```java
package se.kth.castor;

import org.apache.commons.math3.fraction.BigFraction;

import java.util.function.Consumer;

public class Printer {
    int cellWidth = 4;
    int cellHeight = 4;
    int line = 0;
    int cell = 0;
    Runnable setColorT;
    Runnable setColorF;
    Consumer<Pixel> printPixel;
    public String maxRes = "";
    public String lastPixel = "";

    public Printer(Runnable setColorT, Runnable setColorF, Consumer<Pixel> printPixel) {
        this.setColorF = setColorF;
        this.setColorT = setColorT;
        this.printPixel = printPixel;
    }

    int pow(int a, int b) {
        if (b == 0)
            return 1;
        if (b == 1)
            return a;
        if (b % 2 == 0)
            return pow(a * a, b / 2); // even a=(a^2)^b/2
        else
            return a * pow(a * a, b / 2); // odd a=a*(a^2)^b/2
    }

    public void print(BigFraction[] v) {
        for (int i = 1; i < v.length; i++) {
            print(v[i]);
        }
        maxRes = (cell * cellWidth) + "x" + (2 * line * cellHeight);
        newLine();
    }

    public void print(BigFraction f) {
        print((short) f.getNumeratorAsInt(), (short) f.getDenominatorAsInt());
    }

    public void print(short num, short den) {
        int bit = 15;
        for (int j = 0; j < 4; j++) {
                for (int k = 0; k < 4; k++) {
                    if (((num & pow(2, bit)) > 0) ^ ((j % 2 == 0) ^ (k % 2 == 0))) {
                        setColorT.run();
                    } else {
                        setColorF.run();
                    }
                    printPixel.accept(new Pixel(cell * cellWidth + k, 2 * line * cellHeight + j));
                    bit--;
                }
            }
            bit = 15;
            for (int j = 0; j < 4; j++) {
                for (int k = 0; k < 4; k++) {
                    if (((den & pow(2, bit)) > 0) ^ ((j % 2 == 0) ^ (k % 2 == 0))) {
                        setColorT.run();
                    } else {
                        setColorF.run();
                    }
                    printPixel.accept(new Pixel(cell * cellWidth + k, (2 * line + 1) * cellHeight + j));
                    bit--;
                }
            }
            lastPixel = (cell * cellWidth + 3) + "," + ((2 * line + 1) * cellHeight + 3);
            cell++;
        }

        public class Pixel {
            public Pixel(int x, int y) {
                this.x = x;
                this.y = y;
            }

            public int x;
            public int y;
            public int w = 1;
            public int h = 1;
        }

        public void newLine() {
            line++;
            cell = 0;
        }
    }
```