```java
import components.queue.Queue;

/**
 * {@code Queue} represented as a {@code Sequence} of entries, with
 * implementations of primary methods.
 *
 * @param <T>
 *            type of {@code Queue} entries
 * @correspondence this = $this.entries
 */
public class HelloWorld {

    public static void main(String[] args) {

    }

    /**
     * Refactors the given {@code Statement} so that every IF_ELSE statement
     * with a negated condition (NEXT_IS_NOT_EMPTY, NEXT_IS_NOT_ENEMY,
     * NEXT_IS_NOT_FRIEND, NEXT_IS_NOT_WALL) is replaced by an equivalent
     * IF_ELSE with the opposite condition and the "then" and "else" BLOCKs
     * switched. Every other statement is left unmodified.
     *
     * @param s
     *            the {@code Statement}
     * @updates s
     * @ensures <pre>
     * s = [#s refactored so that IF_ELSE statements with "not"
     *   conditions are simplified so the "not" is removed]
     * </pre>
     */
    public static void simplifyIfElse(Statement s) {
        switch (s.kind()) {
            case BLOCK: {

                int length = s.lengthOfBlock();
                for (int i = 0; i < length; i++) {
                    Statement child = s.removeFromBlock(i);
                    simplifyIfElse(child);
```

```
42                      s.addToBlock(i, child);
43                  }
44
45                  break;
46              }
47              case IF: {
48
49                  Statement child = s.newInstance();
50                  Statement.Condition condition =
    s.disassembleIf(child);
51                  simplifyIfElse(child);
52                  s.assembleIf(condition, child);
53
54                  break;
55              }
56              case IF_ELSE: {
57
58                  Statement childIf = s.newInstance();
59                  Statement childElse = s.newInstance();
60                  Statement.Condition condition =
    s.disassembleIfElse(childIf,
61                          childElse);
62                  switch (condition.name()) {
63                      case "NEXT_IS_NOT_EMPTY": {
64                          condition = condition.NEXT_IS_EMPTY;
65                          simplifyIfElse(childIf);
66                          simplifyIfElse(childElse);
67                          s.assembleIfElse(condition, childElse,
    childIf);
68                      }
69
70                      case "NEXT_IS_NOT_ENEMY": {
71                          condition = condition.NEXT_IS_ENEMY;
72                          simplifyIfElse(childIf);
73                          simplifyIfElse(childElse);
74                          s.assembleIfElse(condition, childElse,
    childIf);
75                          break;
76
77                      }
78                      case "NEXT_IS_NOT_FRIEND": {
79                          condition = condition.NEXT_IS_FRIEND;
80                          simplifyIfElse(childIf);
81                          simplifyIfElse(childElse);
```

```
 82                          s.assembleIfElse(condition, childElse,
    childIf);
 83                          break;
 84
 85                      }
 86                      case "NEXT_IS_NOT_WALL": {
 87                          condition = condition.NEXT_IS_WALL;
 88                          simplifyIfElse(childIf);
 89                          simplifyIfElse(childElse);
 90                          s.assembleIfElse(condition, childElse,
    childIf);
 91                          break;
 92
 93                      }
 94                  }
 95
 96                  break;
 97              }
 98              case WHILE: {
 99
100                  Statement child = s.newInstance();
101                  Statement.Condition condition =
    s.disassembleWhile(child);
102                  simplifyIfElse(child);
103                  s.assembleWhile(condition, child);
104
105                  break;
106              }
107              case CALL: {
108                  // nothing to do here...can you explain why?
109                  break;
110              }
111              default: {
112                  // this will never happen...can you explain why?
113                  break;
114              }
115          }
116      }
117  }
```