Technology
Arts Sciences
TH Köln

Fakultät für Informations-, Medien- und Elektrotechnik
Institut für Nachrichtentechnik

Prof. Dr. Hans W. Nissen
Email: hans.nissen@th-koeln.de

Entwicklung komplexer Software Systeme

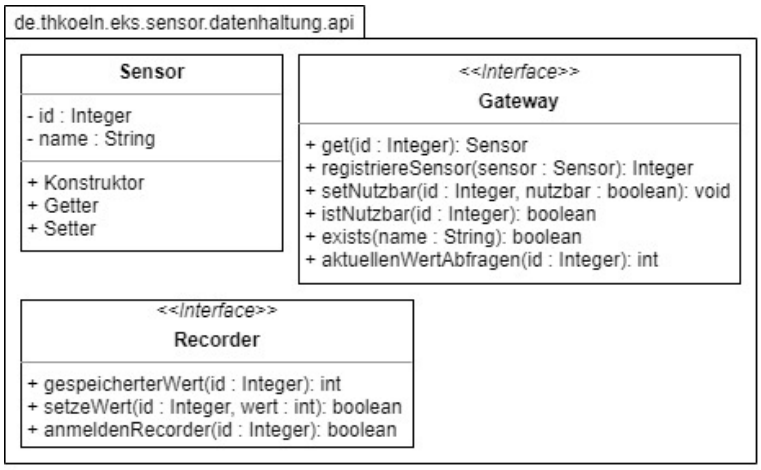
Praktikumsblatt 3
Gruppe B
- Hausaufgaben -

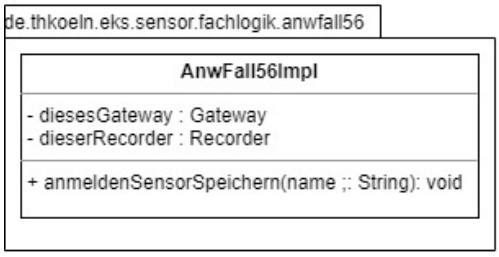
Ziel: Erstellung von Mockito-Testfällen

Abgabe der Lösungen: Bis zum 16.01., 08:00 Uhr morgens, im Master-Branch des Gitlab-Repositories P3EKS<IhreTeamnummer>. Abzugeben ist das vollständige IntelliJ-Projekt.
Hinweis: Ihr Gitlab-Repository zu diesem Praktikumsversuch enthält bereits ein IntelliJ-Projekt, welches Sie als Grundlage für Ihre Implementierung verwenden sollen – Sie sollen also den Ordner P3EKS< IhreTeamnummer> als IntelliJ-Projekt öffnen.

Gegeben sind die

- folgenden 3 Klassen der Schicht Datenhaltung:
  - Entitätsklasse `Sensor`,
  - Interface-Klasse `Gateway`,
  - Interface-Klasse `Recorder`,
- und die Klasse `AnwFall56Impl` in der Schicht Fachlogik, welche den (fiktiven) Anwendungsfall 56 implementiert. Dieser Anwendungsfall (bzw. seine Implementierung) ist nur für diese Aufgabe konstruiert und macht sonst wenig Sinn.





`Sensor`:

- `name` ist der Name des Sensors
- `id` ist die eindeutige Nummer des Sensors

Technology
Arts Sciences
TH Köln

Fakultät für Informations-, Medien- und Elektrotechnik
Institut für Nachrichtentechnik

Prof. Dr. Hans W. Nissen
Email: hans.nissen@th-koeln.de

`Gateway`: ist das Gateway zu allen vorhandene Sensoren im Haus und speichert und verwaltet alle bekannten Sensoren

- `get(id)` liefert den Sensor mit der Nummer `id`
- `registriereSensor(sensor)` erzeugt den Sensor `sensor` im `Gateway` und liefert als Ergebnis die eindeutige Nummer des neuen Sensors. (Hinweis: Die `id` des als Parameter übergebenen Sensors muss `null` sein)
- `setNutzbar(id, nutzbar)` setzt die Nutzbarkeit des Sensors mit der Nummer `id` auf `nutzbar` (also `true` oder `false`).
- `istNutzbar(id)` liefert als Ergebnis die Nutzbarkeit des Sensors mit der Nummer `id` (also `true` oder `false`).
- `exists(name)` prüft, ob im `Gateway` ein Sensor mit dem Namen `namen` bereits registriert ist.
- `aktuellenWertAbfragen(id)` ermittelt den aktuellen Messwert vom Sensor mit der Nummer `id`. Der Messwert ist in unserem Beispiel immer vom Datentyp `int`.

`Recorder`: speichert einen Messwert der am Gateway registrierten Sensoren

- `gespeicherterWert(id)`: gibt den im Recorder derzeit gespeicherten Messwertes des Sensors mit der Nummer `id` zurück.
- `setzeWert(id, wert)`: speichert für den Sensor mit der Nummer `id` den Messwert `wert` im Recorder.
- `anmeldenRecorder(id)`: meldet den Sensor mit der Nummer `id` dem Recorder hinzu, damit ab jetzt Messwerte dieses Sensors gespeichert werden können.

`AnwFall56Impl`:

- `anmeldenSensorSpeichern(name)` registriert einen neuen Sensor basierend auf dem Namen `namen` am Gateway, meldet diesen Sensor beim Recorder an und speichert im Recorder den aktuellen Messwert des Sensors. Diese Methode ist eine korrekte Implementierung des Anwendungsfalls 56 und darf nicht verändert werden.

Ihre Aufgaben:
Erstellen Sie im Modul `de.thkoeln.eks.sensor.fachlogik` im dortigen Verzeichnis `src/test/java` im Paket `de.thkoeln.eks.sensor.fachlogik.anwfall56` die Testklasse `TestAnwFall56Impl` mit dem unten aufgeführten Testfall.
Erstellen Sie hierfür für die Interfaces `Gateway` und `Recorder` entsprechende Mocks. Verwenden Sie die Dependency Injection (DI) für die Injektion dieser Mocks.

Testfall: Prüft mit Mockito die Aufrufe der Methode `anmeldenSensorSpeichern(sensorname)` der Klasse `AnwFall56Impl`.
In diesem Testfall müssen die folgenden Bedingungen geprüft werden:

- Die Reihenfolge der Aufrufe muss exakt wie hier angegeben eingehalten werden!
- Aufruf `exists()` mit `sensorname` (Hinweis: Rückgabe von `exists()` muss natürlich `false` sein.)
- Aufruf `registriereSensor()` muss mit Sensor mit folgenden Attributen erfolgen (ArgumentCaptor hierfür verwenden!):
    - `name` ist String `sensorname`+"MeinSensor" (Hinweis: Hört sich falsch an, soll aber in dieser Aufgabe so gemacht werden ;-)

Technology
Arts Sciences
TH Köln

Fakultät für Informations-, Medien- und Elektrotechnik
Institut für Nachrichtentechnik

Prof. Dr. Hans W. Nissen
Email: hans.nissen@th-koeln.de

    - `id` ist `null`
- Aufruf `get()` mit `ID` (= Rückgabe von `registriereSensor()`) (Hinweis: Hier soll ein (im Testfall selbst erzeugtes) Sensor-Objekt mit korrektem Namen und korrekter ID als Ergebnis geliefert werden)
- Aufruf `aktuellenWertAbfragen()` mit `ID`
- Aufruf `istNutzbar()` mit `ID` (Hinweis: Rückgabe sollte `false` sein)
- Aufruf `setNutzbar()` mit `ID` und `true`
- Aufruf `istNutzbar()` mit `ID` (Hinweis: Rückgabe sollte `true` sein)
- Aufruf `anmeldenRecorder()` mit `ID`
- Aufruf `istNutzbar()` mit `ID` (Hinweis: Rückgabe sollte `true` sein)
- Aufruf `gespeicherterWert()` mit `ID` (Hinweis: Die Rückgabe ist ein beliebiger Wert)
- Aufruf `aktuellenWertAbfragen()` mit `ID`
- Aufruf `setzeWert()` mit `ID` und Rückgabe von `aktuellenWertAbfragen()`
- Aufruf `gespeicherterWert()` mit `ID`

- `get()` muss mindestens 1 Mal aufgerufen werden
- `registriereSensor()` muss genau 1 Mal aufgerufen werden
- `anmeldenRecorder()` muss mindestens 1 Mal aufgerufen werden

Für alle diese Aufrufe ist korrektes stubbing zu erstellen.