CVS 17/18 - Handout 3

Consider a domotic system that provides a hub for a set of devices, sensors or actuators, and control rules.

Sensors A sensor is updated (internally) by an internal timer that updates its state with readings that can be read concurrently by threads in the domotic control system. In this case, sensor readings are simulated by with a random value, or by some function over time. A sensor can be read concurrently. Sensors have a scale, and if a reading is out of the scale, the registered value is either the maximum or the minimum of the scale.

Actuators Actuators have an internal state with a property named "value" (setter and getter). The value can be read and written concurrently.

Rules Rules are objects that have references to sensors and actuators. Their behavior is triggered by an internal timer. There are several examples of rules: Indoor lighting (if someone is present and external light is below a certain value, turn lights on); Office hours (lock doors outside the working hours); AutoWindows (Close windows automatically after 5PM); Rain/Wind Protection (Close windows if rain sensor or wind sensor is above a threshold).

CVS 17/18 - Handout 3

Consider the code skeleton for a classes Probe and SensorInt.

```
import java.util.Random;
import java.util.concurrent.*;

class Probe implements Runnable {

    private SensorInt sensor;

    public Probe(SensorInt sensor)
    { this.sensor = sensor; }

    public void run()
    {
        while( true )
        {
            // produce a new value every 2 seconds
        }
    }
}
```

CVS 17/18 - Handout 3

Consider the code skeleton for a classes Probe and SensorInt.

```
class SensorInt {
    int value;
    int min;
    int max;

    Thread th;

    static final int SAMPLING = 3;

    SensorInt(int min, int max) {
        this.min = min;
        this.max = max;
        this.value = min;
        this.th = new Thread(new Probe(this));
        this.th.start();
    }

    int get() { return this.value; }

    void set(int value) { this.value = value; }

    public static void main(String args[]) throws InterruptedException {
        SensorInt s = new SensorInt(0,10);
        while(true) {
            // get and print a sample every 5 seconds
        }
    }
}
```

CVS 17/18 - Handout 3

You must to complete the implementation of the **Sensor** above and verify its correction. You should use monitors to discipline the access to the shared state and verify the access to the sensor readings with the help of fractional permissions.

You should add contracts for classes Random and TimeUnit to your verifast specification files.

```
package java.lang;

import java.util.Random;
…
public class Random {

    Random();
        //@ requires true;
        //@ ensures true;

    int nextInt(int n);
        //@ requires true;
        //@ ensures true;
}
```

```
package java.util.concurrent;

...

public abstract class TimeUnit {

  public static TimeUnit SECONDS;

  void sleep(int s);
  //@ requires true;
  //@ ensures true;
}
```