Solutions for Exercises 03

2.1.1 Write a static method `max3()` that takes three `int` arguments and returns the value of the largest one. Add an overloaded function that does the same thing with three `double` values.

2.1.2 Write a static method `odd()` that takes three `boolean` arguments and returns `true` if an odd number of the argument values are `true`, and `false` otherwise.

2.1.3 Write a static method `majority()` that takes three `boolean` arguments and returns `true` if at least two of the argument values are `true`, and `false` otherwise. Do not use an `if` statement.

2.1.4 Write a static method `eq()` that takes two `int` arrays as arguments and returns `true` if the arrays have the same length and all corresponding pairs of of elements are equal, and `false` otherwise.

2.1.5 Write a static method `areTriangular()` that takes three `double` arguments and returns `true` if they could be the sides of a triangle (none of them is greater than or equal to the sum of the other two). See EXERCISE 1.2.15.

2.1.7 Write a static method `sqrt()` that takes a `double` argument and returns the square root of that number. Use Newton's method (see PROGRAM 1.3.6) to compute the result.

2.1.9 Write a static method `lg()` that takes a `double` argument `n` and returns the base-2 logarithm of `n`. You may use Java's `Math` library.

2.1.10 Write a static method `lg()` that takes an `int` argument `n` and returns the largest integer not larger than the base-2 logarithm of `n`. Do *not* use the `Math` library.

2.1.11 Write a static method `signum()` that takes an `int` argument `n` and returns `-1` if `n` is less than `0`, `0` if `n` is equal to `0`, and `+1` if `n` is greater than `0`.

MyMath.java

```java
import static java.lang.System.out;
import java.util.Arrays;
public class MyMath {
    public static int max (int... numbers) {  //  Varargs
        int max = numbers[0];
        for (int i = 1; i < numbers.length; i++)
            if (max < numbers[i]) max = numbers[i];
        return max;
    }

    public static double max (double... numbers) {  //  Varargs
        double max = numbers[0];
        for (int i = 1; i < numbers.length; i++)
            if (max < numbers[i]) max = numbers[i];
        return max;
    }

    public static int max3 (int a, int b, int c) {   // Exe 2.1.1
        return max( a, b, c );
    }

    public static double max3 (double a, double b, double c) {  // Exe 2.1.1
        return max( a, b, c );
    }
```

MyMath.java

```
25
26    public static boolean odd (boolean p, boolean q, boolean r) {  // Exe 2.1.2
27       return p ^ q ^ r;   // Exclusive OR operator: T^T=F, F^F=F, T^F=T, F^T=T
28    }
29
30    public static boolean majority (boolean p, boolean q, boolean r) { // E 2.1.3
31       return p&&q || q&&r || p&&r;
32    }
33
34    public static boolean eq (int[] a, int[] b) {  // Exe 2.1.4
35       if (a.length != b.length) return false;
36       for (int i = 0; i < a.length; i++)
37          if (a[i] != b[i]) return false;
38       return true;
39    }
40
41    public static boolean areTriangular (double a, double b, double c) { // E 2.1.5
42       return a+b > c && b+c > a && c+a > b;
43    }
44
```

MyMath.java

```java
    public static double sqrt (double a) {  // Exe 2.1.7
        double t = a, t_;
        do {
            t_ = t;
            t = 0.5 * (t_ + a/t_);
        } while (t != t_);
        return t;
    }

    public static double lg (double n) {    // Exe 2.1.9
        return Math.log(n)/Math.log(2.0);
    }

    public static int lg (int n) {    // Exe 2.1.10
        int p = 0;
        while (n > 1) {
            p++;
            n >>= 1;
        }
        return p;
    }

```

MyMath.java

```
67    public static int signum (int n) {    // Exe 2.1.11
68       if (n < 0) return -1;
69       if (n > 0) return 1;
70       return 0;
71    }
72
73    public static void main (String[] args) {
74       out.println( "max(1,2,3,4,3,2,1) = " + max(1,2,3,4,3,2,1) );
75       out.println( "max(1.0,2,3,4,3,2,1) = " + max(1.0,2,3,4,3,2,1) );
76       out.println( "max(.99) = " + max(.99) );
77       out.println( "max3(3,4,5) = " + max3(3,4,5) );
78       out.println( "max3(1.0,8,9) = " + max3(1.0,8,9) );
79
80       final boolean T = true, F = false;
81       out.println( "odd(T,T,T) = " + odd(T,T,T) );
82       out.println( "odd(T,T,F) = " + odd(T,T,F) );
83       out.println( "odd(T,F,T) = " + odd(T,F,T) );
84       out.println( "odd(T,F,F) = " + odd(T,F,F) );
85       out.println( "odd(F,T,T) = " + odd(F,T,T) );
86       out.println( "odd(F,T,F) = " + odd(F,T,F) );
87       out.println( "odd(F,F,T) = " + odd(F,F,T) );
88       out.println( "odd(F,F,F) = " + odd(F,F,F) );
```

MyMath.java

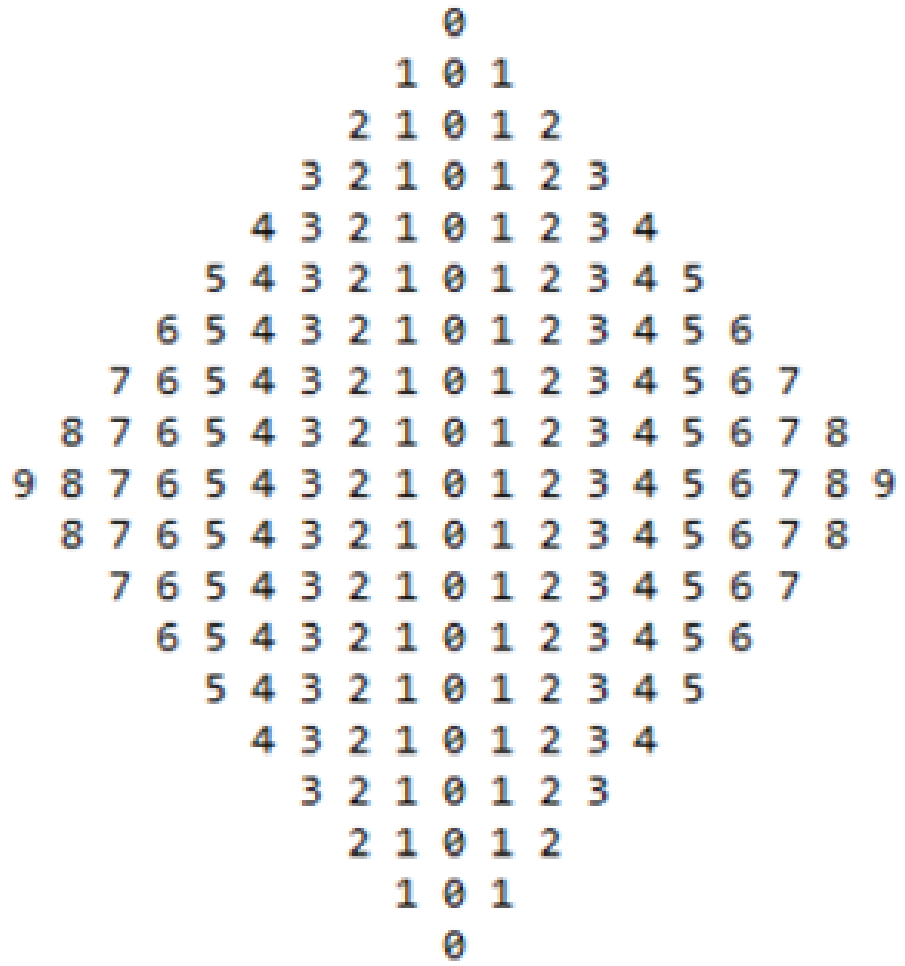
```
89
90      out.println( "majority(T,T,T) = " + majority(T,T,T) );
91      out.println( "majority(T,T,F) = " + majority(T,T,F) );
92      out.println( "majority(T,F,T) = " + majority(T,F,T) );
93      out.println( "majority(T,F,F) = " + majority(T,F,F) );
94      out.println( "majority(F,T,T) = " + majority(F,T,T) );
95      out.println( "majority(F,T,F) = " + majority(F,T,F) );
96      out.println( "majority(F,F,T) = " + majority(F,F,T) );
97      out.println( "majority(F,F,F) = " + majority(F,F,F) );
98
99      int[] a = {1,2};
100     int[] b = {1,2,3};
101     int[] c = {1,2,3};
102     out.println( "a = " + Arrays.toString(a));
103     out.println( "b = " + Arrays.toString(b));
104     out.println( "c = " + Arrays.toString(c));
105     out.println( "eq(a,b) = " + eq(a,b));
106     out.println( "eq(b,c) = " + eq(b,c));
107
```

MyMath.java

```
107
108         out.println( "areTriangular(3,4,5) = " + areTriangular(3,4,5) );
109         out.println( "areTriangular(3,2,5) = " + areTriangular(3,2,5) );
110         out.println( "areTriangular(2,2.5,5) = " + areTriangular(2,2.5,5) );
111
112         out.println( "sqrt(2) = " + sqrt(2) );
113         out.println( "sqrt(3.0) = " + sqrt(3.0) );
114
115         out.println( "lg(100.0) = " + lg(100.0) );
116         out.println( "lg(100) = " + lg(100) );
117
118         out.println( "signum(-100) = " + signum(-100) );
119         out.println( "signum(0) = " + signum(0) );
120         out.println( "signum(5) = " + signum(5) );
121     }
122 }
123
```

```
H:\work\2018A\WarmUp05\Exercises03>javac MyMath.java

H:\work\2018A\WarmUp05\Exercises03>java MyMath
max(1,2,3,4,3,2,1) = 4
max(1.0,2,3,4,3,2,1) = 4.0
max(.99) = 0.99
max3(3,4,5) = 5
max3(1.0,8,9) = 9.0
odd(T,T,T) = true
odd(T,T,F) = false
odd(T,F,T) = false
odd(T,F,F) = true
odd(F,T,T) = false
odd(F,T,F) = true
odd(F,F,T) = true
odd(F,F,F) = false
majority(T,T,T) = true
majority(T,T,F) = true
majority(T,F,T) = true
majority(T,F,F) = false
majority(F,T,T) = true
majority(F,T,F) = false
majority(F,F,T) = false
majority(F,F,F) = false
a = [1, 2]
b = [1, 2, 3]
c = [1, 2, 3]
eq(a,b) = false
eq(b,c) = true
areTriangular(3,4,5) = true
areTriangular(3,2,5) = false
areTriangular(2,2.5,5) = false
sqrt(2) = 1.414213562373095
sqrt(3.0) = 1.732050807568772
lg(100.0) = 6.643856189774725
lg(100) = 6
signum(-100) = -1
signum(0) = 0
signum(5) = 1
```

6. Write a java program with output



in the Console.

The easiest way:

KISS: Keep It Simple & Stupid

```
H:\work\JavaProg\2018Spring\WarmUp04>javac Diamond01.java

H:\work\JavaProg\2018Spring\WarmUp04>java Diamond01
                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
```

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\JavaProg\2018Spring\WarmUp04>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\JavaProg\2018Spring\WarmUp04>javac Diamond04.java

H:\work\JavaProg\2018Spring\WarmUp04>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
  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
```

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\2018A\WarmUp05\Exercises03>javac Diamond05.java

H:\work\2018A\WarmUp05\Exercises03>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\2018A\WarmUp05\Exercises03>javac Diamond051.java

H:\work\2018A\WarmUp05\Exercises03>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

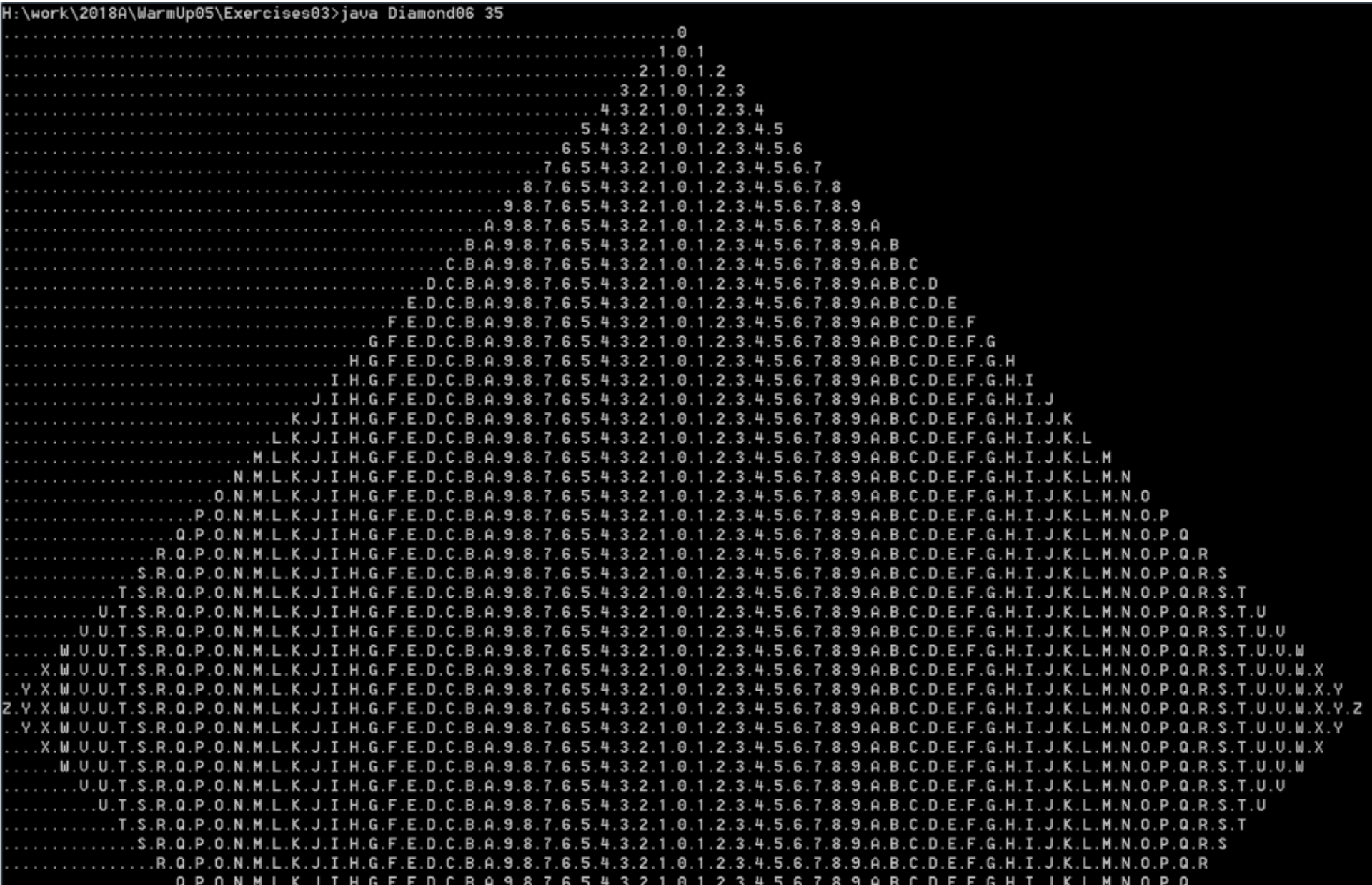
```java
// Thanks to Mr. Z for the elegant design.
public class Diamond052 {
   public static void main (String[] args) {
      for (int i = 0; i <= 18; i++) {
         int L = 9 + (i<=9 ? i : 18-i);
         for (int j = 0; j <= L; j++)
            System.out.print(
               Math.abs(j-9) + Math.abs(i-9)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

```
H:\work\2018A\WarmUp05\Exercises03>javac Diamond052.java

H:\work\2018A\WarmUp05\Exercises03>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

H:\work\2018A\WarmUp05\Exercises03>
```

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\2018A\WarmUp05\Exercises03>javac Diamond06.java

H:\work\2018A\WarmUp05\Exercises03>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
```


H:\work\2018A\WarmUp05\Exercises03>java Diamond06 35
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
J.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.J
K.J.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.J.K
L.K.J.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.J.K.L
M.L.K.J.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.J.K.L.M
N.M.L.K.J.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.J.K.L.M.N
O.N.M.L.K.J.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.J.K.L.M.N.O
P.O.N.M.L.K.J.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.J.K.L.M.N.O.P
Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q
R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R
S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S
T.S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S.T
U.T.S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S.T.U
V.U.T.S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S.T.U.V
W.V.U.T.S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S.T.U.V.W
X.W.V.U.T.S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S.T.U.V.W.X
Y.X.W.V.U.T.S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S.T.U.V.W.X.Y
Z.Y.X.W.V.U.T.S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S.T.U.V.W.X.Y.Z
Y.X.W.V.U.T.S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S.T.U.V.W.X.Y
X.W.V.U.T.S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S.T.U.V.W.X
W.V.U.T.S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S.T.U.V.W
V.U.T.S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S.T.U.V
U.T.S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S.T.U
T.S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S.T
S.R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R.S
R.Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q.R
Q.P.O.N.M.L.K.J.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.J.K.L.M.N.O.P.Q

Criteria for Good Programs:

1) Run Effectively (Correctly）& Efficiently (Objectives 目的)
2) Easy to be Extended and Modified (Approaches 手段)
3) Easy to be Understood (Pre-conditions 前提)