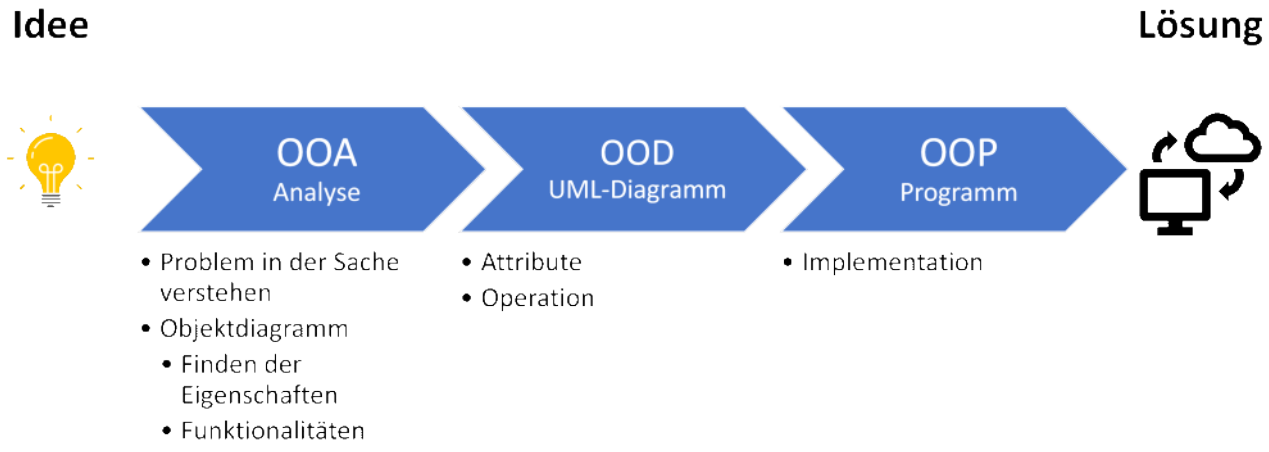
# Übung "SmartHome"

Montag, 17. Oktober 2022 11:24

## Auftrag

Sie haben den Auftrag, für einen Kunden, der Lösungen für die Hausautomatisierung entwickelt, ein Simulationsprogramm zu entwerfen sowie zu implementieren.
Dabei folgen Sie Schritt für Schritt dem nachfolgenden, in der OO üblichen, Vorgehensweise.




## Idee (Kundenwunsch)

Um das Verhalten verschiedener Aktoren zu testen, möchte die Firma Home-IT eine Software

entwickeln. Die Software soll eine Wohnung abbilden. Folgende Räume sollen modelliert und implementiert werden:

- Küche (Status Kochherd als Enum)
- Bad/WC (Feuchtigkeit als Prozentwert)
- Wohnzimmer (Farbe Ambientebeleuchtung als RGB-Wert)
- Schlafzimmer (Weckzeit als DateTime-Wert)
- Wintergarten (Sonneneinstrahlung in Lux)

In den Räumen soll es möglich sein, folgende Aktoren zu platzieren:

- Heizungsventil (jeder Raum ausser Wintergarten)
- Jalousiesteuerung (jeder Raum ausser Bad/WC)
- Markisensteuerung (Wintergarten)

Weiter muss ein Wettersensor entwickelt werden, der zufällige Wetterdaten generiert. Diese Daten werden dann an die Aktoren gesendet. Folgende Daten sind erforderlich:

- Aussentemperatur
- Windgeschwindigkeit
- Regen ja/nein

Es gelten folgende Regeln

- In jedem Zimmer kann eine Temperatur vorgegeben werden. Ist die Aussentemperatur tiefer als diese definierte Zimmertemperatur, öffnet sich das Heizungsventil und es wird geheizt.
- Die Markise wird ausgefahren, wenn die Aussentemperatur die vorgegebenen Zimmertemperatur überschreitet. Dies jedoch nur, wenn die Windgeschwindigkeit nicht höher als 30km/h beträgt.
- Die Jalousie wird automatisch runtergefahren, wenn die Aussentemperatur die vorgegebene Zimmertemperatur überschreitet. Dies jedoch nur, wenn sich keine Personen im Zimmer befinden.

Die Wetterdaten die vom Wettersensor generiert werden, sollen realistisch sein. D.h. bei der Temperatur ist darauf zu achten, dass ein Verlauf entsteht und diese nicht "wild" springt. Z.B. ist es nicht realistisch, dass nach einer 25°C eine Temperatur von 5°C gemessen wird.
Für die zeitliche Simulation soll der Wettersensor über eine Tick-Operation verfügen. Jeder Aufruf dieser Operation simuliert den Ablauf von 1s.

# Schritt 1: OOA - Analyse

## Die Funktionalität

Im ersten Schritt ist es wichtig, dass Sie die Aufgabe und deren Funktionsweise verstanden haben. Setzen Sie sich mit der
Aufgabe auseinander und notieren Sie sich allfällige Fragen. Identifizieren Sie die Akteure und überlegen Sie sich, wie diese sinnvoll in Klassen abgebildet werden können. Das Objektdiagramm gemäss Ablaufschema unter Auftrag/Idee lassen wir weg (haben wir nicht explizit besprochen). Notieren Sie sich auch Ihre Erkenntnisse (nebst den Fragen).

# Schritt 2: OOD - Klassendiagramm

## Die Operationen

Als nächstes erstellen Sie das Klassendiagramm.
Überlegen Sie sich für jede Klasse die für unsere Problemstellung nötigen Operationen.

| **Aufgabe** |
| --- |
| Überlegen Sie sich für jede Klasse die für unsere Problemstellung nötigen Operationen. Notieren Sie sich diese für jede Klasse. Überlegen Sie sich zu den Operationen ebenfalls die nötigen Parameter. |

Über Datentypen müssen Sie sich noch keine Gedanken machen - diese folgen im nächsten Schritt

### Die Datentypen, Sichtbarkeit und Eigenschaften

Jedes Attribut und jede Operation muss nun mit einer Sichtbarkeit versehen werden. Zudem müssen Datentypen definiert werden.

| **Aufgabe** |
| --- |
| Überlegen Sie sich für die Attribute sowie Operationen geeignete Sichtbarkeiten. Dazu gehört auch, einzelne Attribute mit den nötigen Merkmale wie readonly zu versehen.<br>Definieren Sie für jedes Attribut, jedem Parameter sowie Rückgabewert einen geeigneten Datentyp. |

### Klassendiagramm erstellen

Jetzt sind alle Informationen vorhanden, um einen ersten Entwurf für ein Klassendiagramm zu erstellen.

| **Aufgabe** |
| --- |
| Erstellen Sie nun ein erstes Klassendiagramm. Ergänzen Sie die Klasse mit alle nötigen Informationen: Name der Klasse, Attribute mit Datentyp und Sichtbarkeit sowie Merkmale, Operationen mit Parameter inkl. Datentyp, Rückgabeparameter sowie Sichtbarkeit. |

### Generalisierung / Abstraktion

Nun geht es darum, redundante Informationen in den einzelnen Klassen zu identifizieren. Diese können ggf. zu einer generalisierten Basisklasse zusammengefasst werden.

| **Aufgabe** |
| --- |
| Identifizieren Sie redundante Informationen in Ihrem Klassendiagramm und fassen Sie diese zu generalisierten Basisklassen zusammen. Überlegen Sie sich ebenfalls, welche Klassen sowie Methoden Sie ggf. abstract definieren können. |

## Schritt 3: OOP - C#-Programm

Nun geht es an die konkrete Umsetzung. Grundsätzlich war in Schritt 1 sowie Schritt 2 noch keine konkrete Sprache für die Implementierung im Spiel. D.h. diese konzeptionellen Schritte werden allgemein gelöst. Das Resultat in Form eines Klassendiagramms kann dann in einer beliebigen Sprache umgesetzt werden. Wir wählen für unsere Umsetzung natürlich C#

| **Aufgabe** |
| --- |
| Implementieren Sie Ihr Klassendiagramm mit C#. Erstellen Sie dazu ein neues Projekt vom Typ "Console App". Die einzelnen Methoden sollen bei einer konkreten Aktion geeignete Ausgaben auf der Konsole vornehmen. Implementieren Sie ebenfalls die Main-Methode in der die Wohnung mit allen nötigen Komponenten erstellt wird. Danach simulieren Sie eine Zeitspanne von 1h. D.h. Sie rufen die Tick-Methode des Wettersensors 60x auf. Anschliessend erstellen Sie von ihrer Konsole einen Printscreen mit allen Ausgaben und legen diese auf Ihrer OneNote-Seite ab. |

## Schritt 4: Testing

Nun soll das System getestet werden. Damit dies möglich wird, muss der Wettersensor ausgewechselt werden, damit dieser nicht zufällige sondern deterministische Wetterdaten generiert. Dies erreichen wir, indem wir den Wettersensor der Wohnung mit dem IoC-Konzept erzeugen. D.h. wir können für Testzwecke für den Wettersensor durch einen Mock ersetzen, der wie erwähnt keine

zufälligen sondern deterministische Wetterdaten erzeugt.

| **Aufgabe** |
| --- |
| Erstellen Sie ein Testprojekt. Anschliessend implementieren Sie für den Wettersensor einen Mock. Nun erstellen Sie eine Testklasse. Diese soll eine Methode beinhalten, welche das System erzeugt (analog der Main-Methode). Diese Methode können Sie mit dem Attribut [TestInitialize] versehen. D.h. diese Methode wird ausgeführt, bevor die Tests ausgeführt werden. Anschliessend überlegen Sie sich geeignete Tests. Dabei ist es wichtig, dass Sie Grenzwerte testen. D.h. wenn z.B. Temperaturvorgabe im Wintergarten 25°C ist, dann ist es nicht interessant, wenn Sie 5 Tests schreiben um zu prüfen, ob die Markise bei 10°C, 12°C, 15°C, 20°C nicht ausfährt. Viel interessanter ist der Test, was z.B. bei 24°C, 25°C, 26°C, -25°C, 100°C passiert. |

## Schritt 5: Abschluss

Sie haben jetzt ein Softwareprojekt von A-Z durchgeführt. Bereinigen Sie Ihre Solution, erstellen Sie eine ZIP-Datei und legen Sie diese auf OneNote ab.