```java
 1  package org.sdu.dk.frhou18;
 2
 3  import org.sdu.dk.frhou18.metamodel.*;
 4
 5  import java.util.HashMap;
 6  import java.util.Map;
 7
 8  public class StateMachine {
 9      public StateMachine() {
10
11      }
12      public Machine machine = new Machine();
13      public Transition currentTransition = null;
14      public State currentState = null;
15
16      public Machine build() {
17          if (machine.initialState == null){
18              machine.initialState = machine.states.get(0);
19          }
20          machine.currentState = machine.initialState;
21          return machine;
22      }
23
24      public StateMachine state(String string) {
25          currentState = new State(string);
26          machine.states.add(currentState);
27          return this;
28      }
29
30      public StateMachine initial() {
31          machine.initialState = currentState;
32          return this;
33      }
34
35      public StateMachine when(String string) {
36          currentTransition = new Transition(string);
37          currentState.transitions.add(currentTransition);
38          return this;
39      }
40
41      public StateMachine to(String string) {
42          currentTransition.targetState = new State(string);
43          return this;
44      }
45
46      public StateMachine integer(String string) {
47          machine.integers.put(string, 0);
48          return this;
```

```java
49          }
50
51      public StateMachine set(String string, int i) {
52          currentTransition.hasSetOperation = true;
53          currentTransition.setIntOperationValue = i;
54          currentTransition.operandVariableName = string;
55          return this;
56      }
57
58      public StateMachine increment(String string) {
59          currentTransition.hasIncrementOption = true;
60          currentTransition.operandVariableName = string;
61          return this;
62      }
63
64      public StateMachine decrement(String string) {
65          currentTransition.hasDecrementOption = true;
66          currentTransition.operandVariableName = string;
67          return this;
68      }
69
70      public StateMachine ifEquals(String string, int i) {
71          currentTransition.conditionalVariableName = string;
72          currentTransition.conditional = Transition.conditionals.EQUAL;
73          currentTransition.conditionalCompareValue = i;
74          return this;
75      }
76
77      public StateMachine ifGreaterThan(String string, int i) {
78          currentTransition.conditionalVariableName = string;
79          currentTransition.conditional = Transition.conditionals.GREATER;
80          currentTransition.conditionalCompareValue = i;
81          return this;
82      }
83
84      public StateMachine ifLessThan(String string, int i) {
85          currentTransition.conditionalVariableName = string;
86          currentTransition.conditional = Transition.conditionals.LESS;
87          currentTransition.conditionalCompareValue = i;
88          return this;
89      }
90
91 }
92
```

```java
1  package org.sdu.dk.frhou18;
2
3  import org.sdu.dk.frhou18.metamodel.Machine;
4  import org.sdu.dk.frhou18.metamodel.State;
5  import org.sdu.dk.frhou18.metamodel.Transition;
6
7  import java.util.List;
8  import java.util.NoSuchElementException;
9  import java.util.stream.Collectors;
10
11  public class MachineInterpreter {
12      Machine machine;
13
14      public void run(Machine m) {
15          this.machine = m;
16      }
17
18      public State getCurrentState() {
19          return machine.currentState;
20      }
21
22      public void processEvent(String string) {
23          List<Transition> transitions;
24          try {
25              transitions = machine.currentState.transitions.stream()
26                      .filter(transition -> transition.getEvent().equals(string)).
  collect(Collectors.toList());
27          }catch (NoSuchElementException e){
28              return;
29          }
30          for (var transition : transitions) {
31              if (transition.isConditional()) {
32                  var conditional = machine.integers.get(transition.
  conditionalVariableName);
33                  switch (transition.conditional) {
34                      case EQUAL:
35                          if (conditional != transition.conditionalCompareValue)
  continue;
36                          break;
37                      case GREATER:
38                          if (conditional <= transition.conditionalCompareValue)
  continue;
39                          break;
40                      case LESS:
41                          if (conditional >= transition.conditionalCompareValue)
  continue;
42                          break;
43                      default:
```

```
44                    continue;
45                }
46            }
47            if (transition.hasSetOperation()) {
48                machine.integers.put(transition.operandVariableName, transition.
   setIntOperationValue);
49            } else if (transition.hasIncrementOperation()) {
50                machine.integers.put(transition.operandVariableName, machine.
   integers.get(transition.operandVariableName) + 1);
51            } else if (transition.hasDecrementOperation()) {
52                machine.integers.put(transition.operandVariableName, machine.
   integers.get(transition.operandVariableName) - 1);
53            }
54
55            machine.currentState = machine.states.stream().filter(state -> state.
   getName().equals(transition.targetState.getName())).findFirst().get();
56            return;
57        }
58    }
59
60    public int getInteger(String string) {
61        return machine.integers.get(string);
62    }
63
64 }
65
```

```java
 1 package org.sdu.dk.frhou18.metamodel;
 2
 3 import java.util.ArrayList;
 4 import java.util.List;
 5
 6 public class State {
 7
 8     public String name;
 9     public List<Transition> transitions;
10
11     public State(String name) {
12         this.name = name;
13         this.transitions = new ArrayList<>();
14     }
15
16     public State() {
17     }
18
19     public String getName() {
20         return name;
21     }
22
23     public List<Transition> getTransitions() {
24         return transitions;
25     }
26
27     public Transition getTransitionByEvent(String string) {
28         return transitions.stream().filter(t -> t.getEvent().equals(string)).
   findFirst().get();
29     }
30
31 }
32
```

```java
1  package org.sdu.dk.frhou18.metamodel;
2
3  import java.util.ArrayList;
4  import java.util.HashMap;
5  import java.util.List;
6  import java.util.Map;
7
8  public class Machine {
9
10     public List<State> states = new ArrayList<>();
11     public State initialState;
12     public Map<String, Integer> integers = new HashMap<>();
13     public State currentState;
14
15
16     public List<State> getStates() {
17         return states;
18     }
19
20     public State getInitialState() {
21         return initialState;
22     }
23
24     public State getState(String string) {
25         return states.stream().filter(state -> state.getName().equals(string)).
   findFirst().get();
26     }
27
28     public int numberOfIntegers() {
29         return this.integers.keySet().size();
30     }
31
32     public boolean hasInteger(String string) {
33         return this.integers.containsKey(string);
34     }
35
36 }
37
```

```java
1  package org.sdu.dk.frhou18.metamodel;
2
3  public class Transition {
4
5      public String transitionEvent;
6      public State targetState;
7      public boolean hasIncrementOption;
8      public boolean hasDecrementOption;
9      public String conditionalVariableName = null;
10     public boolean hasSetOperation;
11     public int setIntOperationValue;
12     public String operandVariableName = null;
13     public enum conditionals {GREATER, LESS, EQUAL}
14     public conditionals conditional = null;
15     public int conditionalCompareValue;
16
17     public Transition() {
18     }
19
20     public Transition(String transitionEvent) {
21         this.transitionEvent = transitionEvent;
22     }
23
24     public String getEvent() {
25         return transitionEvent;
26     }
27
28     public State getTarget() {
29         return targetState;
30     }
31
32     public boolean hasSetOperation() {
33         return hasSetOperation;
34     }
35
36     public boolean hasIncrementOperation() {
37         return hasIncrementOption;
38     }
39
40     public boolean hasDecrementOperation() {
41         return hasDecrementOption;
42     }
43
44     public String getOperationVariableName() {
45         return operandVariableName;
46     }
47
48     public boolean isConditional() {
```

```
49          return conditional != null;
50      }
51
52      public String getConditionVariableName() {
53          return conditionalVariableName;
54      }
55
56      public Integer getConditionComparedValue() {
57          return conditionalCompareValue;
58      }
59
60      public boolean isConditionEqual() {
61          return conditional.equals(conditionals.EQUAL);
62      }
63
64      public boolean isConditionGreaterThan() {
65          return conditional.equals(conditionals.GREATER);
66      }
67
68      public boolean isConditionLessThan() {
69          return conditional.equals(conditionals.LESS);
70      }
71
72      public boolean hasOperation() {
73          return hasDecrementOption || hasIncrementOption;
74      }
75 }
76
```