```java
import components.simplereader.SimpleReader;

/**
 * Homework 22
 *
 * @author Sam Espanioly
 *
 */
public final class HW22 {

    /**
     * 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.
                 */

                // TODO - fill in case
                s.addToBlock(count, s);

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

                // TODO - fill in case
                s.notify();

                break;
            }
            case IF_ELSE: {
                /*
                 * Add up the number of calls to primitive instructions in the
                 * "then" and "else" bodies of the IF_ELSE.
                 */

                // TODO - fill in case

                break;
            }
```

```
62              case WHILE: {
63                  /*
64                   * Find the number of calls to primitive instructions in the
65                   * body of the WHILE.
66                   */
67
68                  // TODO - fill in case
69
70                  break;
71              }
72              case CALL: {
73                  /*
74                   * This is a leaf: the count can only be 1 or 0. Determine
75                   * whether this is a call to a primitive instruction or not.
76                   */
77
78                  // TODO - fill in case
79
80                  break;
81              }
82              default: {
83                  // this will never happen...can you explain why?
84                  break;
85              }
86          }
87          return count;
88          // NOT FINISHED BECAUSE I DID NOT UNDERSTAND MATERIALS FROM SLIDES (yelling on the
    side)
89      }
90
91  //
92  //      IF next-is-empty THEN
93  //          move
94  //      ELSE
95  //          IF next-is-wall THEN
96  //              turnright
97  //              turnright
98  //              move
99  //          END IF
100 //      END IF
101 //
102 //      WHILE true DO
103 //          turnright
104 //          IF next-is-enemy THEN
105 //              TurnAround
106 //          ELSE
107 //              skip
108 //          END IF
109 //          turnleft
110 //      END WHILE
111 //
112 //      WHILE next-is-enemy DO
113 //          infect
114 //          TurnAround
115 //          move
116 //          turnright
117 //      END WHILE
```

```
118 //
119 //    IF next-is-friend THEN
120 //        turnright
121 //        turnright
122 //        WHILE true DO
123 //            infect
124 //        END WHILE
125 //    END IF
126 //
127 //    IF next-is-not-empty THEN
128 //        turnleft
129 //        turnleft
130 //    ELSE
131 //        WHILE next-is-empty DO
132 //            move
133 //        END WHILE
134 //        IF next-is-enemy THEN
135 //            infect
136 //        END IF
137 //        skip
138 //    END IF
139
140 //I need help with this, I missed class :[
141
142     /**
143      * Main method.
144      *
145      * @param args
146      *             the command line arguments
147      */
148     public static void main String[] args) {
149         SimpleReader in = new SimpleReader1L();
150         SimpleWriter out = new SimpleWriter1L();
151         /*
152          * Put your main program code here
153          */
154         /*
155          * Close input and output streams
156          */
157         in.close();
158         out.close();
159     }
160
161 }
162
```