Actors – Home Automation System

Aufgabe

Wir hatten die spannende Aufgabe, ein Home-Automation-System zu entwickeln. Dabei wurde uns die Verwendung des Akka-Frameworks empfohlen. Akka ist ein leistungsstarkes Framework zur Entwicklung skalierbarer, fehlertoleranter Anwendungen, das auf dem Aktoren Modell basiert und asynchrone sowie verteilte Kommunikation ermöglicht. In unserem Projekt spielen die Hauptakteure AirConditioner, Blinds, MediaStation und der Fridge eine zentrale Rolle.

Dank der Vorlage haben wir das Projekt in drei Hauptteile aufgeteilt: Aktuator, Sensor und Simulator. Vor unserem Treffen haben wir eine detaillierte Liste erstellt, um sicherzustellen, dass wir genau das bekommen, was wir benötigen.

Package actuator

- Class AirConditioner
  - Schaltet AC ein und aus
- Class Blinds
  - Öffnet und schließt die Jalousien
- Class MediaSation
  - Spielt Filme ab
- Class Fridge
  - Verwaltet Produkte
- Class OrderProcessor
  - Bestellt aufgebrauchte Produkte
- Class Product
  - Produkte die im Kühlschrank sind

Package app

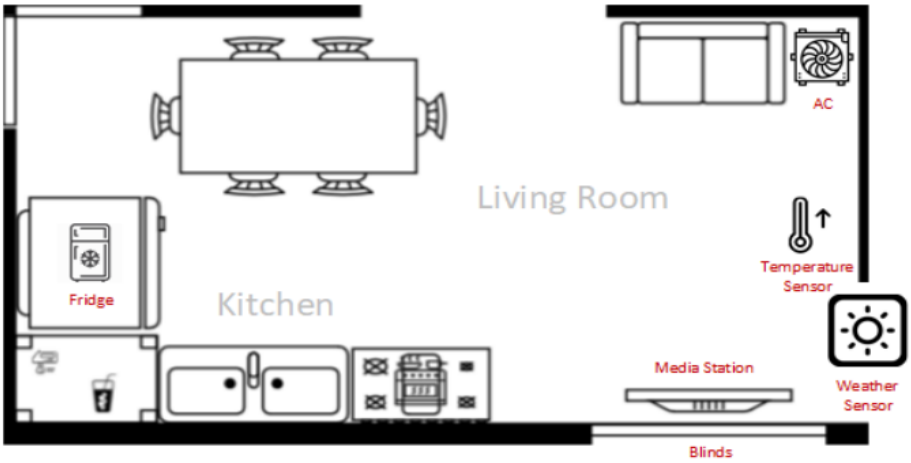
- Terminal
  - Liest input des Users
    Führt commando aus

Package simulator

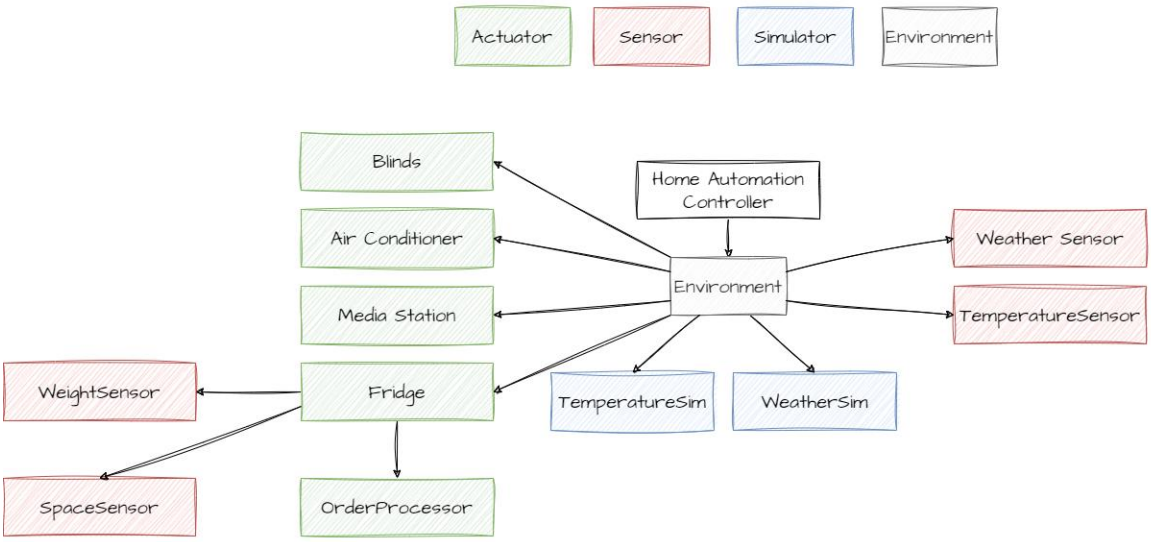
- Class TemperaturSim
- Class WeatherSim
- Package Data
  - TemperatureData
  - WeatherData

Package sensor

- Class TemperaturSensor
  - misst die Temperatur und gibt sie weiter
- Class WeatherSensor
  - Misst das Wetter und gibt es weiter
- Class WeightSensor
- Class SpaceSensor



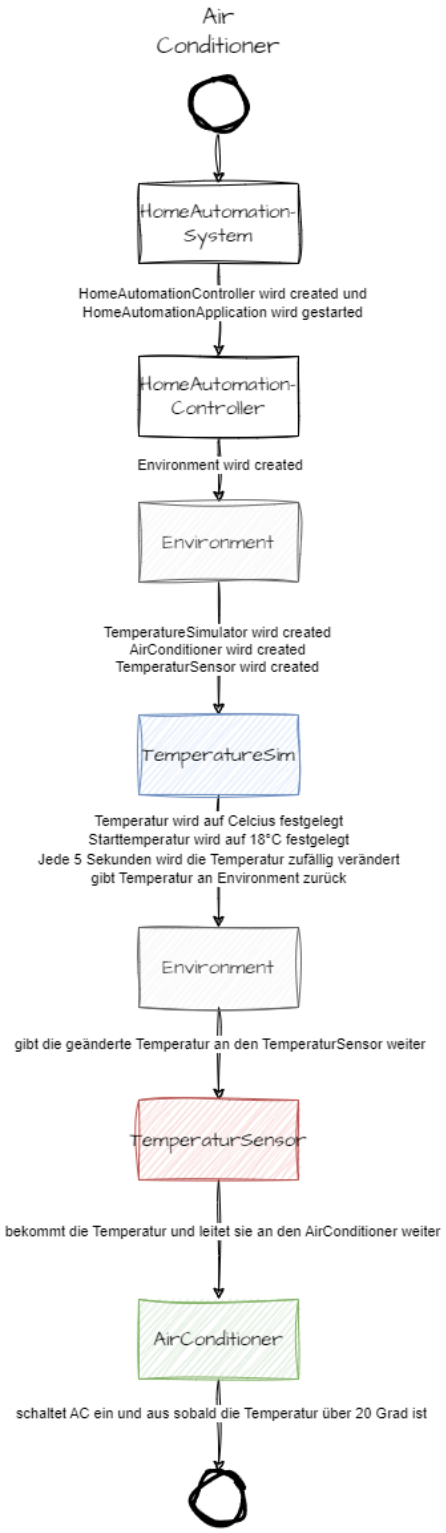
Aufbau



Unsere Applikation ist wie folgt strukturiert. Wir haben den **`Home Automation Controller`**, der das `Environment` erstellt. Im Environment werden die meisten `Aktuatoren`, **`Sensoren`** und **`Simulatoren`** erstellt. Wenn wir von links nach rechts schauen, sehen wir zuerst die `Blinds`, gefolgt vom `Air-Conditioner`, der `MediaStation` und schließlich dem `Fridge`. Der `Fridge` wiederum erstellt den Aktuator `OrderProcessor` sowie die beiden Sensoren **`WeightSensor`** und **`SpaceSensor`**. Das Environment erstellt zusätzlich die Simulationen **`TemperatureSim`** und **`WeatherSim`** sowie die Sensoren **`WeatherSensor`** und **`TemperatureSensor`**. Eine detaillierte Erklärung der Aufgaben jeder Klasse erfolgt in den nachfolgenden Abschnitten. Der **`WeatherSensor`** und der **`TemperaturSensor`** bekommen und schicken die ganze Zeit die Daten und sind daher immer im Betrieb, da sich das Wetter und die Temperatur sich ständig ändern, während der **`WeightSensor`** und der **`SpaceSensor`** angefragt werden müssen, da nur wenn man Bestellt die Sensoren benutzt werden.

Unsere Vorgehensweise

Wir haben uns als aller erstens an den Air-Conditioner ran gewagt.



Lassen Sie uns nun gemeinsam jeden Schritt durchgehen, was genau mit der Klimaanlage passiert, wenn wir unser Programm starten.

Nachdem wir das Programm gestartet haben, wird der **`Home-Automation-Controller`** erstellt und die **`Home-Automation-Anwendung`** gestartet. Anschließend wird vom **`Home-Automation-Controller`** die Umgebung (Environment) erstellt.

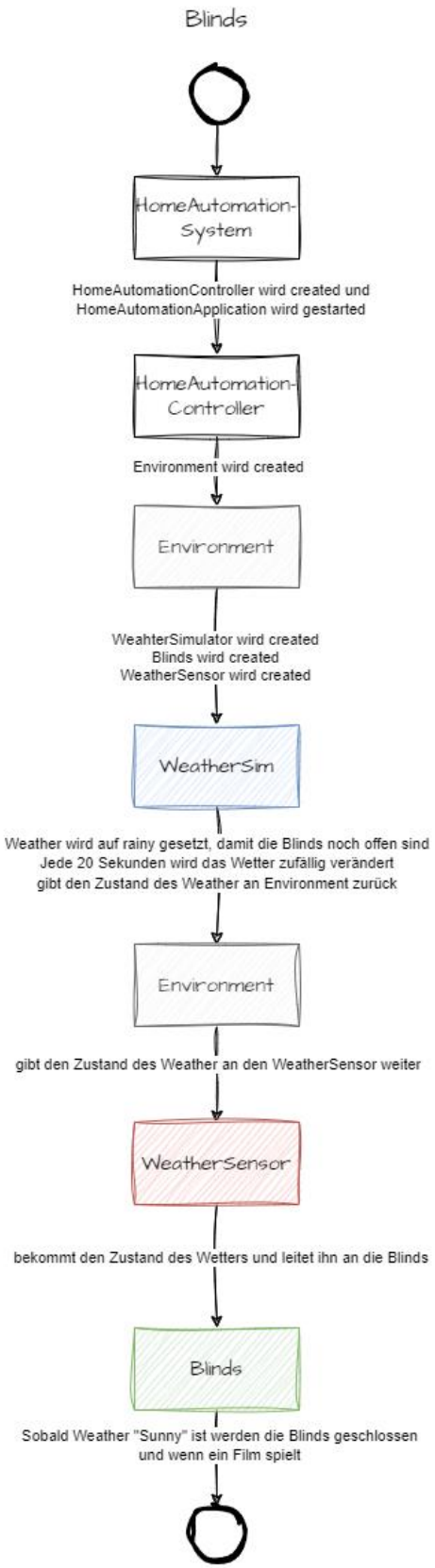
Unsere `Environment` Klasse repräsentiert die Umgebung und hier werden alle erforderlichen Komponenten erstellt. Zuallererst wird der `TemperaturSimulator`, der `TemperaturSensor` und der `Air-Conditioner` erstellt.

Wir haben uns dafür entschieden, die Temperatur in Celsius darzustellen. Die Starttemperatur beträgt 20°C und alle 5 Sekunden wird sie zufällig verändert. Man kann jedoch im Terminal mit dem Command `SetTemperature` die Temperatur verändern. Die geänderte Temperatur wird dann an das `Environment` gesendet, die sie an den `TemperaturSensor` weiterleitet.

Der TemperaturSensor liest die Temperatur ab und gibt sie an den `Air-Conditioner` weiter. Dieser überprüft die Temperatur und schaltet sich ein oder aus, abhängig davon, ob die Temperatur über 20°C liegt (wenn ja, wird die Klimaanlage eingeschaltet) oder unter 20°C (wenn ja, wird die Klimaanlage ausgeschaltet). Man kann den Air-Conditioner speziell auch ausschalten und einschalten mit dem Befehl `AirCondition on/off.`

Als nächstens haben wir das Weather definiert und die Blinds implementiert.

Lassen Sie uns nun gemeinsam jeden Schritt durchgehen, was genau mit den Jalousien passiert, wenn wir unser Programm starten. Es ist nicht viel anders zum Air-Conditioner.



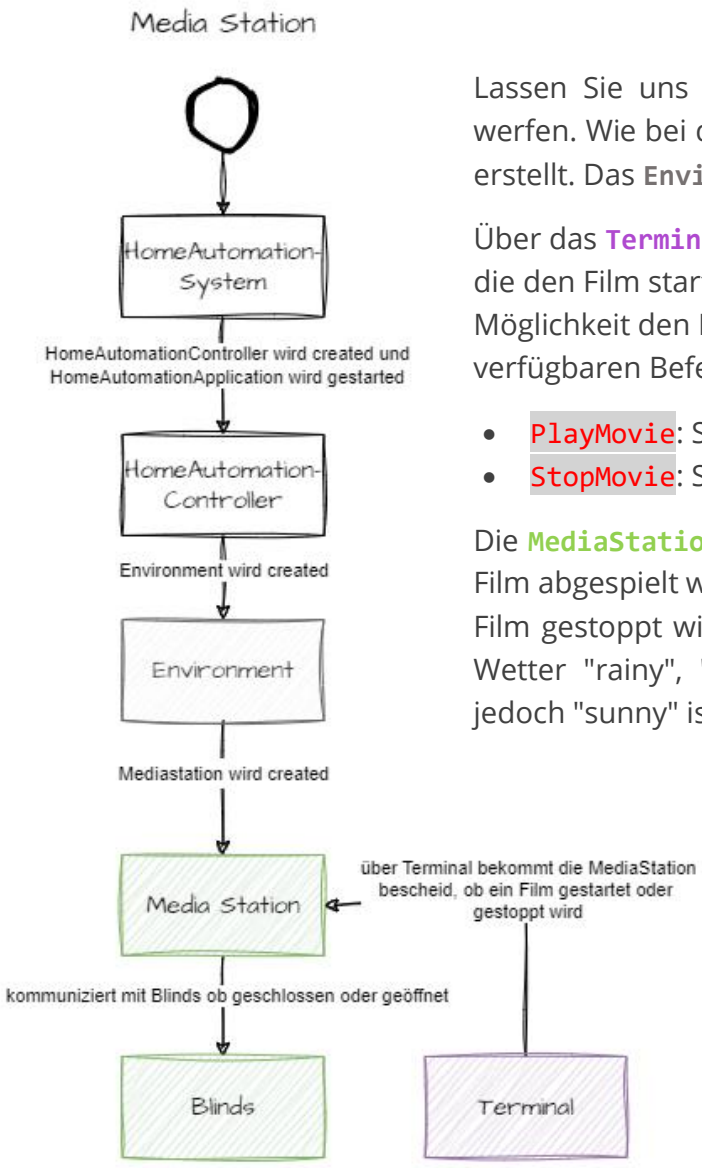
Wir starten das Programm und es wird der **Home-Automation-Controller** erstellt und die **Home-Automation-Anwendung** gestartet. Anschließend wird vom **Home-Automation-Controller** die Umgebung (Environment) erstellt.

Unsere `Environment` Klasse repräsentiert die Umgebung und hier werden alle erforderlichen Komponenten erstellt. Zuallererst wird der `WeatherSimulator`, der `WeatherSensor` und die `Blinds` erstellt.

Im `WeatherSimulator` ist der Standardwert für das Wetter auf "rainy" festgelegt. Mithilfe des Befehls `SetWeather` kann man das Wetter jedoch auf einen bestimmten Zustand umstellen lassen. Wir haben uns bewusst für „rainy“ entschieden, da in diesem Fall die `Blinds` noch geöffnet sind und es für uns einfacher war, Tests durchzuführen. Für den Zustand des Wetters haben wir uns für "sunny", "rainy", "snowy" und "cloudy" entschieden. Das Wetter soll alle 20 Sekunden zufällig geändert werden, und der neue Zustand wird an das `Environment` zurückgesendet.

Das `Environment` leitet den Zustand an den `WeatherSensor` weiter, der ihn an die `Blinds` weiterleitet. Die `Blinds` überprüfen nun das Wetter. Wenn es "rainy", "cloudy" oder "snowy" ist, werden die `Blinds` geöffnet, es sei denn, sie sind bereits geöffnet, dann bleiben sie in dieser Position. Wenn das Wetter jedoch "sunny" ist, werden die `Blinds` geschlossen. Beachten Sie jedoch, dass die Blinds auch geschlossen bleiben, wenn in der `MediaStation` ein Film abgespielt wird, und sie nach dem Wetter nicht mehr geöffnet werden, bis der Film beendet ist.

Als vorletztes haben wir die MediaStation gemacht.



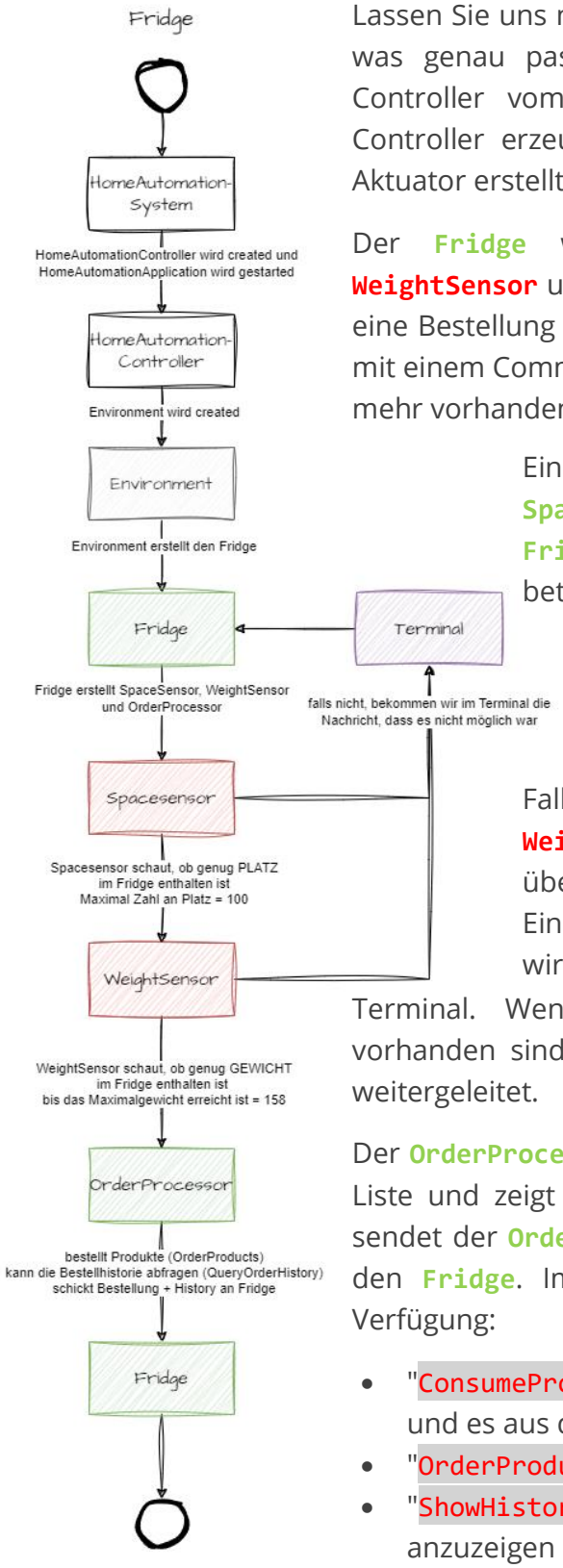
Lassen Sie uns nun einen kurzen Blick auf die MediaStation werfen. Wie bei den anderen Komponenten wird auch hier alles erstellt. Das `Environment` erstellt dann die `MediaStation`.

Über das `Terminal` können nun Befehle eingegeben werden, die den Film starten und stoppen lassen. Es gibt keine andere Möglichkeit den Film zu stoppen nur mit dem Befehl. Die verfügbaren Befehle lauten:

- `PlayMovie`: Startet die Wiedergabe eines Films.
- `StopMovie`: Stoppt die Wiedergabe des Films.

Die `MediaStation` kommuniziert mit den `Blinds`, und wenn ein Film abgespielt wird, bleiben die `Blinds` geschlossen. Sobald der Film gestoppt wird, werden die `Blinds` nur geöffnet, wenn das Wetter "rainy", "snowy" oder "cloudy" ist. Wenn das Wetter jedoch "sunny" ist, werden die `Blinds` geöffnet.

Zu guter Letzt widmen wir uns dem spannendsten Teil der Fridge.



Lassen Sie uns nun den `Fridge` genauer betrachten und sehen, was genau passiert. Zunächst wird der Home Automation Controller vom **`Home Automation System`** erstellt. Dieser Controller erzeugt das `Environment`, in dem der `Fridge` als Aktuator erstellt wird.

Der `Fridge` wiederum erstellt den **`SpaceSensor`**, den **`WeightSensor`** und den `OrderProcessor`. Es kann auf zwei Arten eine Bestellung aufgegeben werden, erstens über das Terminal mit einem Command und das zweite ist, wenn das Produkt nicht mehr vorhanden ist, wird es auch automatisch nachbestellt.

Eine Bestellung wird aufgegeben, indem der `SpaceSensor` überprüft, ob ausreichend Platz im `Fridge` vorhanden ist. Die maximale Kapazität beträgt 120 Einheiten.

Wenn nicht genügend Platz verfügbar ist, wird eine entsprechende Nachricht im Terminal angezeigt.

Falls jedoch genügend Platz vorhanden ist, wird der **`WeightSensor`** aktiviert, um das Gewicht im Fridge zu überprüfen. Das maximale Gewicht beträgt 158 Einheiten. Wenn das Gewichtslimit überschritten wird, erhalten Sie eine Benachrichtigung im Terminal. Wenn jedoch ausreichend Platz und Gewicht vorhanden sind, wird die Bestellung an den `OrderProcessor` weitergeleitet.

Der `OrderProcessor` tätigt die Bestellung, speichert sie in einer Liste und zeigt eine Rechnung im Terminal an. Anschließend sendet der `OrderProcessor` die Bestellung und die Historie an den `Fridge`. Im Terminal stehen verschiedene Befehle zur Verfügung:

- "`ConsumeProduct Name`": um ein Produkt zu konsumieren und es aus dem Fridge zu entfernen
- "`OrderProduct Name Amount`": um ein Produkt zu bestellen
- "`ShowHistory`": um die Liste der bereits bestellten Produkte anzuzeigen
- "`ShowContent`": um die Liste der aktuellen Produkte im Fridge anzuzeigen

Classdiagramm with dependencies and inner classes

Classdiagramm with dependencies, methods, properties and inner classes