Solutions for WarmUp Prog.

DragonCurve.java

```java
// Dragon Curve Instructions
// d(n) = d(n-1) + "L" + r(n-1)
// r(n) = d(n-1) + "R" + r(n-1)
// @ Max, March 2020

public class DragonCurve {
    public static void main (String[] args) {
        int n = Integer.parseInt( args[0] );
        System.out.println( instructions( n ) );
    }

    public static String instructions (int n) {
        String d = "F", r = "F";
        for (int i = 1; i <= n; i++) {
            String d_ = d, r_ = r;
            d = d_ + "L" + r_;
            r = d_ + "R" + r_;
        }
        return d;
    }
}

```

DragonCurveTest.bat

```
java DragonCurve 0 > DragonCurvesOut.txt
java DragonCurve 1 >> DragonCurvesOut.txt
java DragonCurve 2 >> DragonCurvesOut.txt
java DragonCurve 3 >> DragonCurvesOut.txt
java DragonCurve 4 >> DragonCurvesOut.txt
java DragonCurve 5 >> DragonCurvesOut.txt
java DragonCurve 6 >> DragonCurvesOut.txt
java DragonCurve 7 >> DragonCurvesOut.txt
java DragonCurve 8 >> DragonCurvesOut.txt

```

DragonCurvesOut.txt

```
F
FLF
FLFLFRF
FLFLFRFLFLFRFRF
FLFLFRFLFLFRFRFLFLFLFRFRFLFRFRF
FLFLFRFLFLFRFRFLFLFLFRFRFLFRFRFLFLFLFRFLFLFRFRFRFLFLFRFRFLFRFRF
FLFLFRFLFLFRFRFLFLFLFRFRFLFRFRFLFLFLFRFLFLFRFRFRFLFLFRFRFLFRFRFLFLFLFRFLFLF
FLFLFRFLFLFRFRFLFLFLFRFRFLFRFRFLFLFLFRFLFLFRFRFRFLFLFRFRFLFRFRFLFLFLFRFLFLF
FLFLFRFLFLFRFRFLFLFLFRFRFLFRFRFLFLFLFRFLFLFRFRFRFLFLFRFRFLFRFRFLFLFLFRFLFLF

```

dragoncurve.c

```c
// Dragon Curve Instructions
// d(n) = d(n-1) + 'L' + r(n-1)
// @ Max, March 2020

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void main (int argc, char* argv[]) {
    int n = atoi( argv[1] );
    char s[(int)pow(2, n+1)];
    // length(d(n)) = 1 + 2^1 + 2^2... + 2^n = 2^(n+1)-1

    s[0] = 'F';         // d(0)
    int M = 1;
    for (int i = 1; i <= n; i++) {
        // d(i-1) in s[0..M)
        s[M] = 'L';     // concat a 'L'
        int M2 = M + M;
        for (int k = M-1; k >= 0; k--)    //  concat r(i-1)
            s[M2 - k] =                   // reverse d(i-1)
                s[k] == 'F' ? 'F' : (s[k] == 'L' ? 'R' : 'L');
        M = M2 + 1;
        // d(i) in s[0..M)
    }
    // d(n) in s[0..M)
    s[M] = '\0';        // end the string

    printf( "%s\n", s );
}
```

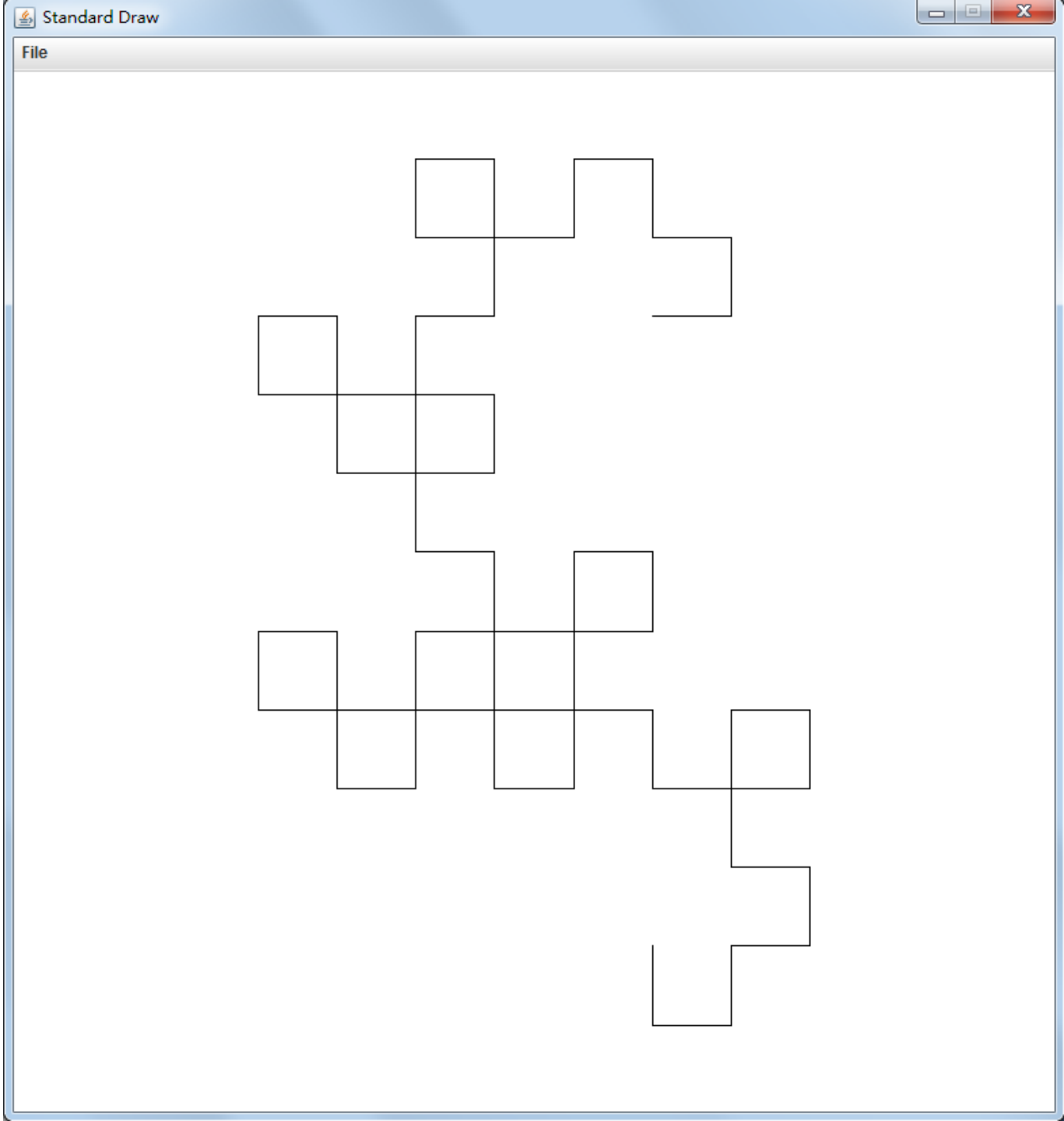
dcTest.bat

```
dragoncurve 0 > dcOut.txt
dragoncurve 1 >> dcOut.txt
dragoncurve 2 >> dcOut.txt
dragoncurve 3 >> dcOut.txt
dragoncurve 4 >> dcOut.txt
dragoncurve 5 >> dcOut.txt
dragoncurve 6 >> dcOut.txt
dragoncurve 7 >> dcOut.txt
dragoncurve 8 >> dcOut.txt
```

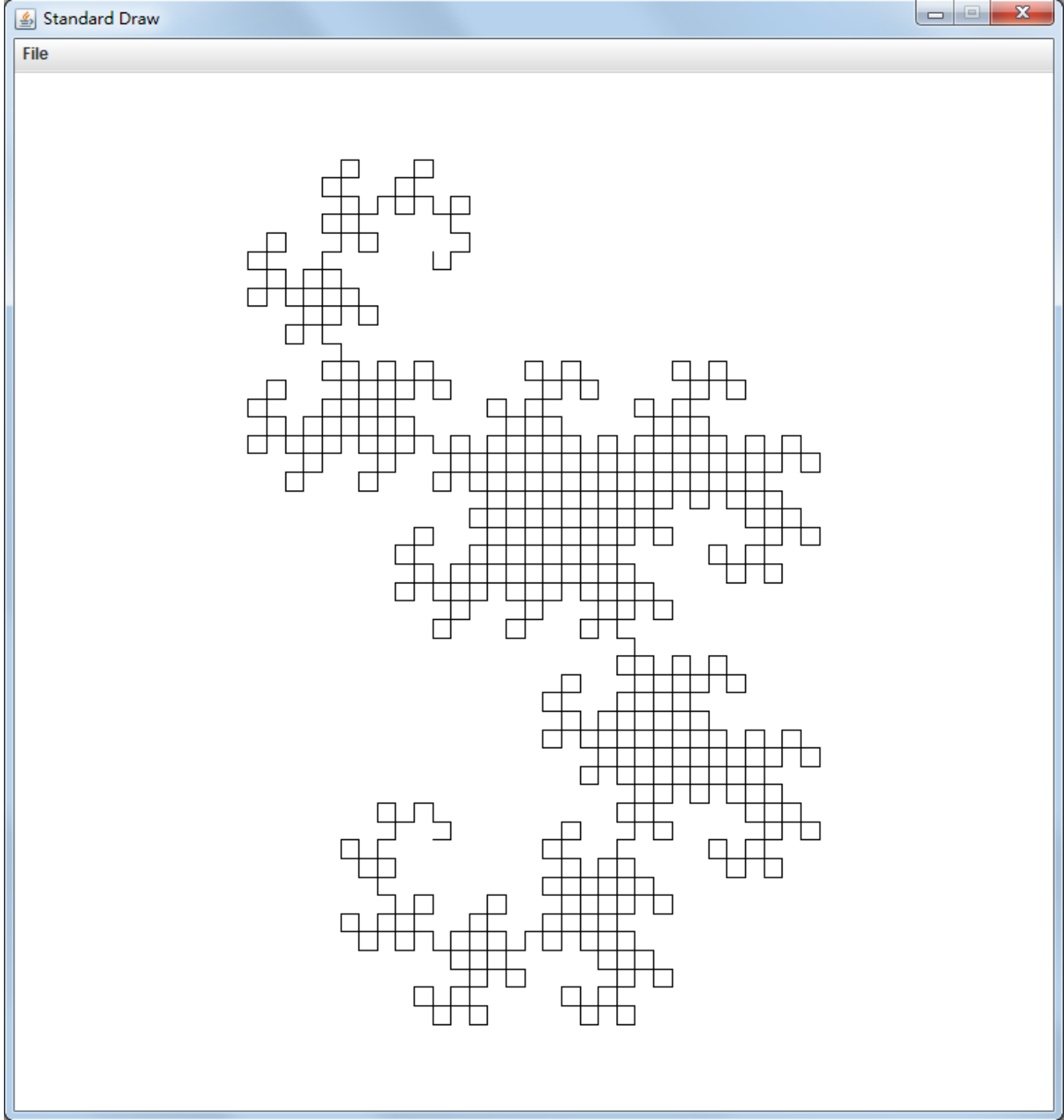
dcOut.txt

```
F
FLF
FLFLFRF
FLFLFRFLFLFRFRF
FLFLFRFLFLFRFRFLFLFLFRFRFLFRFRF
FLFLFRFLFLFRFRFLFLFLFRFRFLFRFRFLFLFLFRFLFLFRFRFRFLFLFRFRFLFRFRF
FLFLFRFLFLFRFRFLFLFLFRFRFLFRFRFLFLFLFRFLFLFRFRFRFLFLFRFRFLFRFRFLFLFLFRFLFLFRFR
FLFLFRFLFLFRFRFLFLFLFRFRFLFRFRFLFLFLFRFLFLFRFRFRFLFLFRFRFLFRFRFLFLFLFRFLFLFRFR
FLFLFRFLFLFRFRFLFLFLFRFRFLFRFRFLFLFLFRFLFLFRFRFRFLFLFRFRFLFRFRFLFLFLFRFLFLFRFR
```

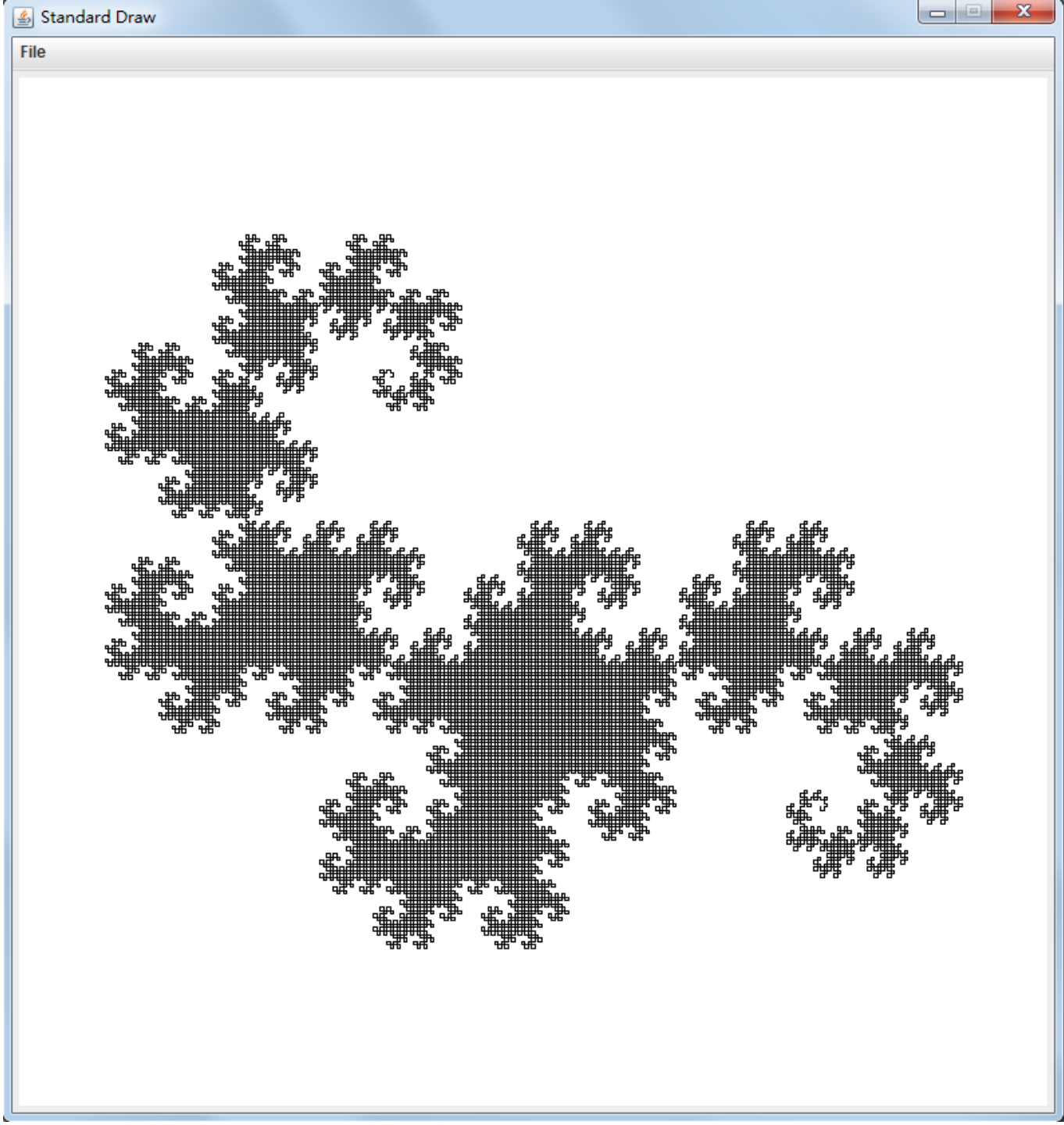
Dragon Curve
Level 6



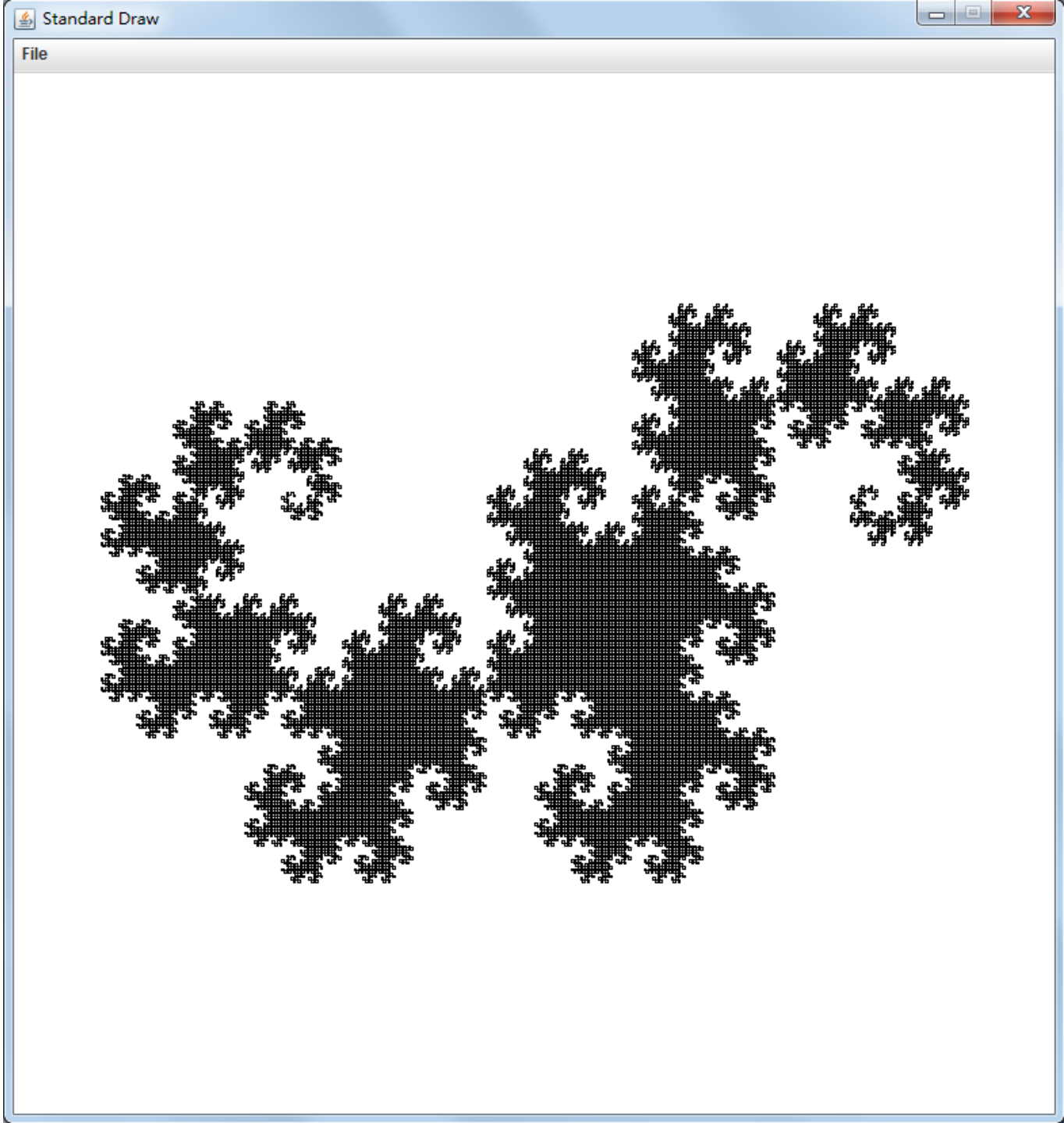
Dragon Curve
Level 10



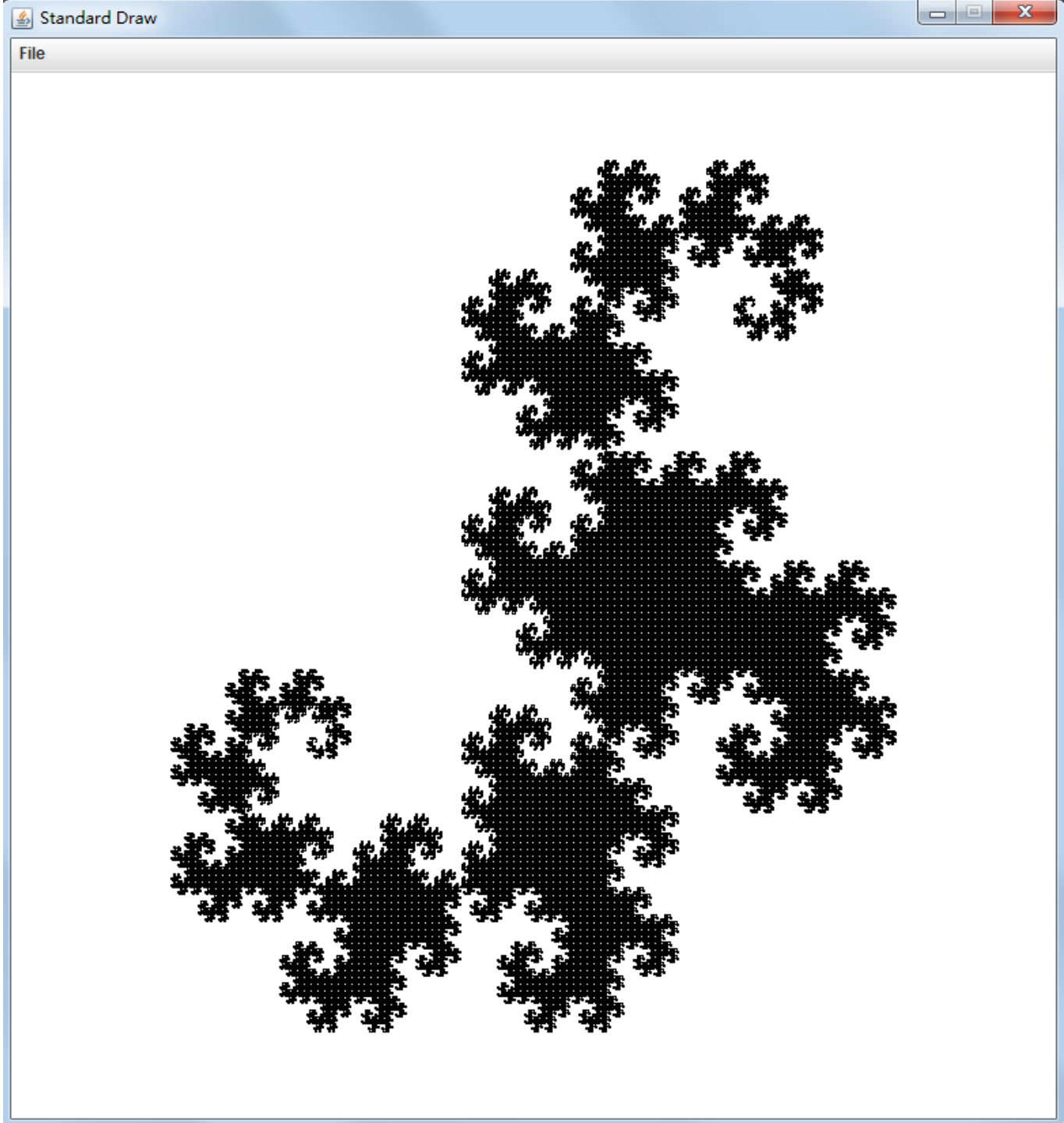
Dragon Curve
Level 15



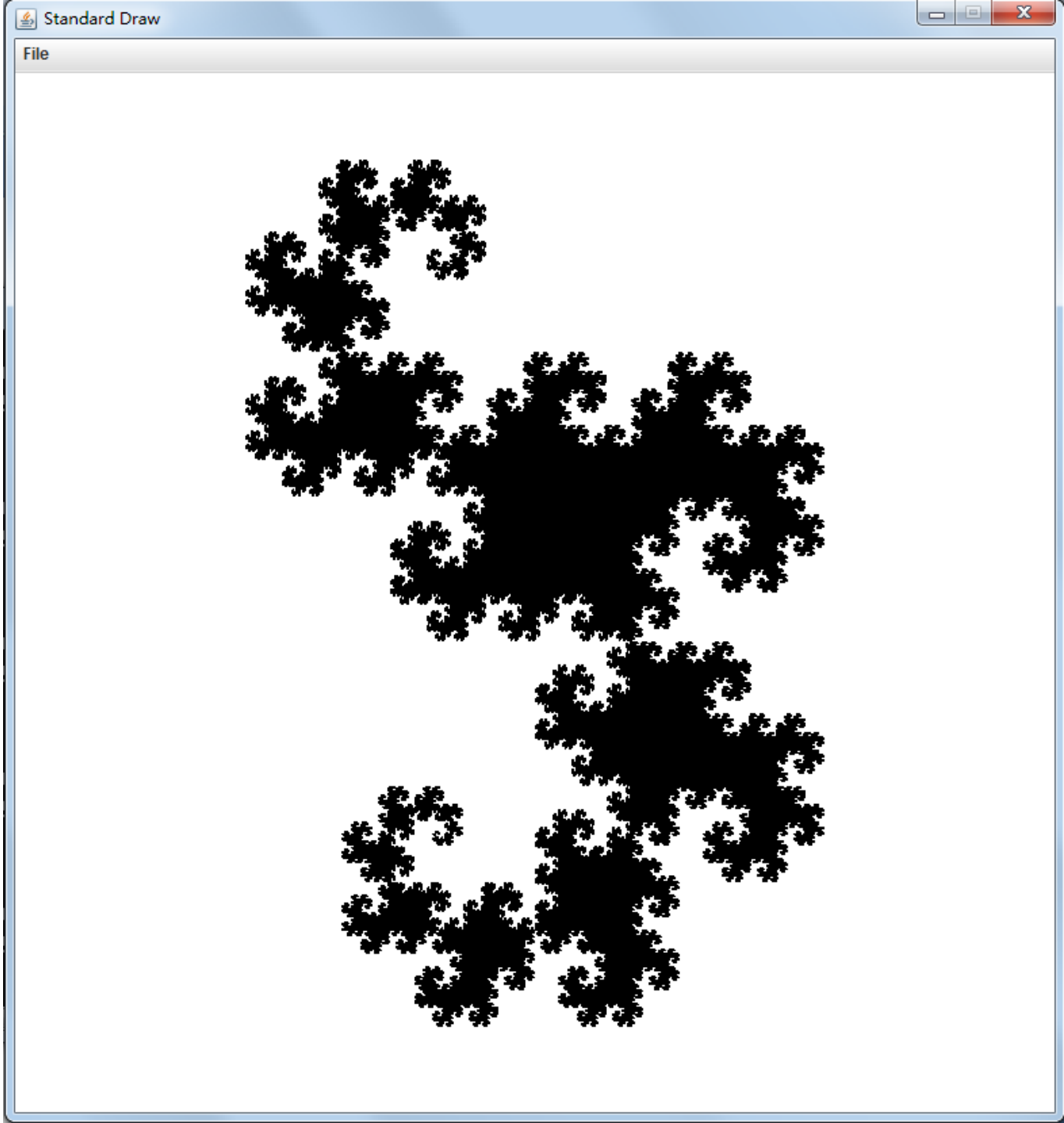
Dragon Curve
Level 16



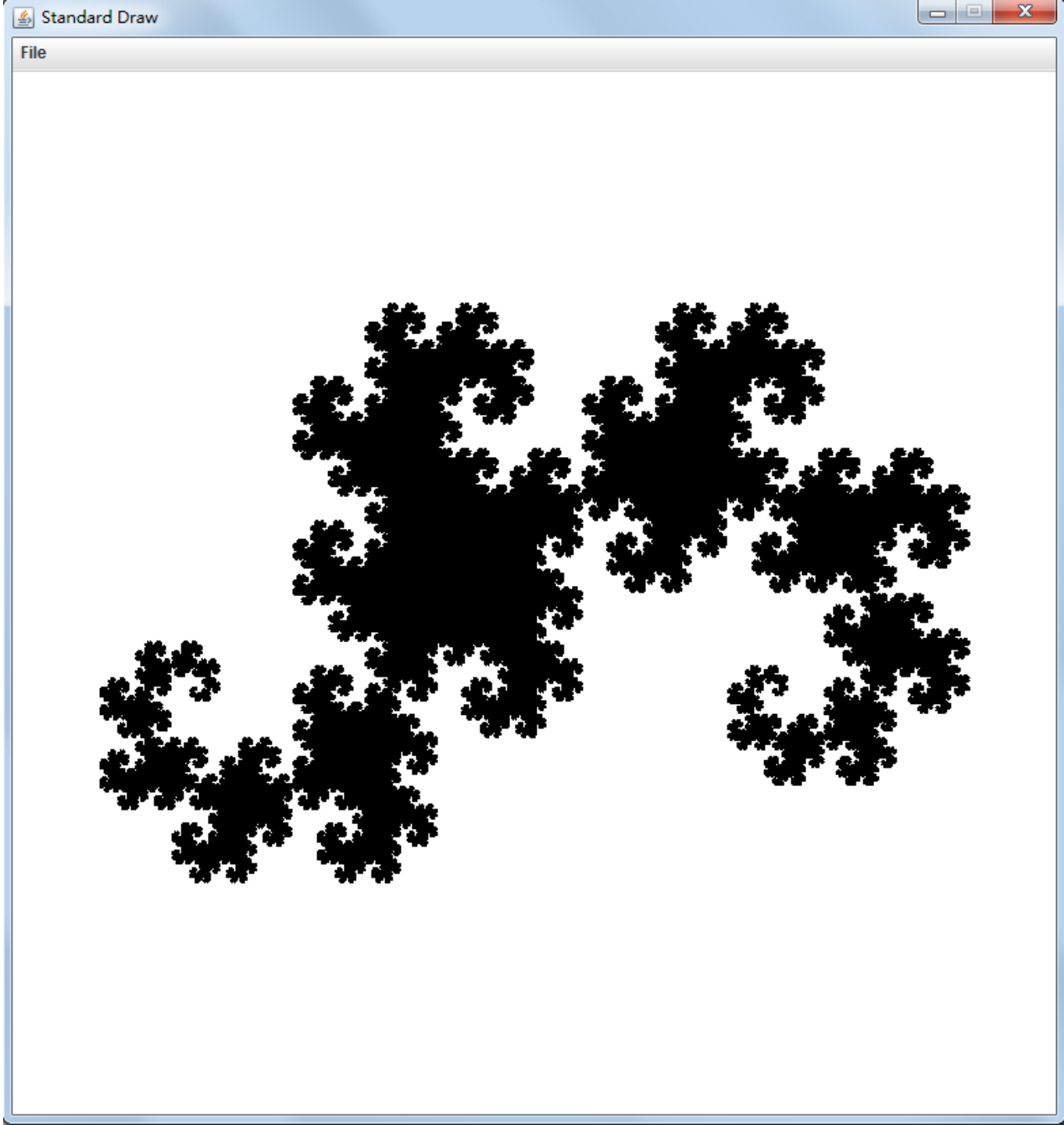
Dragon Curve
Level 17



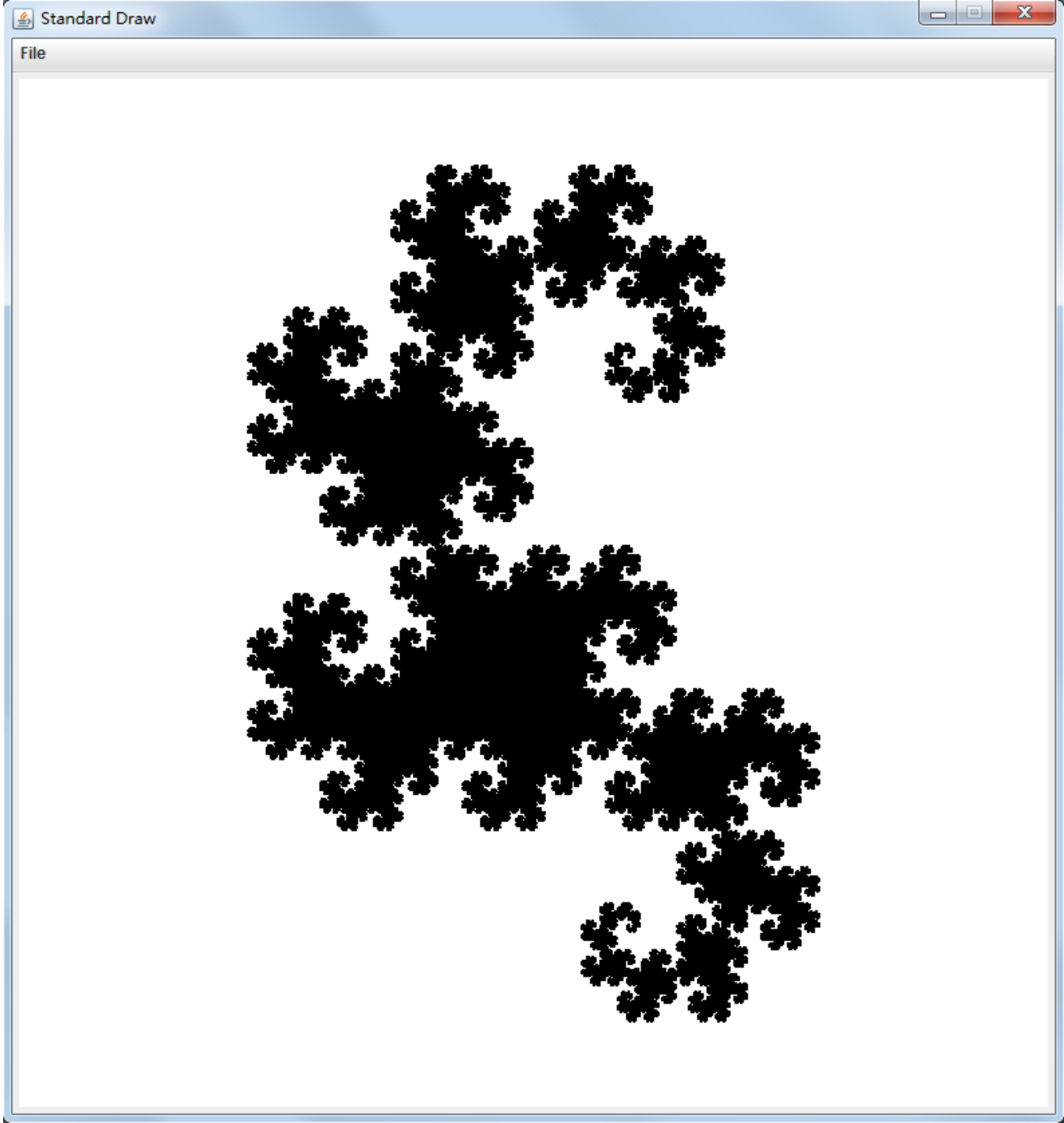
Dragon Curve
Level 18



Dragon Curve
Level 20

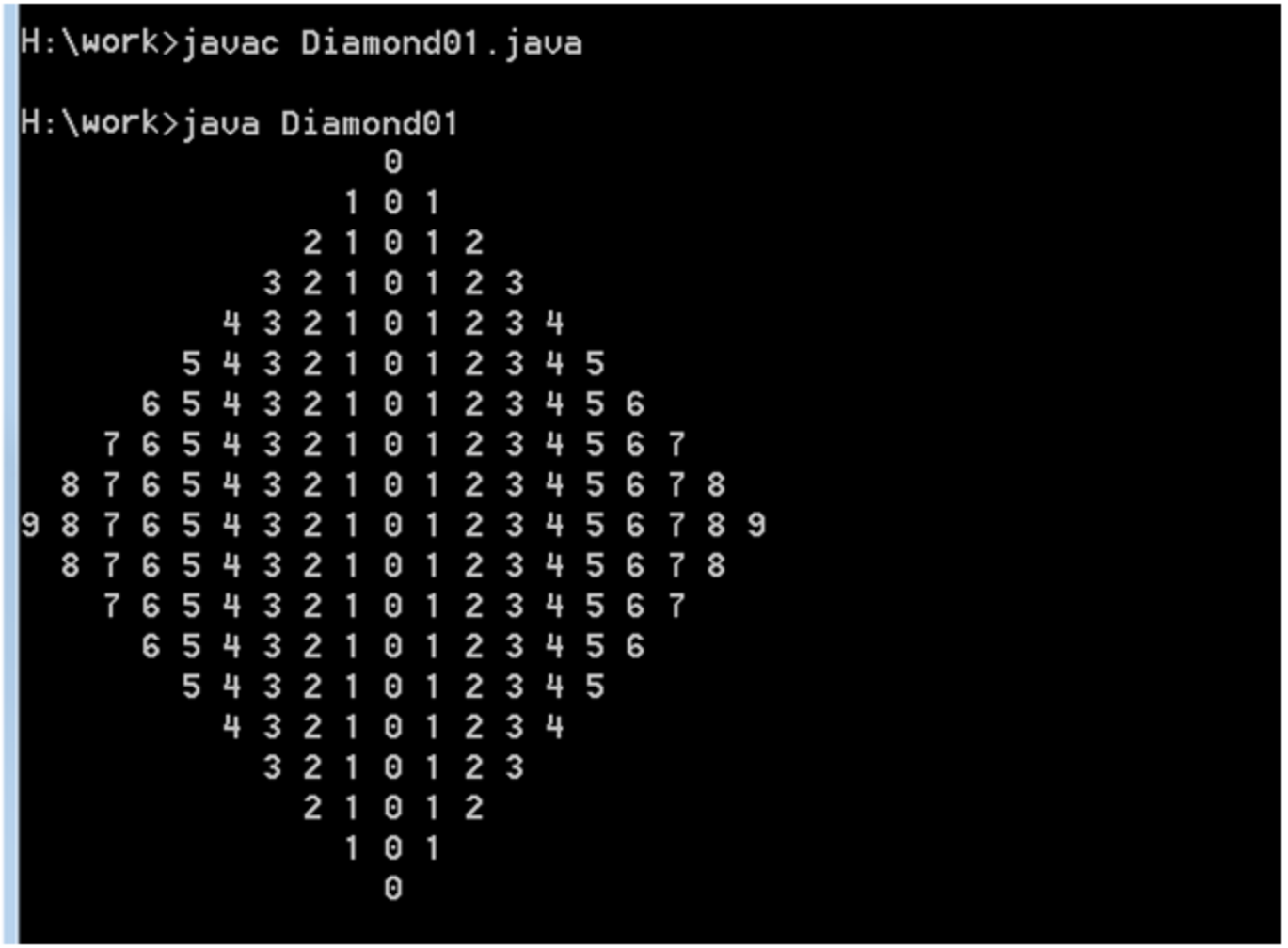


Dragon Curve
Level 22



The easiest way:

KISS: Keep It Simple & Stupid



Diamond01.java

```
public class Diamond01 {    // Diamond01.java
   public static void main (String[] args) {
      System.out.println( "                    0" );
      System.out.println( "                  1 0 1" );
      System.out.println( "                2 1 0 1 2" );
      System.out.println( "              3 2 1 0 1 2 3" );
      System.out.println( "            4 3 2 1 0 1 2 3 4" );
      System.out.println( "          5 4 3 2 1 0 1 2 3 4 5" );
      System.out.println( "        6 5 4 3 2 1 0 1 2 3 4 5 6" );
      System.out.println( "      7 6 5 4 3 2 1 0 1 2 3 4 5 6 7" );
      System.out.println( "    8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8" );
      System.out.println( "9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9" );
      System.out.println( "    8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8" );
      System.out.println( "      7 6 5 4 3 2 1 0 1 2 3 4 5 6 7" );
      System.out.println( "        6 5 4 3 2 1 0 1 2 3 4 5 6" );
      System.out.println( "          5 4 3 2 1 0 1 2 3 4 5" );
      System.out.println( "            4 3 2 1 0 1 2 3 4" );
      System.out.println( "              3 2 1 0 1 2 3" );
      System.out.println( "                2 1 0 1 2" );
      System.out.println( "                  1 0 1" );
      System.out.println( "                    0" );
   }
}
```

19 actions. Too much Duplication!

Diamond02.java

```
1  public class Diamond02 {    // Diamond02.java
2     public static void main (String[] args) {
3        String diamond =
4           "                    0\n" +
5           "                  1 0 1\n" +
6           "                2 1 0 1 2\n" +
7           "              3 2 1 0 1 2 3\n" +
8           "            4 3 2 1 0 1 2 3 4\n" +
9           "          5 4 3 2 1 0 1 2 3 4 5\n" +
10          "        6 5 4 3 2 1 0 1 2 3 4 5 6\n" +
11          "      7 6 5 4 3 2 1 0 1 2 3 4 5 6 7\n" +
12          "    8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8\n" +
13          "9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9\n" +
14          "  8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8\n" +
15          "    7 6 5 4 3 2 1 0 1 2 3 4 5 6 7\n" +
16          "      6 5 4 3 2 1 0 1 2 3 4 5 6\n" +
17          "        5 4 3 2 1 0 1 2 3 4 5\n" +
18          "          4 3 2 1 0 1 2 3 4\n" +
19          "            3 2 1 0 1 2 3\n" +
20          "              2 1 0 1 2\n" +
21          "                1 0 1\n" +
22          "                  0\n" ;
23       System.out.print( diamond );
24    }
25 }
```

1 String
With
Structs.
1 action!

Works,
But
Not
Flexible!

The Patterns:

Each line : Blanks + mirrorDigits + newLine

Top Part [0 .. n] + Bottom Part [n-1 .. 0]

Beyond 0 .. 9

A .. Z : 10 .. 35

Conditional Operator (? :)

```
v = condition ? exp1 : exp2;
```

```
if (condition) v = exp1;
else           v = exp2;
```

Diamond03.java

```java
public class Diamond03 {    // Diamond03.java
    public static void main (String[] args) {
        int N = args.length > 0 ? Integer.parseInt( args[0] ) : 9;
        String diamond = "";
        for (int i = 0; i <= N; i++)
            diamond += blanks( 2 * (N-i) ) + mirrorDigits( i ) + "\n";
        for (int i = N-1; i >= 0; i--)
            diamond += blanks( 2 * (N-i) ) + mirrorDigits( i ) + "\n";
       System.out.print( diamond );
    }
    public static String blanks (int n) {
        String s = "";
        while (n-- > 0) s += " ";
        return s;
    }
    public static final String DIGITS =
        "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"; //constant like Math.PI
    public static String mirrorDigits (int n) {
        String s = "0";
        for (int i = 1; i <= n; i++) {
            char c = DIGITS.charAt(i);
            s = c + " " + s + " " + c;
        }
        return s;
    }
}
```


H:\work>java Diamond03 18
0
1 0 1
2 1 0 1 2
3 2 1 0 1 2 3
4 3 2 1 0 1 2 3 4
5 4 3 2 1 0 1 2 3 4 5
6 5 4 3 2 1 0 1 2 3 4 5 6
7 6 5 4 3 2 1 0 1 2 3 4 5 6 7
8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8
9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9
A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A
B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B
C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C
D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D
E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E
F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F
G F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F G
H G F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F G H
I H G F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F G H I
H G F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F G H
G F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F G
F E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E F
E D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D E
D C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C D
C B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B C
B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B
A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A
9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9
8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8
7 6 5 4 3 2 1 0 1 2 3 4 5 6 7
6 5 4 3 2 1 0 1 2 3 4 5 6
5 4 3 2 1 0 1 2 3 4 5
4 3 2 1 0 1 2 3 4
3 2 1 0 1 2 3
2 1 0 1 2
1 0 1
0

The Patterns:

Each line : Blanks + mirrorDigits + newLine

Top Part [0 .. n] + Bottom Part [n-1 .. 0]

Mapping:

Line: 0 .. 2n

In Top Part [0 .. n], i = line when line <= n;

In Bottom Part [n-1 .. 0], i = 2n – line when line > n.

```
i = line <= n ? line : 2*n - line;
```

Diamond04.java

```java
public class Diamond04 {   // Diamond04.java
   public static void main (String[] args) {
      int N = args.length > 0 ? Integer.parseInt( args[0] ) : 9;
      String diamond = "";
      for (int line = 0; line <= 2*N; line++) {
         int i = line <= N ? line : 2*N - line;
         diamond += blanks( 2*(N-i) ) + mirrorDigits( i ) + "\n";
      }
      System.out.print( diamond );
   }
   public static String blanks (int n) {
      String s = "";
      while (n-- > 0) s += " ";
      return s;
   }
   public static final String DIGITS =
      "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"; //constant like Math.PI
   public static String mirrorDigits (int n) {
      String s = "0";
      for (int i = 1; i <= n; i++) {
         char c = DIGITS.charAt(i);
         s = c + " " + s + " " + c;
      }
      return s;
   }
}
```

```
H:\work>javac Diamond04.java

H:\work>java Diamond04 15
                      0
                    1 0 1
                  2 1 0 1 2
                3 2 1 0 1 2 3
              4 3 2 1 0 1 2 3 4
            5 4 3 2 1 0 1 2 3 4 5
          6 5 4 3 2 1 0 1 2 3 4 5 6
        7 6 5 4 3 2 1 0 1 2 3 4 5 6 7
      8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8
    9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9
  A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A
B A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A B
  A 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 A
    9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9
      8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8
        7 6 5 4 3 2 1 0 1 2 3 4 5 6 7
          6 5 4 3 2 1 0 1 2 3 4 5 6
            5 4 3 2 1 0 1 2 3 4 5
              4 3 2 1 0 1 2 3 4
                3 2 1 0 1 2 3
                  2 1 0 1 2
                    1 0 1
                      0
```

Diamond05.java

```java
// Thanks to Mr. Z for the elegant design.
public class Diamond05 {   // Diamond05.java
   public static void main (String[] args) {
      for (int i = 0; i <= 18; i++)
         for (int j = 0; j <= 18; j++)
            System.out.print(
               Math.abs(j-9) + Math.abs(i-9) <= 9 ?
               Math.abs(j-9) + (j==18 ? "\n" : ".") :
               j==18 ? "\n" : ".."
            );
   }
}
```

```
H:\work>javac Diamond05.java

H:\work>java Diamond05
..................0..................
................1.0.1................
..............2.1.0.1.2..............
............3.2.1.0.1.2.3............
..........4.3.2.1.0.1.2.3.4..........
........5.4.3.2.1.0.1.2.3.4.5........
......6.5.4.3.2.1.0.1.2.3.4.5.6......
....7.6.5.4.3.2.1.0.1.2.3.4.5.6.7....
..8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8..
9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9
..8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8..
....7.6.5.4.3.2.1.0.1.2.3.4.5.6.7....
......6.5.4.3.2.1.0.1.2.3.4.5.6......
........5.4.3.2.1.0.1.2.3.4.5........
..........4.3.2.1.0.1.2.3.4..........
............3.2.1.0.1.2.3............
..............2.1.0.1.2..............
................1.0.1................
..................0..................
```

Diamond051.java

```java
// Thanks to Mr. Z for the elegant design.
public class Diamond051 {   // Diamond051.java
   public static void main (String[] args) {
      for (int i = 0; i <= 18; i++)
         for (int j = 0; j <= 18; j++)
            System.out.print(
               Math.abs(j-9) + Math.abs(i-9) <= 9 ?
               Math.abs(j-9) + (j==18 ? "\n" : ".") :
               (j==18 ? ".\n" : "..")
            );
   }
}
```

```
H:\work>javac Diamond051.java

H:\work>java Diamond051
..................0..................
................1.0.1................
..............2.1.0.1.2..............
............3.2.1.0.1.2.3............
..........4.3.2.1.0.1.2.3.4..........
........5.4.3.2.1.0.1.2.3.4.5........
......6.5.4.3.2.1.0.1.2.3.4.5.6......
....7.6.5.4.3.2.1.0.1.2.3.4.5.6.7....
..8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8..
9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9
..8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8..
....7.6.5.4.3.2.1.0.1.2.3.4.5.6.7....
......6.5.4.3.2.1.0.1.2.3.4.5.6......
........5.4.3.2.1.0.1.2.3.4.5........
..........4.3.2.1.0.1.2.3.4..........
............3.2.1.0.1.2.3............
..............2.1.0.1.2..............
................1.0.1................
..................0..................
```

Diamond052.java

```java
// Thanks to Mr. Z for the elegant design.
public class Diamond052 {
   public static void main (String[] args) {
      for (int i = 0; i <= 18; i++) {
         int L = 9 + (i<=9 ? i : 18-i);
         for (int j = 0; j <= L; j++)
            System.out.print(
               Math.abs(j-9) + Math.abs(i-9) <= 9 ?
               Math.abs(j-9) + (j==L ? "\n" : ".") :
               ".."
            );
      }
   }
}
```

```
H:\work>javac Diamond052.java

H:\work>java Diamond052
..................0
................1.0.1
..............2.1.0.1.2
............3.2.1.0.1.2.3
..........4.3.2.1.0.1.2.3.4
........5.4.3.2.1.0.1.2.3.4.5
......6.5.4.3.2.1.0.1.2.3.4.5.6
....7.6.5.4.3.2.1.0.1.2.3.4.5.6.7
..8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8
9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9
..8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8
....7.6.5.4.3.2.1.0.1.2.3.4.5.6.7
......6.5.4.3.2.1.0.1.2.3.4.5.6
........5.4.3.2.1.0.1.2.3.4.5
..........4.3.2.1.0.1.2.3.4
............3.2.1.0.1.2.3
..............2.1.0.1.2
................1.0.1
..................0
```

Diamond06.java

```java
// Thanks to Mr. Z for the elegant design.
import static java.lang.Math.*;
public class Diamond06 {
   public static void main (String[] args) {
      final String DIGITS = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
      int N = args.length > 0 ? Integer.parseInt( args[0] ) : 9;
      for (int i = 0; i <= 2*N; i++) {
         int L = N + (i<=N ? i : 2*N-i);
         for (int j = 0; j <= L; j++)
            System.out.print(
               abs(j-N) + abs(i-N) <= N ?
               DIGITS.charAt(abs(j-N)) + (j==L ? "\n" : ".") :
               ".."
            );
      }
   }
}
```

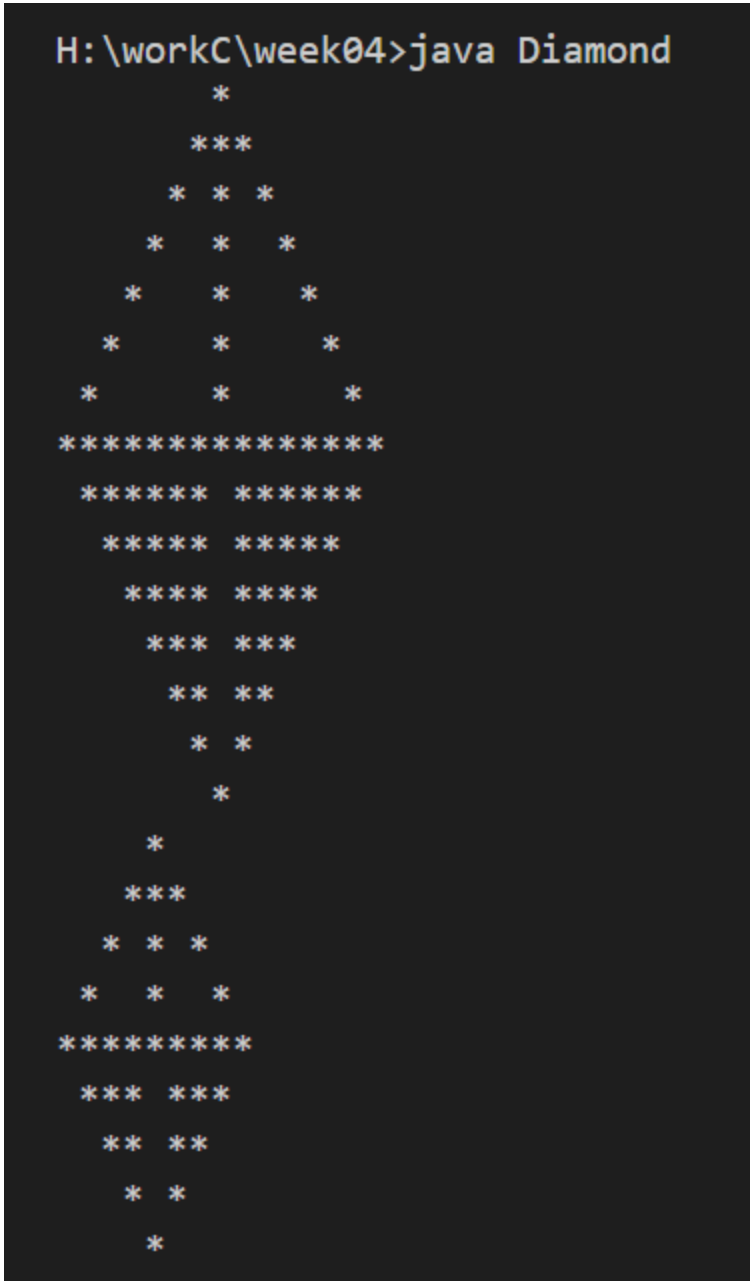
```
H:\work>javac Diamond06.java

H:\work>java Diamond06 15
..............................0
.............................1.0.1
............................2.1.0.1.2
...........................3.2.1.0.1.2.3
..........................4.3.2.1.0.1.2.3.4
.........................5.4.3.2.1.0.1.2.3.4.5
........................6.5.4.3.2.1.0.1.2.3.4.5.6
.......................7.6.5.4.3.2.1.0.1.2.3.4.5.6.7
......................8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8
.....................9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9
....................A.9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9.A
...................B.A.9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9.A.B
.................C.B.A.9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9.A.B.C
................D.C.B.A.9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9.A.B.C.D
..............E.D.C.B.A.9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9.A.B.C.D.E
............F.E.D.C.B.A.9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
..............E.D.C.B.A.9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9.A.B.C.D.E
................D.C.B.A.9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9.A.B.C.D
..............C.B.A.9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9.A.B.C
............B.A.9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9.A.B
..........A.9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9.A
.....................9.8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8.9
......................8.7.6.5.4.3.2.1.0.1.2.3.4.5.6.7.8
.......................7.6.5.4.3.2.1.0.1.2.3.4.5.6.7
........................6.5.4.3.2.1.0.1.2.3.4.5.6
.........................5.4.3.2.1.0.1.2.3.4.5
..........................4.3.2.1.0.1.2.3.4
...........................3.2.1.0.1.2.3
............................2.1.0.1.2
.............................1.0.1
..............................0
```

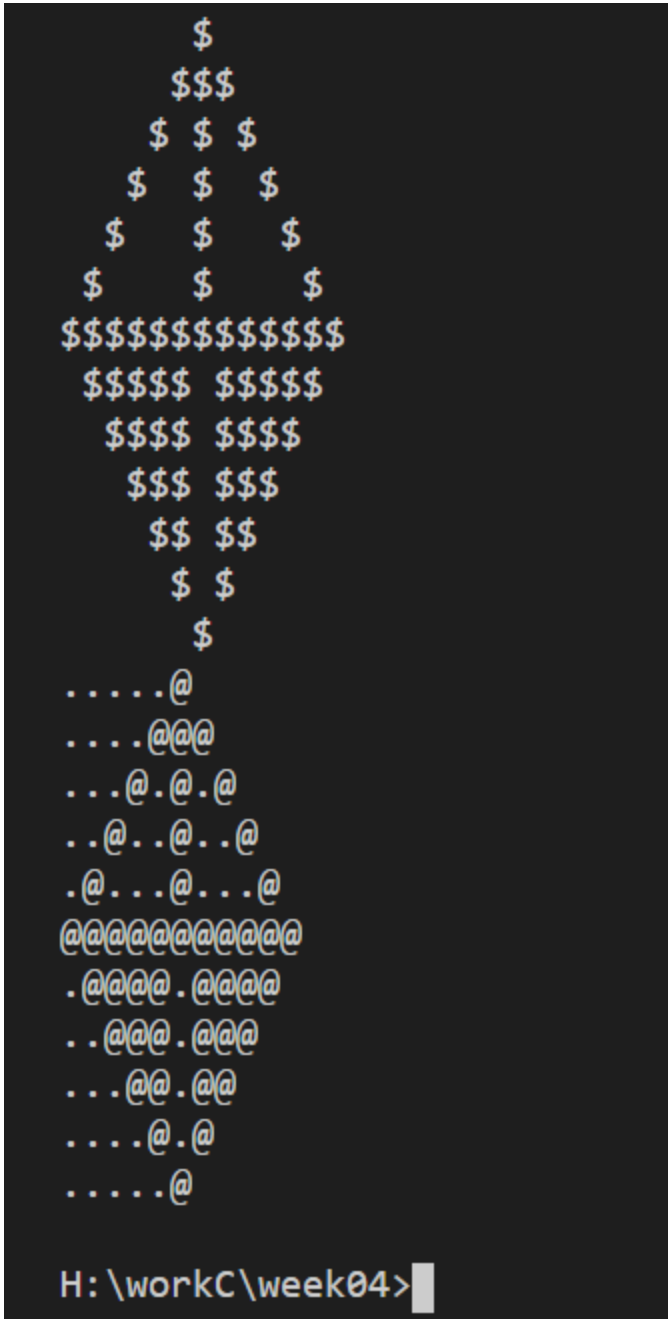
Diamond.java

```
/**
   Diamond.java
   @Max, 2017-03-27, 2018-04-02, 2020-03-22

      ( n, flash, blank )
Diamond( 6,  '*',  '.' )

.....*        <== barln(n-1, 1) = barln(5,1)
....***        ||
...*.*.*       || i = 2..5   [2..n-1]
..*..*..*      || bar(n-i,1) + bar(i-2,1) + barln(i-2,1)
.*...*...*     ||
***********   <== barln(0, 2*n-1) = barln(0,11)
.****.****     ||
..***.***      || i = 5..2   [n-1..2] (down to)
...**.**       || bar(n-i, i-1) + barln(1, i-1)
....*.*        ||
.....*        <== barln(n-1, 1) = barln(5,1)

*/

```

```java

public class Diamond {
    private static char flash, blank;
    public static String diamond (int n, char... color) {
        Diamond.flash = color.length >= 1 ? color[0] : '*';
        Diamond.blank = color.length >= 2 ? color[1] : ' ';
        String image = barln( n-1, 1 );

        for (int i = 2; i < n; i++) {
            image += bar( n-i, 1);
            image += bar( i-2, 1);
            image += barln( i-2, 1);
        }

        image += barln( 0, 2*n - 1 );

        for (int i = n-1; i > 1; i--) {
            image += bar( n-i, i-1 );
            image += barln( 1, i-1 );
        }

        image += barln( n-1, 1 );

        return image;
    }

```

```java

    public static String diamond () {
        return diamond( 8 );
    }

    private static String bar (int nBlank, int nFlash) {
        return repeatChars( nBlank, blank ) + repeatChars( nFlash, flash );
    }

    private static String barln (int nBlank, int nFlash) {
        return bar( nBlank, nFlash ) + '\n';
    }

    private static String repeatChars (int n, char c) {
        String s = "";
        while (n-- > 0) s += c;
        return s;
    }

    public static void main (String[] args) {
        System.out.print( diamond() );
        System.out.print( diamond( 5 ) );
        System.out.print( diamond( 7, '$' ) );
        System.out.print( diamond( 6, '@', '.' ) );
    }
}

```


H:\workC\week04>java Diamond
*
***
* * *
* * *
*   *   *
*   *   *
*   *   *
****
****** ******
***** *****
**** ****
*** ***
** **
* *
*
*
***
* * *
* * *
****
*** ***
** **
* *
*


$
$$$
$ $ $
$ $ $
$ $ $
$ $ $
$$$$$$$$$$$$
$$$$$ $$$$$
$$$$ $$$$
$$$ $$$
$$ $$
$ $
$
.....@
....@@@
...@.@.@
..@..@..@
.@...@...@
@@@@@@@@@@
.@@@@.@@@@
..@@@.@@@
...@@.@@
....@.@
.....@
H:\workC\week04>

Tips:

If `int[][] a = int[N][N]` with elements 0 or 1,

let `int[][] s = int[N][N]`, `s[r][c]` denotes the size of the maximum sub-matrix start from `a[r][c]` towards right-down direction.

Then we could compute `s[r][c]` in the following way (using pseudo-code):

```
s[N-1][N-1 .. 0] = a[N-1][N-1 .. 0]    // last row
s[N-2 .. 0][N-1] = a[N-2 .. 0][N-1]    // last column

for (r from N-2 down to 0)
  for (c from N-2 down to 0)
     s[r][c] = (a[r][c] == 0) ? 0 :
                1 + min(s[r][c+1], s[r+1][c], s[r+1][c+1])
```

LargestBlock.java

```java
public class LargestBlock {
    public static void main (String[] args) {
        int[][] a = {
            { 1, 0, 1, 0, 1 },
            { 1, 1, 1, 0, 1 },
            { 1, 0, 1, 1, 1 },
            { 1, 0, 1, 1, 1 },
            { 1, 0, 1, 1, 1 }
        };
        outputMatrix( a, "a[][]:" );

        int[][] s = findAllBlocks( a );
        outputMatrix( s, "s[][]:" );

        outputLargestBlock( s );
    }

```

LargestBlock.java

```
18    public static void outputLargestBlock (int[][] b) {
19       final int N = b.length;
20       int maxBlockSize = 0;
21       for (int r = 0; r < N; r++)
22          for (int c = 0; c < N; c++)
23             if (maxBlockSize < b[r][c]) maxBlockSize = b[r][c];
24
25       System.out.print( "The maximum square submatrix is at" );
26       for (int r = 0; r < N; r++)
27          for (int c = 0; c < N; c++)
28             if (b[r][c] == maxBlockSize)
29                System.out.printf( " (%d, %d)", r, c );
30       System.out.printf( " with size %d\n", maxBlockSize );
31    }
32
33    public static void outputMatrix (int[][] a, String tips) {
34       System.out.println( tips );
35       for (int r = 0; r < a.length; r++)
36          System.out.println( java.util.Arrays.toString( a[r] ) );
37    }
```

LargestBlock.java

```
38
39      public static int[][] findAllBlocks (int[][] a) {
40          final int N = a.length;
41          int[][] s = new int[N][N];
42          for (int c = N-1; c >= 0; c--)
43              s[N-1][c] = a[N-1][c];    // last row
44          for (int r = N-2; r >= 0; r--)
45              s[r][N-1] = a[r][N-1];    // last column
46
47          for (int r = N-2; r >= 0; r--)
48              for (int c = N-2; c >= 0; c--)
49                  s[r][c] = a[r][c]==0 ? 0 :
50                            1 + min(s[r][c+1], s[r+1][c], s[r+1][c+1]);
51          return s;
52      }
53
54      public static int min (int a, int b, int c) {
55          return Math.min(a, Math.min(b,c));
56      }
57  }
```

```
H:\work>javac LargestBlock.java

H:\work>java LargestBlock
a[][]:
[1, 0, 1, 0, 1]
[1, 1, 1, 0, 1]
[1, 0, 1, 1, 1]
[1, 0, 1, 1, 1]
[1, 0, 1, 1, 1]
s[][]:
[1, 0, 1, 0, 1]
[1, 1, 1, 0, 1]
[1, 0, 3, 2, 1]
[1, 0, 2, 2, 1]
[1, 0, 1, 1, 1]
The maximum square submatrix is at (2, 2) with size 3
```