SWT6_UE1_G2_Voraberger

S2010307040 | Anna Voraberger Aufwand: 9:45-10:15 17:30-19:00 19:00-19:45 9-11

timer-bean

- timer-bean aus der Übung importiert.
- In HomeControl verwende ich den Timer mit seiner Factory

balcony-power-plant

- Als Modul neu angelegt
- Die InverterApi wird per Factory zur Verfügung gestellt.
- Die Implementierung davon erfolgt in der RandomInverter Klasse. Vor allem für die Testbarkeit wird dieser Klasse der gewünschte von Java.util.Random abgeleitete Zufallsgenerator injiziert. Das wird dem Client in der Factory aber nicht zugängig gemacht. Die Factory erstellt eine InverterApi mit der RandomInverter Implementierung, die ein Standard Random() Objekt bekommt

```
public class RandomInverter implements InverterApi {
  @Setter
  static Random randomizer;

  // gets a generator for random numbers which is used to generate the
value for actual current
  public RandomInverter(Random randomizer) {
    RandomInverter.randomizer = randomizer;
  }

  // returns a random value between 0.000 and 0.800
  // uses a randomizer to generate a value between 0 and 800 and returns
  // the generated value divided by 1000
  @Override
  public double getActualCurrent() {
    int randomNumber = randomizer.nextInt(801);
    return ((double)randomNumber) / 1000;
  }
}
```

```
public class InverterApiFactory {

  public static InverterApi createInverterApi(){
    InverterApi result = new RandomInverter(new Random());
    return result;
```

```
    }
}
```

Tests

- Für die Tests wurde ein RandomStub erstellt, der eine Value-Liste bekommt
- Im Setup des RandomInverter Tests übergebe ich dem Stub eine Liste an Values, die dem RandomInverter übergeben werden.
- Für die Tests habe ich zursätzlich die Implementierung in der module-info exportiert und diese Zeile im Nachgang wieder auskommentiert.

```
  @BeforeAll
  public static void setUp(){
    ArrayList<Integer> randomValues = new ArrayList<>();
    //no need to test negative values since Random.nextInt() only returns
positive Values
    randomValues.add(0);
    randomValues.add(800);
    randomValues.add(100);
    randomValues.add(280);
    randomValues.add(799);
    randomValues.add(1);
    inverter = new RandomInverter(new RandomStub(randomValues));
  }
```

ac-control

- Als Modul neu angelegt
- Die AirConditionApi wird per Factory zur Verfügung gestellt
- Ich habe das Interface um eine Abfrage isOn() erweitert
- Die Implementierung erfolgt in der AirConditionControl-Klasse.
    - Hier wird bei getRoomTemperature immer ein Zufallswert zwischen 19.0 und 30.0 zurückgegeben
    - Mit turnOn() und turnOff() wird immer das Attribut airconIsOn gesetzt und zusätzlich "Turn Aircon On/Off" auf der Konsole ausgegeben
    - isOn() gibt den aktuellen Status zurück und zusätzlich auf der Konsole aus

```
public class AirConditionControl implements AirConditionApi {
  private boolean airconIsOn = false;
  private static Random randomizer;
  private static int min = 190;
  private static int max = 300;


  public AirConditionControl(Random random) {
    this.randomizer = random;
```

```
    }

    @Override
    public void turnOn() {
      airconIsOn = true;
      System.out.println("==== Turn Aircon ON ====");
    }

    @Override
    public void turnOff() {
      airconIsOn = false;
      System.out.println("==== Turn Aircon OFF ====");
    }

    @Override
    public double getRoomTemperature() {
      int rand = randomizer.nextInt(min,max+1);
      return (double)rand / 10;
    }

    public boolean isOn(){
      if (airconIsOn){
        System.out.println("==== Aircon still running ====");
      } else {
        System.out.println("==== Aircon still off ====");
      }
      return airconIsOn;
    }

  }
```

Tests

- Für den Unittest habe ich auch hier wieder einen Random Stub verwendet
- Eine Zusätzliche Testmethode prüft die on-off Methoden

home-control

Home Control verwendet die 3 vorhergehenden Module und stellt damit die Hauptlogik zur Verfügung.

Datenstruktur

- Um die letzten 10 Temperaturen und kWh Werte zu speichern, verwende ich zwei Double-Ended Queues (als private fields angelegt) mit der ArrayDeque Implementierung
- Dadurch kann ich "sauber" vorne neue Werte hinzufügen und hinten wieder rausnehmen.
- Initialgröße ist gleich festgelegt und um 1 erhöht (Übergangszustände), damit es nie geresized werden muss.

```
  private static Deque<Double> pastTemperatures = new ArrayDeque<>
(nrOfTemperatures+1);
  private static Deque<Double> pastCurrents = new ArrayDeque<>
(nrOfCurrents+1);
```

Main Methode

In der Main Methode wird der Timer angelegt und gestartet. Für das, was alle 5 Sekunden passieren soll, habe ich ein Event registriert.

```
  public static void main(String args[]){
    // create Timer
    Timer timer = TimerFactory.createTimer(timerInterval, numberOfTicks);
    // register Event
    timer.addTimerListener(event -> handleTimerEvent());
    // start timer
    timer.start();
  }
```

Verbesserungspotenzial: Ich würde im Realfall eine Möglichkeit anbieten, den Timer unendlich viele Ticks laufen zu lassen. Für "unendlich" müsste ich in diesem Fall prüfen, ob der Timer noch läuft und den Timer ansonsten einfach wieder neu starten.

Handle Timerevent

- Feuert der Timer das Event, füge ich die aktuelle RoomTemperature und Current zu meinen Queues.
- Befinden sich jetzt mehr als 10 Elemente in den Queues, wird jeweils das letzte Element entfernt
- Befinden sich noch keine 10 Elemente darin, returne ich. Handling zuende.
- Prüfen der Durchschnittstemperatur. Ist diese zu kalt, rufe ich die Methode tooCold(), tooWarm() für zu warm.

```
  public static void handleTimerEvent(){
    // add new values
    pastTemperatures.addFirst(aircon.getRoomTemperature());
    pastCurrents.addFirst(inverter.getActualCurrent());

    // remove old values
    if (pastTemperatures.size() > nrOfTemperatures)
      pastTemperatures.removeLast();
    if (pastCurrents.size() > nrOfTemperatures)
      pastCurrents.removeLast();

    printDeques();

    if ((pastTemperatures.size() < nrOfTemperatures) ||
(pastCurrents.size() < nrOfCurrents))
      return; //nothing to do, not enough values yet
```

```
    // check average temperature
    Double averageTemp = averageTemperature();

    if (averageTemp < coldestTemp)
      tooCold();

    if (averageTemp > warmestTemp)
      tooWarm();
  }
```

Too Cold

- Prüfe, ob Aircon an
- Wenn sie läuft, schalte ich sie ab

```
  private static void tooCold() {
    if (!aircon.isOn())
      return;
    aircon.turnOff();
  }
```

Too Warm

- Hole mir den averageCurrent()
- Wenn ausreichend vorhanden ist, wird die Kilmaanlage angeschaltet. Ansonsten nicht.

```
private static void tooWarm() {
    Double averageCurrent = averageCurrent();
    if (averageCurrent < minCurrentForAircon){
      System.out.println("==PV Power not sufficient==");
      return;

    }

    if(aircon.isOn())
      return;

    aircon.turnOn();
  }
```

average

- Ich berechne mit average() den Durchschnitt eines Deques von Double-Werten.
- Zusätzlich habe ich noch eine averageTemperature() und eine averageCurrent() Methode, die die allgemeine average() verwenden, aber jeweils eine ordentliche Konsolenausgabe machen

```
  public static Double average(Deque<Double> deque){
    Double sum = 0.0;
    for (Double d:deque) {
      sum += d;
    }
    return sum / deque.size();
  }

  private static Double averageTemperature() {
    Double averageTemp = average(pastTemperatures);
    System.out.print("Average Temperature: ");
    System.out.printf("%.2f",averageTemp);
    System.out.println();
    return averageTemp;
  }

  private static Double averageCurrent() {
    Double averageCurrent = average(pastCurrents);
    System.out.print("Average Current: ");
    System.out.printf("%.2f",averageCurrent);
    System.out.println();
    return averageCurrent;
  }
```

Tests

Für das Hauptprogramm habe ich keine Unit-Tests geschrieben, sondern das Verhalten "einfach" beobachtet.

Hier die unterschiedlichen Verhaltens-Fälle:

Initiale Befüllung der Queues

Anfangs wird noch kein Durchschnitt berechnet oder sonstiges Verhalten getriggert, da sich noch keine 10 Temperaturen / PV-Daten in den Behältern befinden.

Sichtbar ist auch, dass die Queue die neuen Werte immer hintenanfügt. Am letzten Fall ist auch beobachtbar, dass der alte Wert hinten weggenommen wird.

```
==========================
Past Temperatures: [28.1]
Past Currents: [0.748]
==========================


==========================
Past Temperatures: [20.7, 28.1]
Past Currents: [0.632, 0.748]
==========================
```

```
==========================
Past Temperatures: [20.3, 20.7, 28.1]
Past Currents: [0.148, 0.632, 0.748]
==========================


==========================
Past Temperatures: [28.2, 20.3, 20.7, 28.1]
Past Currents: [0.163, 0.148, 0.632, 0.748]
==========================


==========================
Past Temperatures: [19.9, 28.2, 20.3, 20.7, 28.1]
Past Currents: [0.086, 0.163, 0.148, 0.632, 0.748]
==========================


==========================
Past Temperatures: [24.9, 19.9, 28.2, 20.3, 20.7, 28.1]
Past Currents: [0.726, 0.086, 0.163, 0.148, 0.632, 0.748]
==========================


==========================
Past Temperatures: [26.3, 24.9, 19.9, 28.2, 20.3, 20.7, 28.1]
Past Currents: [0.538, 0.726, 0.086, 0.163, 0.148, 0.632, 0.748]
==========================


==========================
Past Temperatures: [28.0, 26.3, 24.9, 19.9, 28.2, 20.3, 20.7, 28.1]
Past Currents: [0.28, 0.538, 0.726, 0.086, 0.163, 0.148, 0.632, 0.748]
==========================


==========================
Past Temperatures: [27.4, 28.0, 26.3, 24.9, 19.9, 28.2, 20.3, 20.7, 28.1]
Past Currents: [0.231, 0.28, 0.538, 0.726, 0.086, 0.163, 0.148, 0.632,
0.748]
==========================


==========================
Past Temperatures: [22.2, 27.4, 28.0, 26.3, 24.9, 19.9, 28.2, 20.3, 20.7,
28.1]
Past Currents: [0.436, 0.231, 0.28, 0.538, 0.726, 0.086, 0.163, 0.148,
0.632, 0.748]
==========================


Average Temperature: 24,60
Average Current: 0,40
```

```
==== Aircon still off ====
==== Turn Aircon ON ====
=========================
Past Temperatures: [29.9, 22.2, 27.4, 28.0, 26.3, 24.9, 19.9, 28.2, 20.3,
20.7]
Past Currents: [0.6, 0.436, 0.231, 0.28, 0.538, 0.726, 0.086, 0.163,
0.148, 0.632]
=========================
```

Ideale Temperatur

Zwischen 22.0 und 24.0 Grad Durchschnittstemperatur passiert nichts. Hier werden auch keine durchschnittlichen kWh berechnet und es wird nicht geprüft, ob die Klimaanlage derzeit läuft oder nicht.

```
Average Temperature: 23,40
=========================
Past Temperatures: [27.9, 19.3, 24.2, 24.6, 19.5, 21.0, 20.9, 27.5, 27.3,
28.7]
Past Currents: [0.165, 0.626, 0.189, 0.724, 0.34, 0.085, 0.349, 0.747,
0.17, 0.674]
=========================
```

Zu warm

Ist es durchschnittlich wärmer als 24.0 Grad, gibt es drei Möglichkeiten:

1. Es ist genug Strom vorhanden und die Klimaanlage ist noch aus. Dann wird die Klimaanlage angeschaltet.

   ```
   Average Temperature: 24,45
   Average Current: 0,25
   ==== Aircon still off ====
   ==== Turn Aircon ON ====
   =========================
   Past Temperatures: [27.1, 29.4, 27.2, 27.3, 23.0, 29.0, 20.4, 21.6,
   22.5, 20.3]
   Past Currents: [0.359, 0.319, 0.132, 0.194, 0.204, 0.471, 0.169,
   0.502, 0.288, 0.007]
   =========================
   ```

2. Die Klimaanlage läuft bereits. Dann passiert nichts. Abgrenzung in diesem Fall: Ich prüfe hier auch nicht, ob vielleicht zu wenig Strom da ist. Die Klimaanlage bleibt unabhängig vom Stromaufkommen an, da das ja das Wunschverhalten ist. Außerdem rechne ich mit 0.0-0.8 Werten alle 5 Sekunden. Die Wahrscheinlichkeit, dass in 10 Sekunden wieder durchschnittlich mehr als 0.1 verfügbar ist, ist sehr hoch.

   ```
   ```

```
Average Temperature: 24,78
Average Current: 0,26
==== Aircon still running ====
==========================
Past Temperatures: [21.1, 27.1, 29.4, 27.2, 27.3, 23.0, 29.0, 20.4,
21.6, 22.5]
Past Currents: [0.488, 0.359, 0.319, 0.132, 0.194, 0.204, 0.471,
0.169, 0.502, 0.288]
==========================
```

3. Die Klimaanlage läuft noch nicht, doch es ist nicht ausreichend Strom vorhanden. Dieser Fall kommt sehr selten vor. Um Ihn zu Triggern habe ich vorübergehend das Limit auf 0.5 hinaufgesetzt:

```
Average Temperature: 24,50
Average Current: 0,36
==PV Power not sufficient==
==========================
Past Temperatures: [26.2, 26.8, 21.1, 27.2, 28.2, 24.7, 23.7, 23.9,
24.6, 20.0]
Past Currents: [0.464, 0.067, 0.292, 0.501, 0.634, 0.371, 0.196,
0.475, 0.013, 0.789]
==========================
```

Zu kalt

In diesem Fall wird auch nicht nach dem aktuellen Stromaufkommen gefragt. Ist die Klimaanlage noch an, wird sie ausgeschaltet.

```
Average Temperature: 21,81
==== Aircon still running ====
==== Turn Aircon OFF ====
==========================
Past Temperatures: [29.0, 20.4, 21.6, 22.5, 20.3, 23.8, 24.1, 21.9, 19.1,
25.4]
Past Currents: [0.471, 0.169, 0.502, 0.288, 0.007, 0.263, 0.115, 0.421,
0.614, 0.195]
==========================
```