Frederik Alexander Hounsvad Model Driven Software Development 2022-02-21
frhou18@student.sdu.dk Assignment 1

I went about solving the exercise by first copying the source code into a new project created in a standardized manner instead of trying to fight with eclipse. After this I went about creating variables for which there seemed to exist getter functions. After the getter functions were implemented, I started to look at the code in the test cases, and used the way these were written as instructions for how to implement the functions. Once I had implemented the classes I ran the tests and fixed the few errors that resulted in failed tests.

All unit tests were passed.

https://github.com/Hounsvad/Sem8/tree/master/ModelDriven/assignment1

```java
package org.sdu.dk.frhou18;

import org.sdu.dk.frhou18.metamodel.*;

import java.util.HashMap;
import java.util.Map;

public class StateMachine {
    public StateMachine() {

    }
    public Machine machine = new Machine();
    public Transition currentTransition = null;
    public State currentState = null;

    public Machine build() {
        if (machine.initialState == null){
            machine.initialState = machine.states.get(0);
        }
        machine.currentState = machine.initialState;
        return machine;
    }

    public StateMachine state(String string) {
        currentState = new State(string);
        machine.states.add(currentState);
        return this;
    }

    public StateMachine initial() {
        machine.initialState = currentState;
        return this;
    }

    public StateMachine when(String string) {
        currentTransition = new Transition(string);
        currentState.transitions.add(currentTransition);
        return this;
    }

    public StateMachine to(String string) {
        currentTransition.targetState = new State(string);
        return this;
    }

    public StateMachine integer(String string) {
        machine.integers.put(string, 0);
        return this;
    }

    public StateMachine set(String string, int i) {
        currentTransition.hasSetOperation = true;
        currentTransition.setIntOperationValue = i;
        currentTransition.operandVariableName = string;
        return this;
    }

    public StateMachine increment(String string) {
        currentTransition.hasIncrementOption = true;
        currentTransition.operandVariableName = string;
        return this;
    }

    public StateMachine decrement(String string) {
        currentTransition.hasDecrementOption = true;
        currentTransition.operandVariableName = string;
        return this;
    }

    public StateMachine ifEquals(String string, int i) {
        currentTransition.conditionalVariableName = string;
        currentTransition.conditional = Transition.conditionals.EQUAL;
        currentTransition.conditionalCompareValue = i;
        return this;
    }

    public StateMachine ifGreaterThan(String string, int i) {
        currentTransition.conditionalVariableName = string;
        currentTransition.conditional = Transition.conditionals.GREATER;
        currentTransition.conditionalCompareValue = i;
        return this;
    }

    public StateMachine ifLessThan(String string, int i) {
        currentTransition.conditionalVariableName = string;
        currentTransition.conditional = Transition.conditionals.LESS;
        currentTransition.conditionalCompareValue = i;
        return this;
    }

}

```

```java
package org.sdu.dk.frhou18;

import org.sdu.dk.frhou18.metamodel.Machine;
import org.sdu.dk.frhou18.metamodel.State;
import org.sdu.dk.frhou18.metamodel.Transition;

import java.util.List;
import java.util.NoSuchElementException;
import java.util.stream.Collectors;

public class MachineInterpreter {
    Machine machine;

    public void run(Machine m) {
        this.machine = m;
    }

    public State getCurrentState() {
        return machine.currentState;
    }

    public void processEvent(String string) {
        List<Transition> transitions;
        try {
            transitions = machine.currentState.transitions.stream()
                    .filter(transition -> transition.getEvent().equals(string)).
collect(Collectors.toList());
        }catch (NoSuchElementException e){
            return;
        }
        for (var transition : transitions) {
            if (transition.isConditional()) {
                var conditional = machine.integers.get(transition.
conditionalVariableName);
                switch (transition.conditional) {
                    case EQUAL:
                        if (conditional != transition.conditionalCompareValue)
continue;
                        break;
                    case GREATER:
                        if (conditional <= transition.conditionalCompareValue)
continue;
                        break;
                    case LESS:
                        if (conditional >= transition.conditionalCompareValue)
continue;
                        break;
                    default:
                        continue;
                    }
                }
                if (transition.hasSetOperation()) {
                    machine.integers.put(transition.operandVariableName, transition.
setIntOperationValue);
                } else if (transition.hasIncrementOperation()) {
                    machine.integers.put(transition.operandVariableName, machine.
integers.get(transition.operandVariableName) + 1);
                } else if (transition.hasDecrementOperation()) {
                    machine.integers.put(transition.operandVariableName, machine.
integers.get(transition.operandVariableName) - 1);
                }

                machine.currentState = machine.states.stream().filter(state -> state.
getName().equals(transition.targetState.getName())).findFirst().get();
                return;
            }
        }

    public int getInteger(String string) {
        return machine.integers.get(string);
    }

}

```

```java
package org.sdu.dk.frhou18.metamodel;

import java.util.ArrayList;
import java.util.List;

public class State {

    public String name;
    public List<Transition> transitions;

    public State(String name) {
        this.name = name;
        this.transitions = new ArrayList<>();
    }

    public State() {
    }

    public String getName() {
        return name;
    }

    public List<Transition> getTransitions() {
        return transitions;
    }

    public Transition getTransitionByEvent(String string) {
        return transitions.stream().filter(t -> t.getEvent().equals(string)).
findFirst().get();
    }

}

```

```java
package org.sdu.dk.frhou18.metamodel;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Machine {

    public List<State> states = new ArrayList<>();
    public State initialState;
    public Map<String, Integer> integers = new HashMap<>();
    public State currentState;


    public List<State> getStates() {
        return states;
    }

    public State getInitialState() {
        return initialState;
    }

    public State getState(String string) {
        return states.stream().filter(state -> state.getName().equals(string)).
findFirst().get();
    }

    public int numberOfIntegers() {
        return this.integers.keySet().size();
    }

    public boolean hasInteger(String string) {
        return this.integers.containsKey(string);
    }

}

```

```java
package org.sdu.dk.frhou18.metamodel;

public class Transition {

    public String transitionEvent;
    public State targetState;
    public boolean hasIncrementOption;
    public boolean hasDecrementOption;
    public String conditionalVariableName = null;
    public boolean hasSetOperation;
    public int setIntOperationValue;
    public String operandVariableName = null;
    public enum conditionals {GREATER, LESS, EQUAL}
    public conditionals conditional = null;
    public int conditionalCompareValue;

    public Transition() {
    }

    public Transition(String transitionEvent) {
        this.transitionEvent = transitionEvent;
    }

    public String getEvent() {
        return transitionEvent;
    }

    public State getTarget() {
        return targetState;
    }

    public boolean hasSetOperation() {
        return hasSetOperation;
    }

    public boolean hasIncrementOperation() {
        return hasIncrementOption;
    }

    public boolean hasDecrementOperation() {
        return hasDecrementOption;
    }

    public String getOperationVariableName() {
        return operandVariableName;
    }

    public boolean isConditional() {
        return conditional != null;
    }

    public String getConditionVariableName() {
        return conditionalVariableName;
    }

    public Integer getConditionComparedValue() {
        return conditionalCompareValue;
    }

    public boolean isConditionEqual() {
        return conditional.equals(conditionals.EQUAL);
    }

    public boolean isConditionGreaterThan() {
        return conditional.equals(conditionals.GREATER);
    }

    public boolean isConditionLessThan() {
        return conditional.equals(conditionals.LESS);
    }

    public boolean hasOperation() {
        return hasDecrementOption || hasIncrementOption;
    }
}

```