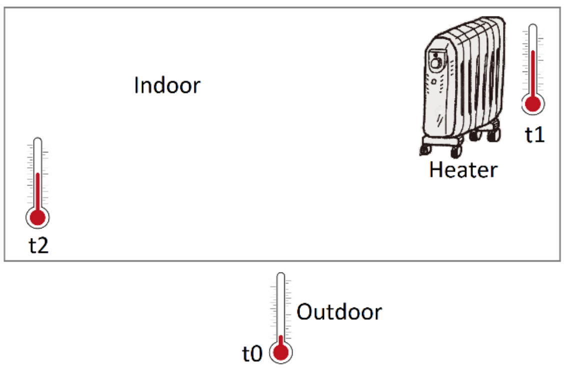
Assignment 1, SDJ2
(MVVM, Observer, State, Threads)

The system:

You must design and implement a simple application for a module to control the heating system in a summerhouse.



There is one heater with the power positions 0, 1, 2 and 3, where 0 indicates that the heater is turned off, 1 is low power, 2 is medium power and 3 is the highest power. There is a timeout in power position 3 such the radiator after a timeout automatically goes to power position 2.

The indoor temperatures are measured in two locations, thermometer t1 near the heater, and thermometer t2 in the opposite corner. The distance between t1 and heater is 1 m and the distance between t2 and the heater is 7 m.

The temperature depends on the heaters power, the distance to the heater and the outdoor temperature. The method given in Appendix on the last page may be used to calculate temperature values.

The assignment:

- From a GUI with at least two windows, you
  - Show the current indoor temperatures (from both indoor thermometers), the outdoor temperature, and the heaters power position.
  - Get warnings if an indoor temperature passes critical values HIGH or LOW
  - Control the heater (only options to turn up and to turn down)
  - Define the critical values HIGH and LOW
  - Optionally, show in tables or charts, the values from the latest say 20 measurements
- From two threads (one for t1 and one for t2) you simulate the current indoor temperature
  - The temperature is measured and send after 4 to 8 seconds (in average every 6th second)
  - You may use the method given in Appendix A.
- Optional: simulate the outdoor temperature in another thread, e.g. using the method in Appendix B. Alternatively, use a fixed value.

Requirements

- You must use MVVM with at least two different windows.
- You must use the Observer design pattern as part of the solution.
- You must use the State design pattern for the different power states for the radiator. This includes that power state 3 has a timeout and automatically turns down the radiator after say 40 seconds
- You must use a thread for each of the thermometers t1 and t2.
- It is required to make a class diagram for the final solution and a state machine diagram for the radiator (both in Astah). In the diagram you must be able clearly to identify the different MVVM parts, the Observer pattern, the State pattern and the classes related to the threads

Deadline

You can work on the assignment in the SDJ2 lessons in week 39 – and of course after class.

Deadline: Wednesday the 9th of March at 23:59

Format

It is ok to work in groups, but you have to hand individually - in a single zip-file with

1) Class diagram, State machine diagram
2) Source code for all Java classes and
3) Related resources like fxml files, and if used, external jar files

Evaluation

Your hand-in will be registered and counts for one of the exam requirements. No feedback will be given.

Appendix A – Thermometer method (to calculate internal temperature)

```
/**
 * Calculating the internal temperature in one of two locations.
 * This includes a term from a heater (depending on location and
 * heaters power), and a term from an outdoor heat loss.
 * Values are only valid in the outdoor temperature range [-20; 20]
 * and when s, the number of seconds between each measurements are
 * between 4 and 8 seconds.
 *
 * @param t  the last measured temperature
 * @param p  the heaters power {0, 1, 2 or 3} where 0 is turned off,
 *           1 is low, 2 is medium and 3 is high
 * @param d  the distance between heater and measurements {1 or 7}
 *    where 1 is close to the heater and 7 is in the opposite corner
 * @param t0 the outdoor temperature (valid in the range [-20; 20])
 * @param s the number of seconds since last measurement [4; 8]
 * @return the temperature
 */
public double temperature(double t, int p, int d, double t0, int s)
{
  double tMax = Math.min(11 * p + 10, 11 * p + 10 + t0);
  tMax = Math.max(Math.max(t, tMax), t0);
  double heaterTerm = 0;
  if (p > 0)
  {
    double den = Math.max((tMax * (20 - 5 * p) * (d + 5)), 0.1);
    heaterTerm = 30 * s * Math.abs(tMax - t) / den;
  }
  double outdoorTerm = (t - t0) * s / 250.0;
  t = Math.min(Math.max(t - outdoorTerm + heaterTerm, t0), tMax);
  return t;
}
```

Appendix B – Thermometer method (to calculate external temperature)

```
/**
 * Calculating the external temperature.
 * Values are only valid if the temperature is being measured
 * approximately every 10th second.
 *
 * @param t0  the last measured external temperature
 * @param min a lower limit (may temporally be deceeded)
 * @param max an upper limit (may temporally be exceeded)
 * @return an updated external temperature
 */
public double externalTemperature(double t0, double min, double max)
{
  double left = t0 - min;
  double right = max - t0;
  int sign = Math.random() * (left + right) > left ? 1 : -1;
  t0 += sign * Math.random();
  return t0;
}
```