```java
/* Recieves two command line integers, n and k, and returns the respective binomial coefficent.
   Uses memoization to optimize the recursive process. */
public class Binomial {

    public static void main(String[] args) {
        System.out.println(binomial(Integer.parseInt(args[0]), Integer.parseInt(args[1])));
    }

    // Computes and returns the Binomial coefficient
    public static long binomial(int n, int k) {
        long[][] memo = new long[n + 1][k + 1];

        if (k > n){
            return 0;
        }
        if (k == 0 || n == 0) {
            return 1;
        }

        return (binomial(n - 1, k, memo) + binomial(n - 1, k - 1, memo));
    }

    public static long binomial(int n, int k, long[][] memo) {
        if (k > n){
            return 0;
        }

        if (k == 0 || n == 0) {
            return 1;
        }

        if (memo[n][k] == 0) {
            memo[n][k] = binomial(n - 1, k, memo) + binomial(n - 1, k - 1, memo);
        }

        return memo[n][k];
    }
}
```

```java
/* Features a function that prints the decimal value of a given integer value. */
public class IntegerToBinary {

    public static void main(String[] args) {
        integerToBinary(Integer.parseInt(args[0]));
        System.out.println("");
    }

    public static void integerToBinary(int n) {
        if (n == 0 || n == 1) {
            System.out.print((int)n);
        } else {
            integerToBinary(n / 2);
            System.out.print(n % 2);
        }
    }
}
```

```java
/** Reads a command line string and checks if it's a palindrome. */
public class Palindrome {

    public static void main(String[]args) {
        System.out.println(isPalindrome(args[0]));
    }

    public static boolean isPalindrome(String s) {
        int n = s.length();
        if (n == 0 || n == 1) {
            return true;
        } else {
            if (s.charAt(0) == s.charAt(n - 1)) {
                return isPalindrome(s.substring(1, n - 1));
            }
        }

        return false;
    }
}
```

```java
/** Prints ths Sierpinski Triangle fractal. */
public class Sierpinski {

    public static void main(String[] args) {
        sierpinski(Integer.parseInt(args[0]));
    }

    // Draws a Sierpinski triangle of depth n on the standard canvass.
    public static void sierpinski (int n) {
        double s = Math.sqrt(3) / 2;
        // first triangle
        StdDraw.line(0, 0, 0.5 ,s);
        StdDraw.line(0.5, s, 1, 0);
        StdDraw.line(1, 0, 0 ,0);

                //  n  X1 X2  X3  Y1 Y2 Y3
        sierpinski (n, 0, 1, 0.5 ,0, 0, s);
    }

    public static void sierpinski(int n, double x1, double x2, double x3,
,                                         double y1, double y2, double y3) {
        // end of drawing rounds
        if (n == 0) {
            return;
        }

        // second triangle
        // left -> middle
        StdDraw.line((x1 + x3) / 2, (y1 + y3) / 2, (x2 + x3) / 2, (y2 + y3) / 2);
        // middle -> right
        StdDraw.line((x2 + x3) / 2, (y2 + y3) / 2, (x1 + x2) / 2, (y1 + y2) / 2);
        // right -> left
        StdDraw.line((x1 + x2) / 2, (y1 + y2) / 2, (x1 + x3) / 2, (y1 + y3) / 2);

        sierpinski(n - 1, x1, (x1 + x2) / 2, (x1 + x3) / 2, y1, (y1 + y2) / 2, (y1 + y3) / 2);
        sierpinski(n - 1, (x1 + x2) / 2, x2, (x3 + x2) / 2, (y1 + y2) / 2, y2, (y3 + y2) / 2);
        sierpinski(n - 1, (x1 + x3) / 2, (x3 + x2) / 2, x3, (y1 + y3) / 2, (y3 + y2) / 2, y3);
    }
}
```