```java
import components.statement.Statement;

/**
 * Utility class with method to count the number of calls to primitive
 * instructions (move, turnleft, turnright, infect, skip) in a given
 * {@code Statement}.
 *
 * @author Put your name here
 *
 */
public final class CountPrimitiveCalls {

    /**
     * Private constructor so this utility class cannot be instantiated.
     */
    private CountPrimitiveCalls() {
    }

    /**
     * Reports the number of calls to primitive instructions (move, turnleft,
     * turnright, infect, skip) in a given {@code Statement}.
     *
     * @param s
     *           the {@code Statement}
     * @return the number of calls to primitive instructions in {@code s}
     * @ensures <pre>
     * countOfPrimitiveCalls =
     *  [number of calls to primitive instructions in s]
     * </pre>
     */
    public static int countOfPrimitiveCalls(Statement s) {
        int count = 0;
        switch (s.kind()) {
            case BLOCK: {
                /*
                 * Add up the number of calls to primitive instructions
                 * in each nested statement in the BLOCK.
                 */

                for (int i=0; i < s.lengthOfBlock(); i++) {
                    count += countOfPrimitiveCalls(s.removeFromBlock(i));
                }

                break;
            }
            case IF: {
                /*
                 * Find the number of calls to primitive instructions in
                 * the body of the IF.
                 */

                for (int i=0; i < s.lengthOfBlock(); i++) {
                    count += countOfPrimitiveCalls(s.removeFromBlock(i));
                }
                break;
            }
            case IF_ELSE: {
                /*
                 * Add up the number of calls to primitive instructions in
                 * the "then" and "else" bodies of the IF_ELSE.
                 */

                for (int i=0; i < s.lengthOfBlock(); i++) {
                    count += countOfPrimitiveCalls(s.removeFromBlock(i));
                }

                break;
            }
            case WHILE: {
                /*
                 * Find the number of calls to primitive instructions in
                 * the body of the WHILE.
                 */

                for (int i=0; i < s.lengthOfBlock(); i++) {
                    count += countOfPrimitiveCalls(s.removeFromBlock(i));
                }

                break;
            }
            case CALL: {
                /*
                 * This is a leaf: the count can only be 1 or 0. Determine
                 * whether this is a call to a primitive instruction or not.
                 */


                // For fucks sake i've felt too sick to go to classes or labs for ANY of my courses
                // so i have no idea what I'm meant to be doing and I'm drowning in work that I
                // don't even understand

                if (s.getClass().equals()) {
                    count++;
                }

                break;
            }
            default: {
                // this will never happen...can you explain why?

                // because all possible results are already addressed

                break;
            }
        }
        return count;
    }

}
```