```
package player.playeragent;

//-------------------------------------
// Systematically generate combinations.
// This code is courtesy of Michael Gilleland
// http://www.merriampark.com/comb.htm#Source
//-------------------------------------

import java.math.BigInteger;

public class CombinationGenerator {

        private int[] a;
        private int n;
        private int r;
        private BigInteger numLeft;
        private BigInteger total;

        //------------
        // Constructor
        //------------

        public CombinationGenerator (int n, int r) {
                if (r > n) {
                        throw new IllegalArgumentException ();
                }
                if (n < 1) {
                        throw new IllegalArgumentException ();
                }
                this.n = n;
                this.r = r;
                a = new int[r];
                BigInteger nFact = getFactorial (n);
                BigInteger rFact = getFactorial (r);
                BigInteger nminusrFact = getFactorial (n - r);
                total = nFact.divide (rFact.multiply (nminusrFact));
                reset ();
        }

        //------
        // Reset
        //------

        public void reset () {
                for (int i = 0; i < a.length; i++) {
                        a[i] = i;
                }
                numLeft = new BigInteger (total.toString ());
        }

        //-----------------------------------------------
        // Return number of combinations not yet generated
        //-----------------------------------------------

        public BigInteger getNumLeft () {
                return numLeft;
        }

        //-----------------------------
        // Are there more combinations?
        //-----------------------------

        public boolean hasMore () {
                return numLeft.compareTo (BigInteger.ZERO) == 1;
        }

        //------------------------------------
        // Return total number of combinations
        //------------------------------------

        public BigInteger getTotal () {
                return total;
        }

        //------------------
        // Compute factorial
        //------------------

        private static BigInteger getFactorial (int n) {
                BigInteger fact = BigInteger.ONE;
                for (int i = n; i > 1; i--) {
```

```
                        fact = fact.multiply (new BigInteger (Integer.toString (i)));
                }
                return fact;
        }

        //-------------------------------------------------------
        // Generate next combination (algorithm from Rosen p. 286)
        //-------------------------------------------------------

        public int[] getNext () {

                if (numLeft.equals (total)) {
                        numLeft = numLeft.subtract (BigInteger.ONE);
                        return a;
                }

                int i = r - 1;
                while (a[i] == n - r + i) {
                        i--;
                }
                a[i] = a[i] + 1;
                for (int j = i + 1; j < r; j++) {
                        a[j] = a[i] + j - i;
                }

                numLeft = numLeft.subtract (BigInteger.ONE);
                return a;

        }
}
```