```java
package player;

import java.util.HashSet;

import edu.neu.ccs.demeterf.demfgen.lib.List;
import gen.Variable;

public class Permutations {
    private List<Variable> vars;

    /**
     * Generates the permutations for the given list of variables.
     *
     * @param vars
     *            The list of variables.
     */
    public Permutations(List<Variable> vars) {
        this.vars = vars;
    }

    /**
     * Generate the combination of variables.
     *
     * @param vars
     *            The list of variables.
     */
    public HashSet<List<Variable>> combinations() {
        HashSet<List<Variable>> result = new HashSet<List<Variable>>();
        Combinations comb = new Combinations(vars.length(), 3);

        int[] indices;

        while (comb.hasMore()) {
            indices = comb.getNext();

            List<Variable> list = List.<Variable> create();

            for (int i = 0; i < indices.length; i++) {
                list = list.append(vars.lookup(indices[i]));
            }

            result.add(list);
        }

        return result;
    }

    /**
     * Generate the permutation of 3 variables based on the given combination.
     *
     * @param comb
     *            The list of combinations.
     */
    public HashSet<List<Variable>> permutations(HashSet<List<Variable>> comb) {
        HashSet<List<Variable>> result = new HashSet<List<Variable>>();

        List<Variable> round;

        for (List<Variable> l : comb) {
            // Identity
            round = List.<Variable> create(l.lookup(0), l.lookup(1), l
                    .lookup(2));
            result.add(round);

            // Transpose 1, 2
            round = List.<Variable> create(l.lookup(0), l.lookup(2), l
                    .lookup(1));
            result.add(round);

            // Transpose 0, 2
            round = List.<Variable> create(l.lookup(2), l.lookup(1), l
                    .lookup(0));
            result.add(round);

            // Transpose 0, 1
            round = List.<Variable> create(l.lookup(1), l.lookup(0), l
                    .lookup(2));
            result.add(round);

            // Rotate right
            round = List.<Variable> create(l.lookup(2), l.lookup(0), l
                .lookup(1));
        result.add(round);

        // Rotate right
        round = List.<Variable> create(l.lookup(1), l.lookup(2), l
                .lookup(0));
        result.add(round);
    }

    return result;
  }
}
```