ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ
ПО ДИСЦИПЛИНЕ
«Защита информации»

Руководители _____ Е.А. Харченко

Москва 2023

СПИСОК ИСПОЛНИТЕЛЕЙ

Студ. группы 201-361 _____ А.Е. Сильченко

Ход работы:

Для генерации и визуализации всех решений уравнения вида:

$$y^2 \equiv x^3 + ax + b \ (mod\ p)$$

Зададим переменный a, b и конечное поле p.

Далее генерируем точки, т.е. перебираем все точки поля p и проверяем лежат ли они на кривой. Метод для генерации точек и проверки представлен на рисунке 1 – 2.

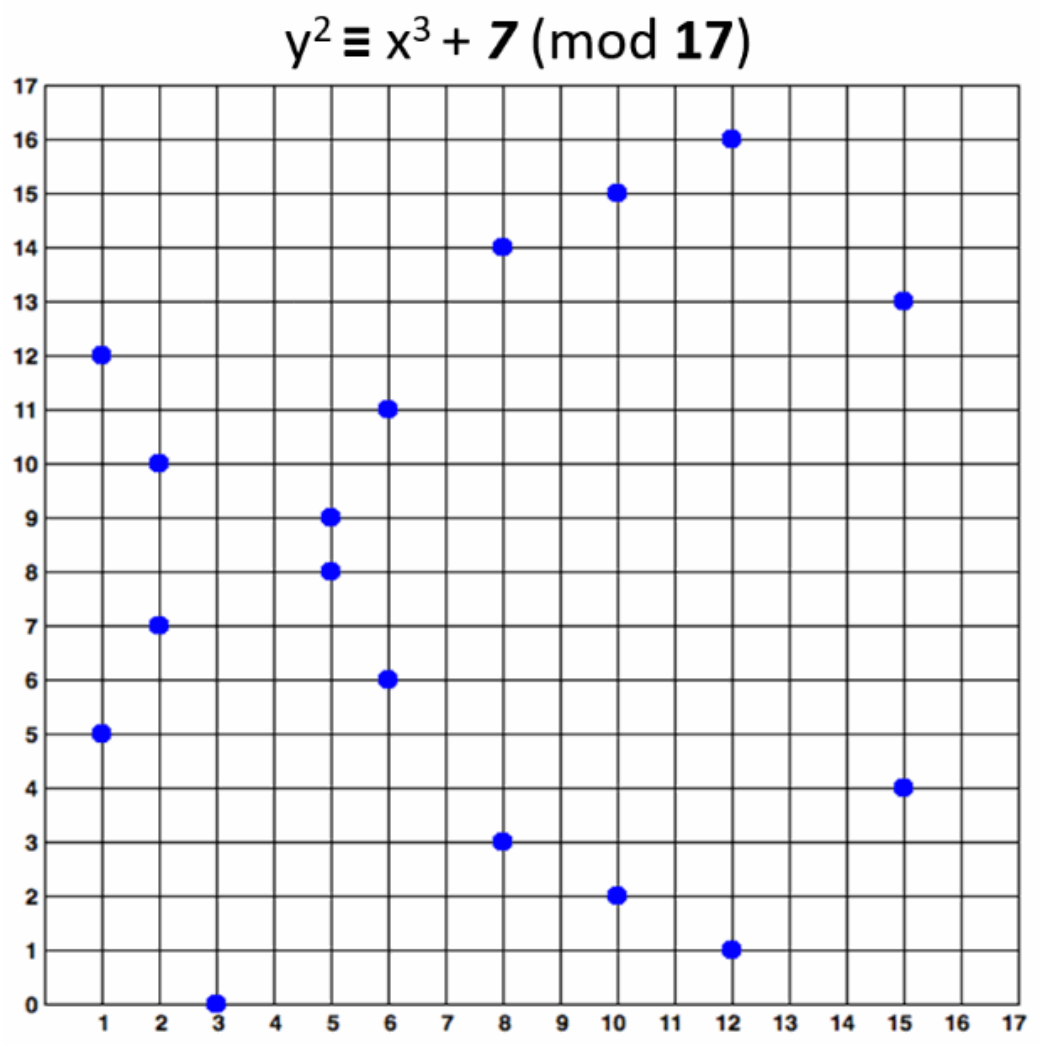
```
1 usage
public static ArrayList<Point> generatePoints(int a, int b, int p) {
    ArrayList<Point> points = new ArrayList<~>();

    for (int x = 0; x < p; x++) {
        for (int y = 0; y < p; y++) {
            Point point = new Point(x, y, a, b, p);
            if (point.isOnCurve()) {
                points.add(point);
            }
        }
    }

    return points;
}
```

Рисунок 1 – Метод для генерации точек.

```
public boolean isOnCurve() {
    int left = (y * y) % p;
    int right = (x * x * x + a * x + b) % p;
    return left == right;
```

Рисунок 2 – Проверка лежит ли точка на кривой.

В данной программе, переменная a = 0, b = 7, а конечное поле p = 17, следовательно эллиптическая кривая должна выглядеть следующим образом:



На рисунке 3, представлена эллиптическая кривая которая получилась в результате выполнения моей программы.

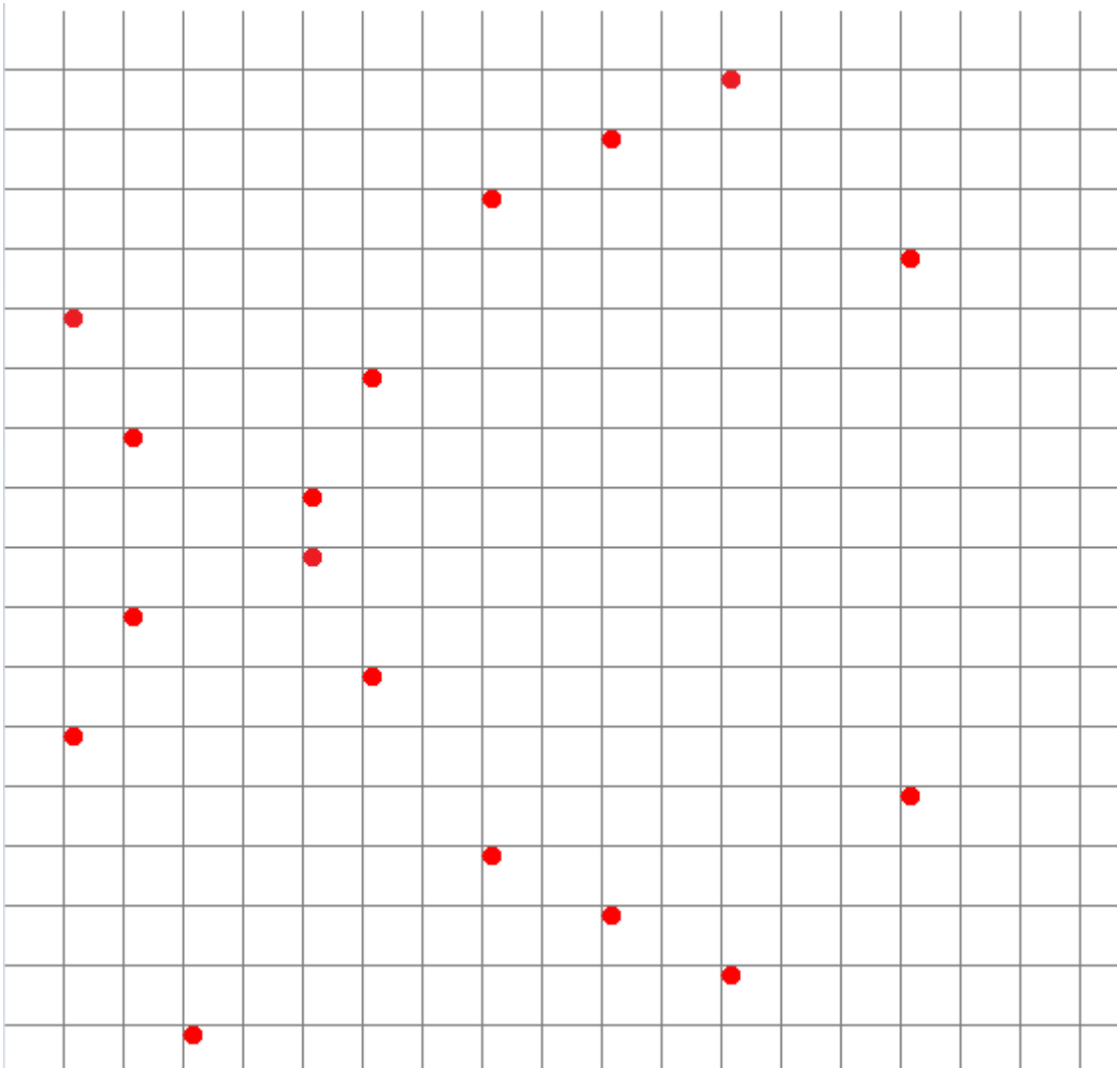


Рисунок 3 – Эллиптическая кривая.

Для сложения точек в данной программе реализован метод summ рисунок 4. В методе рассматриваются два случая: удвоение точки и сложение. Если координаты текущей точки и точки, переданной в качестве аргумента, равны, то это значит, что нужно выполнить удвоение точки, для этого вызывается метод doublePoint(). В противном случае, для сложения двух точек используется формула Лангранжа-Барроуза. Далее вычисляются числитель и знаменатель по формулам: numerator = other.getY() - y и denominator = other.getX() - x соответственно.

Затем вызывается вспомогательная функция multiplicativeInverse(), которая вычисляет обратный элемент denominator в поле p, и вычисляется

значение s по формуле: s = numerator * multiplicativeInverse(denominator, p) % p.

В следующих двух строках вычисляются новые значения координат x и y новой точки, в соответствии с формулами: x3 = (s * s - x - other.getX()) % p и y3 = (s * (x - x3) - y) % p.

Наконец, в последних двух строках параметры x3 и y3 изменяются, если они отрицательны, чтобы получить правильные значения.

В результате метод возвращает новую точку, полученную при сложении двух точек на эллиптической кривой результат представлен на рисунке 5. Также в этой программе использован метод multiplicativeInverse для вычисления обратного элемента в поле p метод представлен на рисунке 6. Внутри метода переменная a принимает значение a mod p, чтобы убедиться, что она находится в диапазоне от 0 до p-1. Затем производится цикл от x = 1 до p-1, чтобы найти такое значение x, при котором (a * x) mod p равно 1. Если такое значение x найдено, функция возвращает его. Если же такого значения не существует, метод возвращает -1.

```
public Point summ(Point other) {
    if (x == other.getX() && y == other.getY()) {
        // Удвоение точки
        return doublePoint();
    }

    int numerator = other.getY() - y; //числитель
    int denominator = other.getX() - x; //знаменатель

    int s = numerator * multiplicativeInverse(denominator, p)%p;

    int x3 = (s * s - x - other.getX()) % p;
    int y3 = (s * (x - x3) - y) % p;

    if (x3 < 0) x3 += p;
    if (y3 < 0) y3 += p;

    return new Point(x3, y3, a, b, p);
}
```

Рисунок 4 - метод summ.

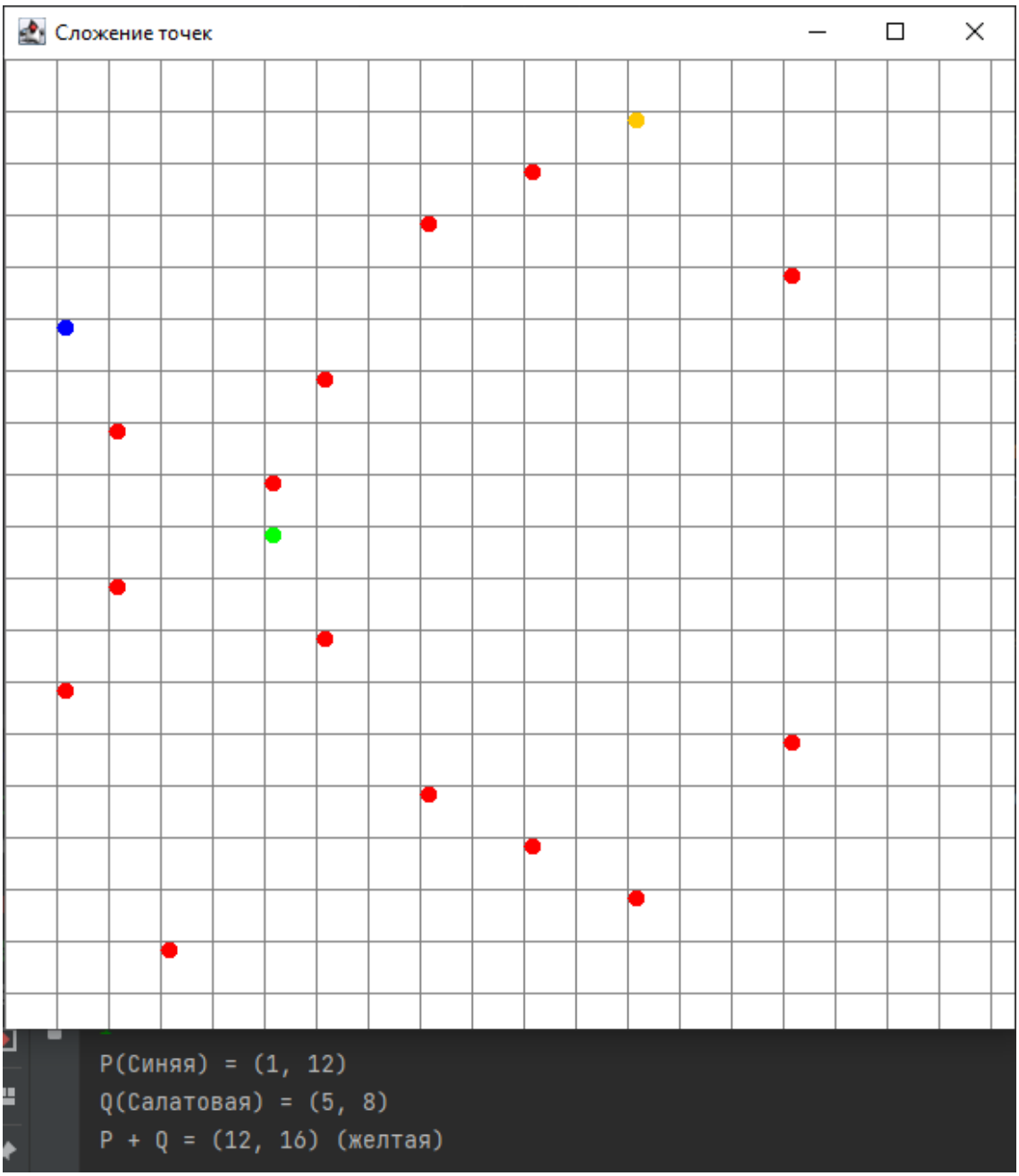


Рисунок 5 – Результат сложения двух точек.

```
private int multiplicativeInverse(int a, int p) {
    a = a % p;
    for (int x = 1; x < p; x++) {
        if ((a * x) % p == 1) {
            return x;
        }
    }
    return -1;
}
```

Рисунок 6 - метод multiplicativeInverse.

Результат сложения двух точек в данном примере можно проверить графически рисунок 7.

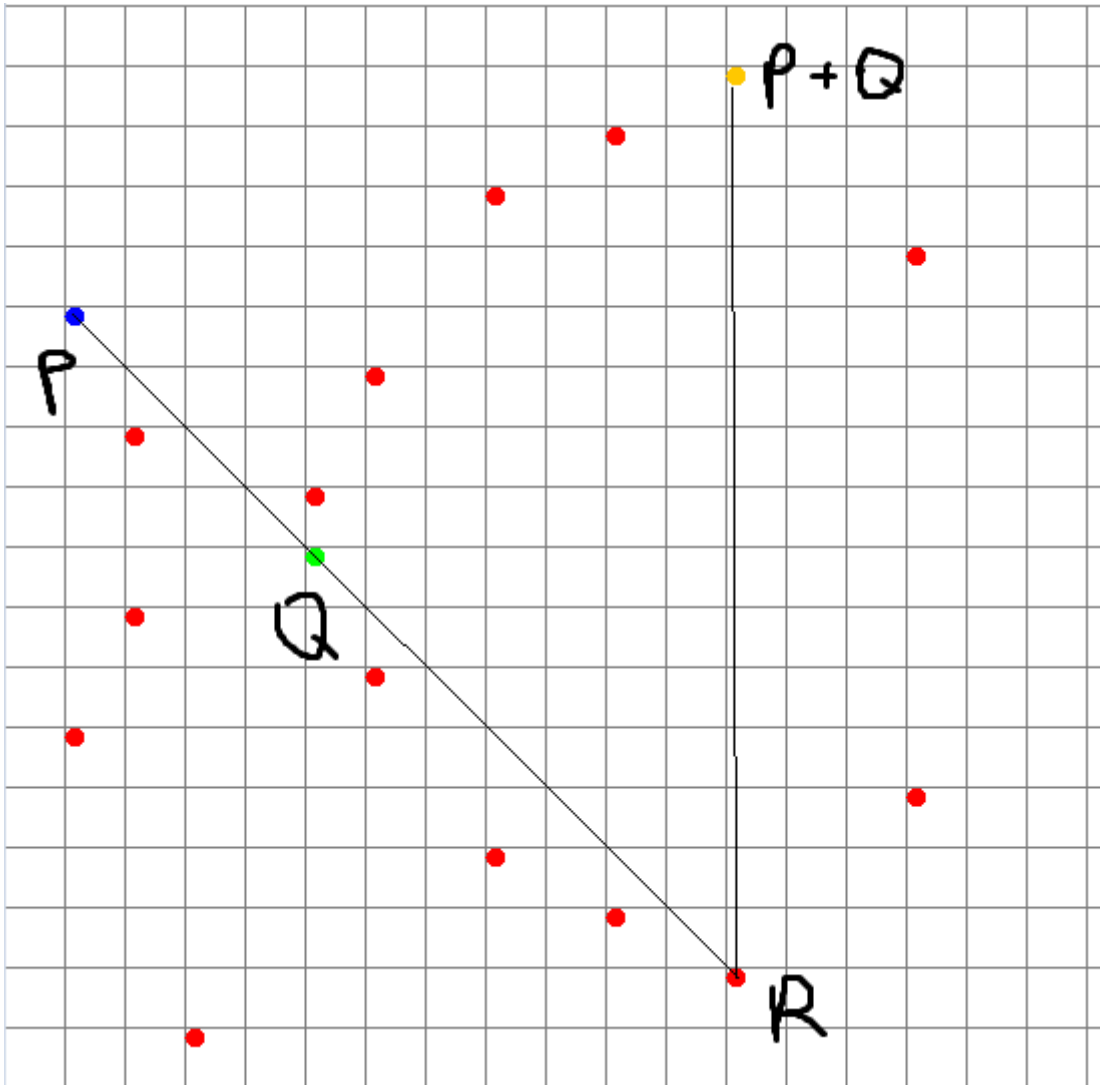


Рисунок 7 – Проверка сложения точек.

Для удвоения точек реализован метод doublePoint. В методе сначала задаются числитель numerator и знаменатель denominator для выражения s, которое вычисляется по формуле Лангранжа-Барроуза, и используемый для последующего вычисления координат точки.

Далее используется вспомогательная функция multiplicativeInverse, которая вычисляет обратный элемент denominator в поле p. Затем значение s вычисляется по формуле: s = numerator * multiplicativeInverse(denominator, p) % p.

Используя s, вычисляются новые значения координат x3 и y3 по формулам:

x3 = s * s - 2 * x

y3 = s * (x - x3) - y

Затем, если одно из значений отрицательно, то к этому значению прибавляется p для получения правильного значения.

Наконец, метод возвращает новую точку с вычисленными координатами x3 и y3.

```
// Операция удвоения точки
3 usages
public Point doublePoint() {
    int numerator = 3 * x * x + a;
    int denominator = 2 * y;

    int s = numerator * multiplicativeInverse(denominator, p) % p;
    int x3 = s * s - 2 * x;
    int y3 = s * (x - x3) - y;

    x3 = (x3 % p + p) % p;
    y3 = (y3 % p + p) % p;

    return new Point(x3, y3, a, b, p);
}
```

Рисунок 8 – метод doublePoint.

Результат удвоения точки (15, 13) представлен на рисунке 9.

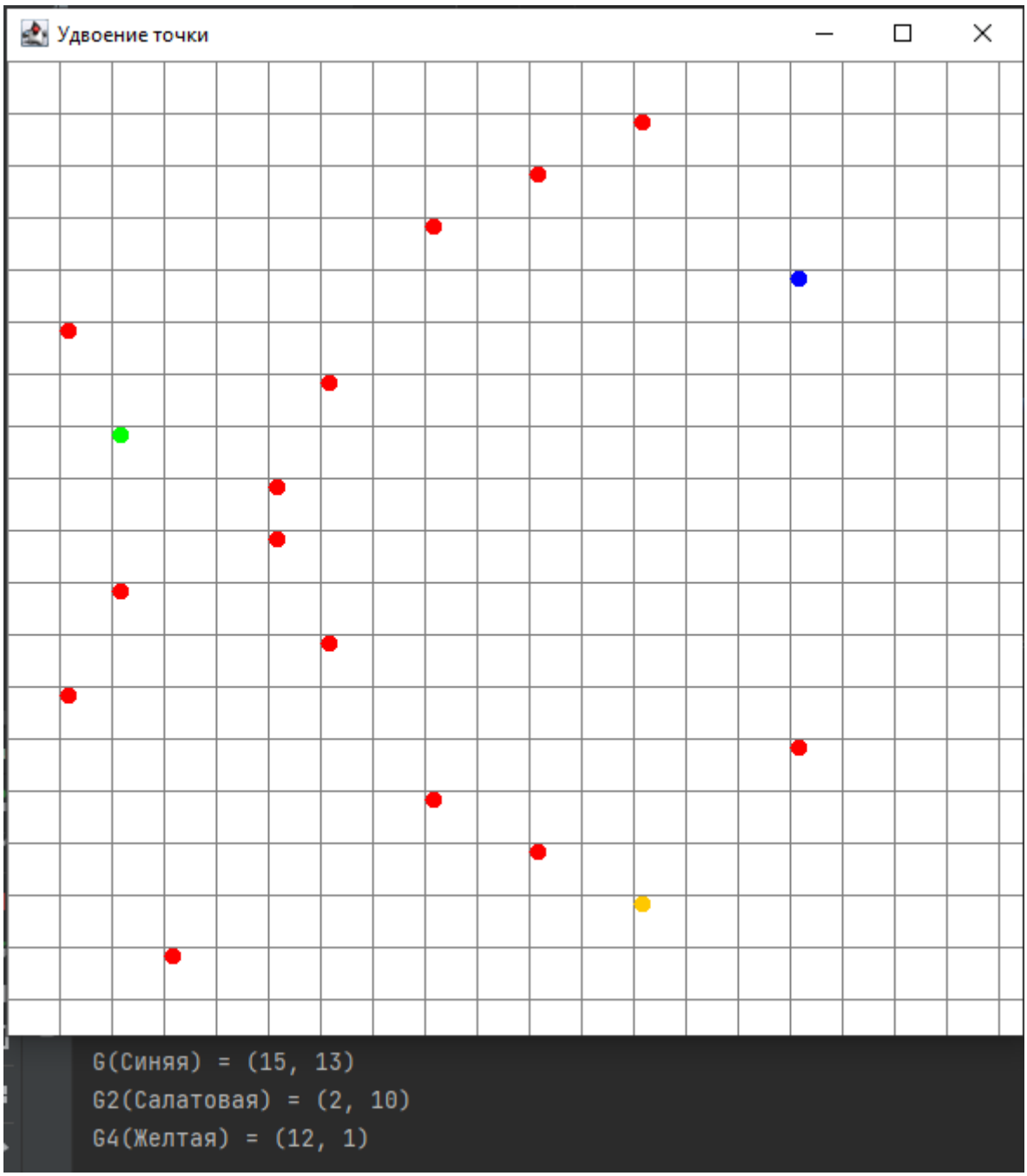


Рисунок 9 – Удвоение точки.

Также для отрисовки осей координат и точек использовался класс Diagram рисунок 10.

```java
public void paint(Graphics g) {
    int width = getSize().width;
    int height = getSize().height;
    int barWidth = Math.max(1, width);
    int[] xArray = new int[points.size()];
    int[] yArray = new int[points.size()];
    int x;
    int y;

    // Отрисовка оси координат
    g.setColor(Color.GRAY);
    for (int i = 0; i < width; i++) {
        g.drawLine( x1: 0 + 30 * i,  y1: 0,  x2: 0 + 30 * i, height);
    }
    for (int i = 0; i < height; i++) {
        g.drawLine( x1: 0,  y1: 0 + 30 * i, width,  y2: 0 + 30 * i);
    }

    // Отрисовка точек
    g.setColor(Color.RED);
    for (int i = 0; i < points.size(); i++) {
        Point point = points.get(i);
        xArray[i] = point.getX();
        x = xArray[i];
        yArray[i] = point.getY();
        y = height / 30 - yArray[i] - 1;
        g.fillOval( x: x * 30,  y: y * 30, width: 10,  height: 10);
    }

    // Дополнительная точка 1
    g.setColor(Color.blue);
    x = summ.getX();
    y = height / 30 - summ.getY() - 1;
    g.fillOval( x: x * 30,  y: y * 30, width: 10,  height: 10);

    // Дополнительная точка 2
    g.setColor(Color.green);
    x = dubl.getX();
    y = height / 30 - dubl.getY() - 1;
    g.fillOval( x: x * 30,  y: y * 30, width: 10,  height: 10);
```

Рисунок 10 – Класс Diagram.