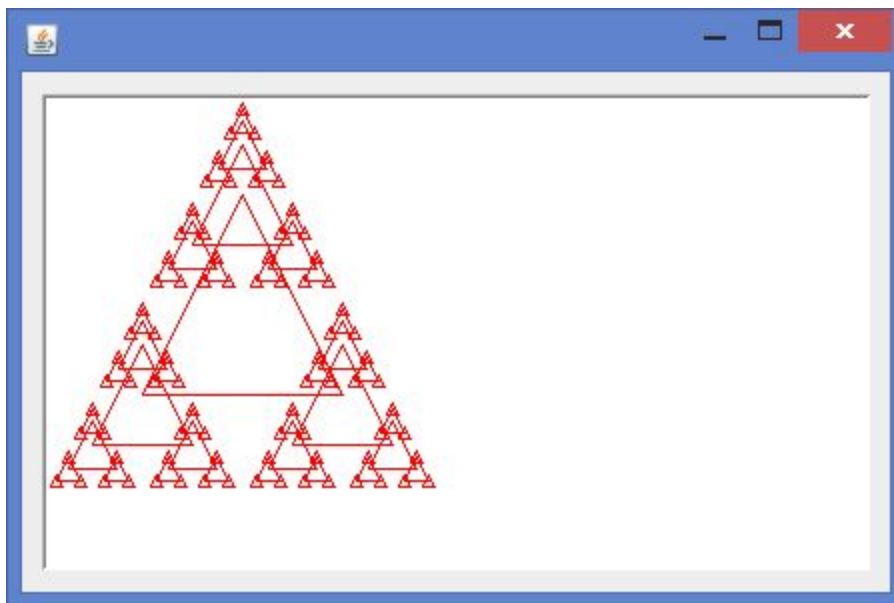
        drawTriangle(g, cx, cy, size);

        if (size < 4) {
            return;
        }
        triangleRec(g, cx, cy - size, size / 2);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        triangleRec(g, 100, 100, 50);
    }
}
```

# Рекурсивный метод для рисования треугольника

Создать приложение, в котором на панели рисуется  
рекурсивный треугольник



# Код рекурсивного метода

```
public class MyPanel3 extends JPanel {
    ...

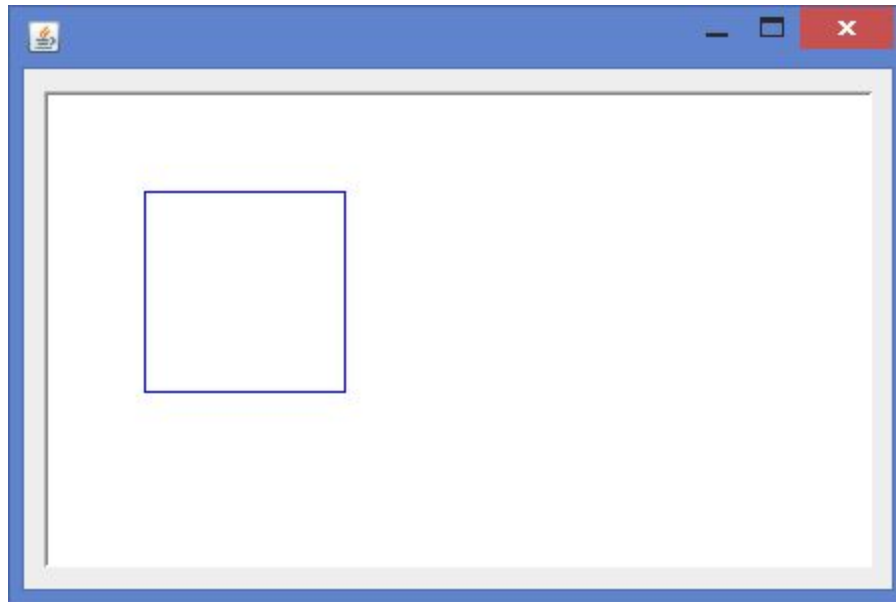
    private void triangleRec(Graphics g,
        int cx, int cy, int size) {
        drawTriangle(g, cx, cy, size);

        if (size < 4) {
            return;
        }
        triangleRec(g, cx, cy - size, size / 2);
        triangleRec(g, cx - size, cy + size, size / 2);
        triangleRec(g, cx + size, cy + size, size / 2);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        triangleRec(g, 100, 100, 50);
    }
}
```

# Метод для рисования квадрата

Создать приложение, в котором на панели рисуется 1 квадрат

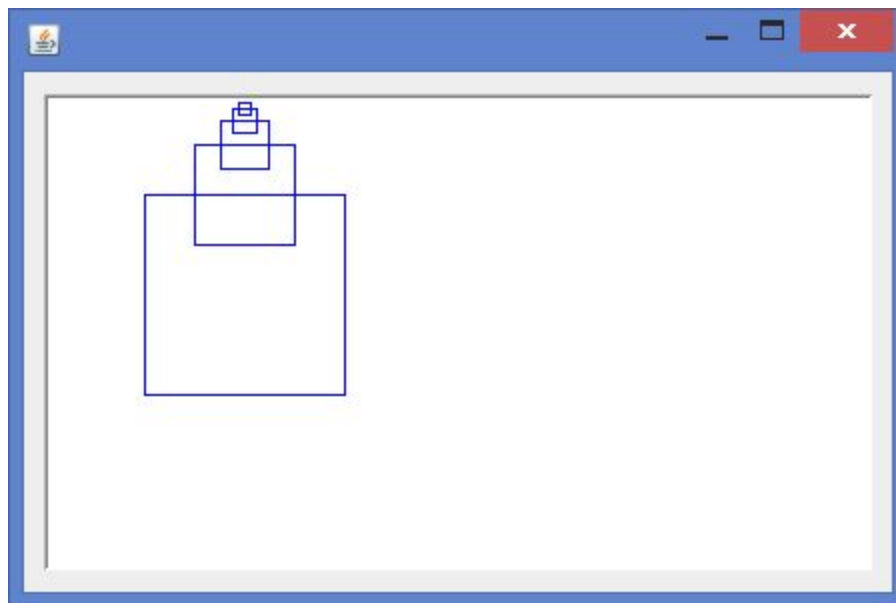


# Метод для рисования квадрата (код)

```
public class MyPanel3 extends JPanel {  
  
    private static void drawSquare(Graphics g,  
        int cx, int cy, int size){  
        int x1 = cx - size;  
        int y1 = cy - size;  
        int width = size * 2;  
        int height = size * 2;  
  
        g.setColor(Color.BLUE);  
        g.drawRect(x1, y1, width, height);  
    }  
  
    @Override  
    public void paint(Graphics g) {  
        super.paint(g);  
        drawSquare(g, 100, 100, 50);  
    }  
}
```

# Рекурсивный метод для рисования квадрата

Создать приложение, в котором на панели рисуется рекурсивный квадрат





# Метод для рисования квадрата (код)

```
public class MyPanel3 extends JPanel {
    ...

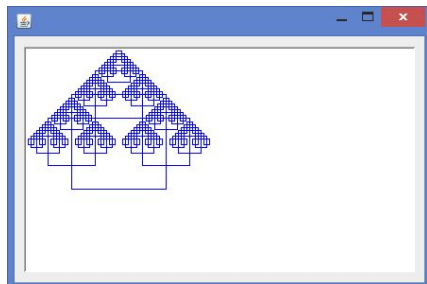
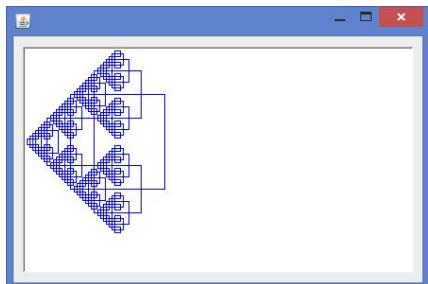
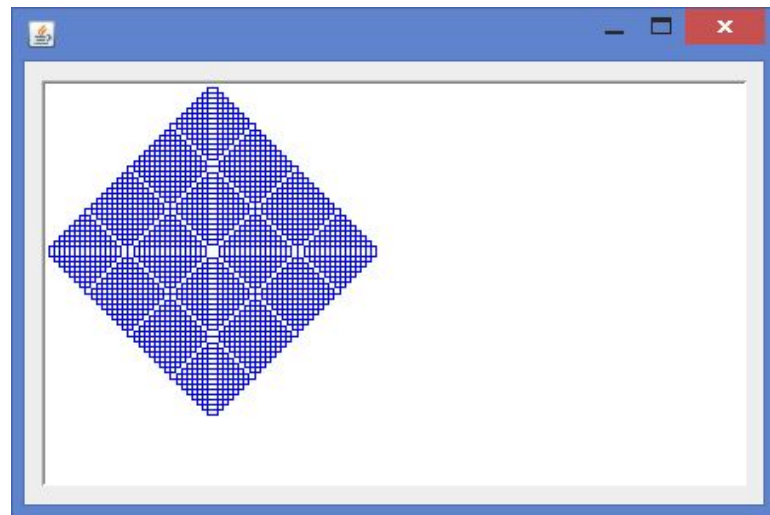
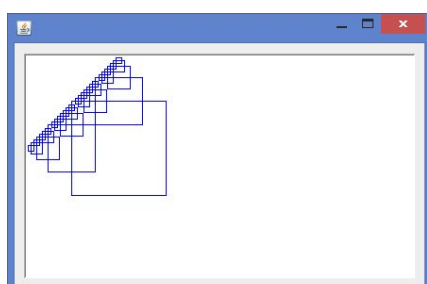
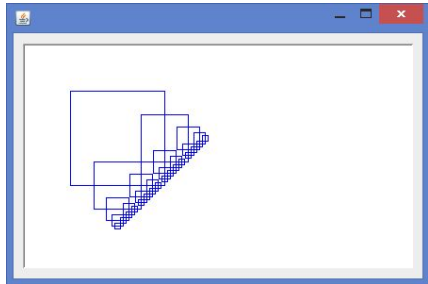
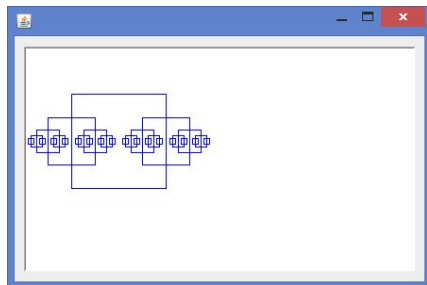
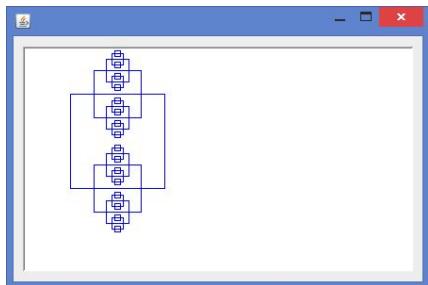
    private void squareRec(Graphics g,
        int cx, int cy, int size) {
        drawSquare(g, cx, cy, size);

        if (size < 4) {
            return;
        }
        squareRec(g, cx, cy - size, size / 2);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        squareRec(g, 100, 100, 50);
    }
}
```

# Рекурсивный метод для рисования квадрата

Создать приложение, в котором на панели рисуется рекурсивный квадрат



# Метод для рисования квадрата (код)

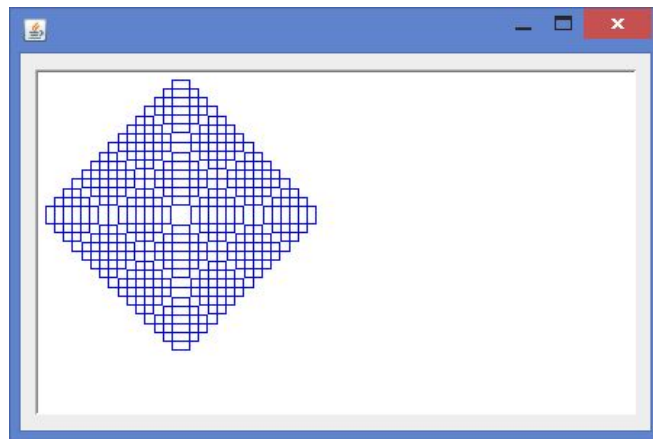
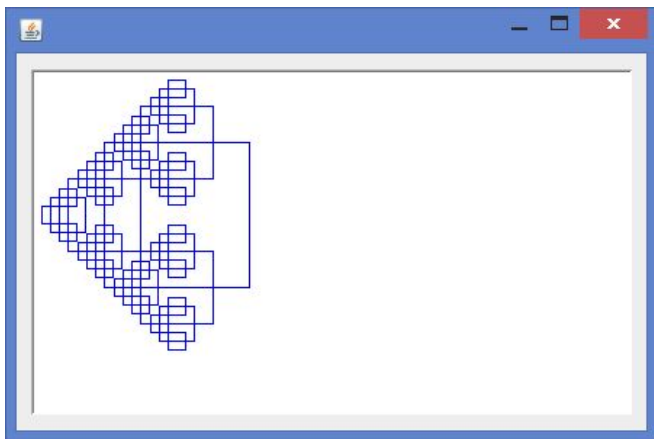
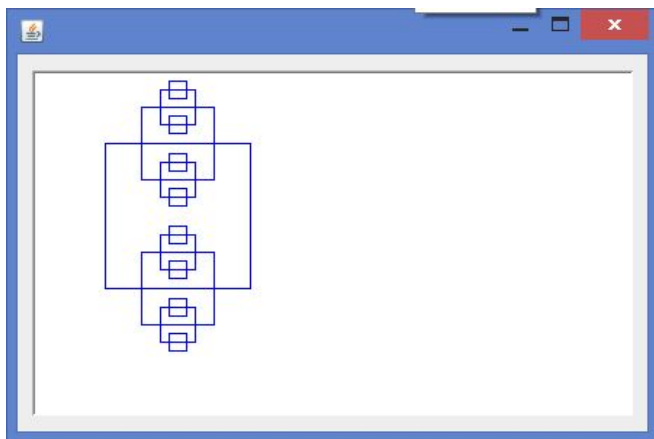
```
public class MyPanel3 extends JPanel {
    ...
    private void squareRec(Graphics g,
        int cx, int cy, int size) {
        drawSquare(g, cx, cy, size);

        if (size < 4) {
            return;
        }
        squareRec(g, cx, cy - size, size / 2);
        squareRec(g, cx, cy + size, size / 2);
        squareRec(g, cx - size, cy, size / 2);
        squareRec(g, cx + size, cy, size / 2);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        squareRec(g, 100, 100, 50);
    }
}
```

# Рекурсивный метод для рисования квадрата

Создать приложение, в котором на панели рисуется рекурсивный квадрат



# Метод для рисования квадрата (код)

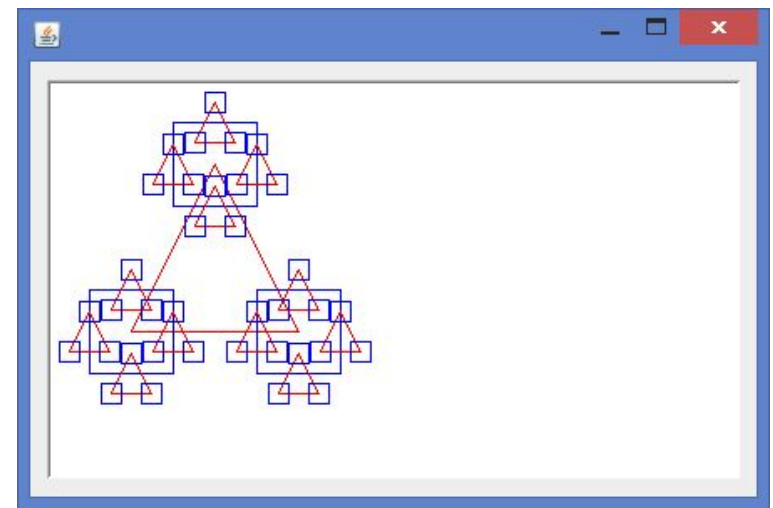
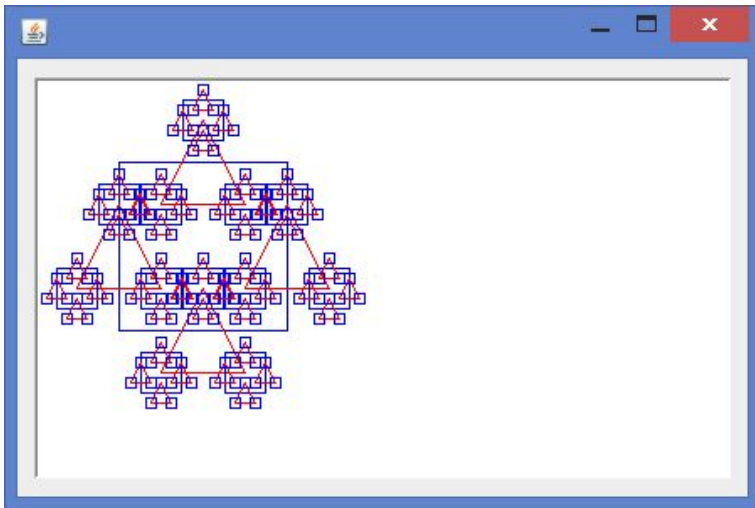
```
public class MyPanel3 extends JPanel {
    ...
    private void squareRec(Graphics g,
        int cx, int cy, int size) {
        drawSquare(g, cx, cy, size);

        if (size < 10) {
            return;
        }
        squareRec(g, cx, cy - size, size / 2);
        squareRec(g, cx, cy + size, size / 2);
        squareRec(g, cx - size, cy, size / 2);
        squareRec(g, cx + size, cy, size / 2);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        squareRec(g, 100, 100, 50);
    }
}
```

# Косвенная рекурсия

Создать приложение, в котором на панели рисуется рекурсивный квадрат, созданный из рекурсивных треугольников, которые созданы из квадратов и т.д.

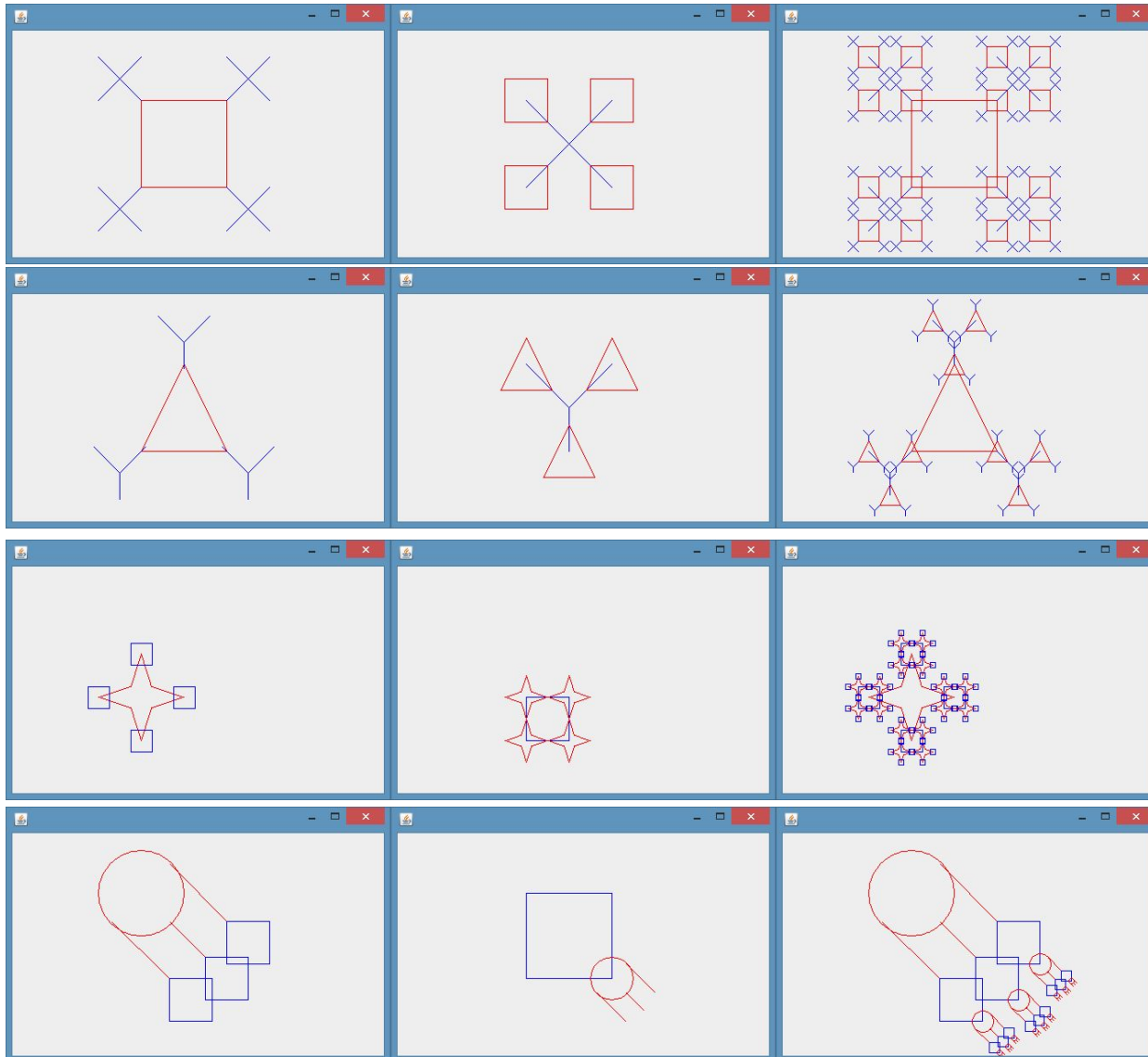


# Код

```
private void triangleRec2(Graphics g,  
    int cx, int cy, int size) {  
    drawTriangle(g, cx, cy, size);  
    if (size < 4) {  
        return;  
    }  
    squareRec2(g, cx, cy - size, size / 2);  
    squareRec2(g, cx - size, cy + size, size / 2);  
    squareRec2(g, cx + size, cy + size, size / 2);  
}
```

```
private void squareRec2(Graphics g,  
    int cx, int cy, int size) {  
    drawSquare(g, cx, cy, size);  
    if (size < 10) {  
        return;  
    }  
    triangleRec2(g, cx, cy - size, size / 2);  
    triangleRec2(g, cx, cy + size, size / 2);  
    triangleRec2(g, cx - size, cy, size / 2);  
    triangleRec2(g, cx + size, cy, size / 2);  
}
```

# Косвенная рекурсия (примеры)





# Домашнее задание

1. Делайте лабы!
2. Начинайте делать курсовую работу
3. Решить загадки 1-9