Übungsaufgabe 1, Deadline

01.04.2016, 23:55 Uhr

1. Drive Analytics Dashboard

Das Drive Analytics Dashboard ist eine Java GUI Anwendung mit der Java UI Technologie Ihrer Wahl: SWT, Swing, JavaFX. Die Anwendung visualisiert Daten von unterschiedlichen automotive Sensoren, welche in einem Fahrzeug verbaut sind. Ihre Aufgabe besteht darin die Daten eines Parksensors (DistanceSensor) zur Abstandsmessung am Heck des Fahrzeuges und eines Weiteren von Ihnen frei wählbaren Sensors zu visualisieren. Weitere mögliche Sensoren könnten z. B. ein Regensensor (liefert die gemessene Luftfeuchtigkeit), ein Temperatursensor, Geschwindigkeitssensor oder Reifendrucksensor sein. Ein Sensor liefert immer Daten in Form eines Byte-Arrays. Die Daten können entweder als double oder als long interpretiert werden. Das Java Interface eines Sensors ist wie folgt definiert:

ISensor.java

```
public interface ISensor {

    public enum SensorDataFormat {
      PERCENT /*double*/,
      ABSOLUTE_VALUE_LONG /*long*/
    }

    String getSensorId();
    byte[] getData();
    SensorDataFormat getDataFormat();
      // format of the elements in den data array, e. g. “double”, “long”
}
```

Die beiden unterschiedlichen Sensorwerte sollen ansprechend formatiert auf einer grafischen Benutzeroberfläche dargestellt werden können, z.B. mit einem Label oder einem Fortschrittsbalken.

Die Benuteroberfläche soll die Sensorwerte in regelmäßigen Zeitintervallen aktualisieren, d. h. die Werte von den verfügbaren Sensoren abfragen (polling). Verwenden Sie dazu das Timer-Bean aus der 1. Übung.

Sensoren müssen als OSGI Service realisiert werden. Jeder Sensor ist mittels separaten OSGi-Bundle zu realisieren (z.B. at.swt6.ParkingSensor, z. B. at.swt6.TemperatureSensor).

Sensoren müssen sich dynamisch beim Starten des Bundles registrieren (Hinweis: verwenden Sie dazu den OSGi-Service-Tracker).

Bonus: Bei sicherheitskritischen Sensoren (z. B. Reifendruck- oder ABS-Sensor) können Millisekunden oftmals entscheidend sein. Aus diesem Grund sollen Sensoren bei einer Statusänderung Notifikationen an einen SensorListener schicken können. Dazu implementieren Sie einen SensorListenerService welcher folgendes Interface anbietet:

ISensor.java

```
public interface SensorListener {
  public void valueChanged(ISensor sensor);
}
```

Beim Starten eines Sensor Bundles wird nach bestehenden SensorListener Services gesucht (Hinweis: verwenden Sie dazu den in der Übung kennengelernten ServiceTracker). Ändert sich der Sensorwert, so wir der SensorListenerService notifiziert (valueChanged). Denken Sie ebenfalls daran, dass beim Bundlestart oder während der Laufzeit SensorListeners nicht verfügbar sein oder wieder entfernt werden können. Wird der SensorListenerService über Sensorwertänderungen notifiziert, so können diese einfach auf der CMD Line ausgegeben werden (keine graphische Visualisierung im Dashboard nötig).

Bundle Struktur:

- at.swt6.main
- at.swt6.sensor
- at.swt6.sensor.DistanceSensor
- at.swt6.sensor.[Ihr eigener Sensor]

1.1. Aufgabe

Verwendung des obigen Sensor-Interfaces und erstellen entsprechender Sensor Implementierungen (Beachten Sie hierbei das Factory-Pattern) Implementierung von zumindest zwei unterschiedlichen Sensoren (z.B.: DistanceSensor und TemperatureSensor). Jeweils einen für absolute Werte und Prozentwerte. Als Daten können die Sensoren zufällige Werte zurückliefern. Erstellung einer Java-basierten Benutzeroberfläche. Sensoren registrieren und deregistrieren sich dynamisch am Dashboard. Benutzeroberfläche wird mittles TimerBean in regelmäßigen Intervallen aktualisiert. Das Timer-Intervall kann man über die UI einstellen. Bonus: Sensor-Bundles können SensorListeners bei Statusänderungen informieren.