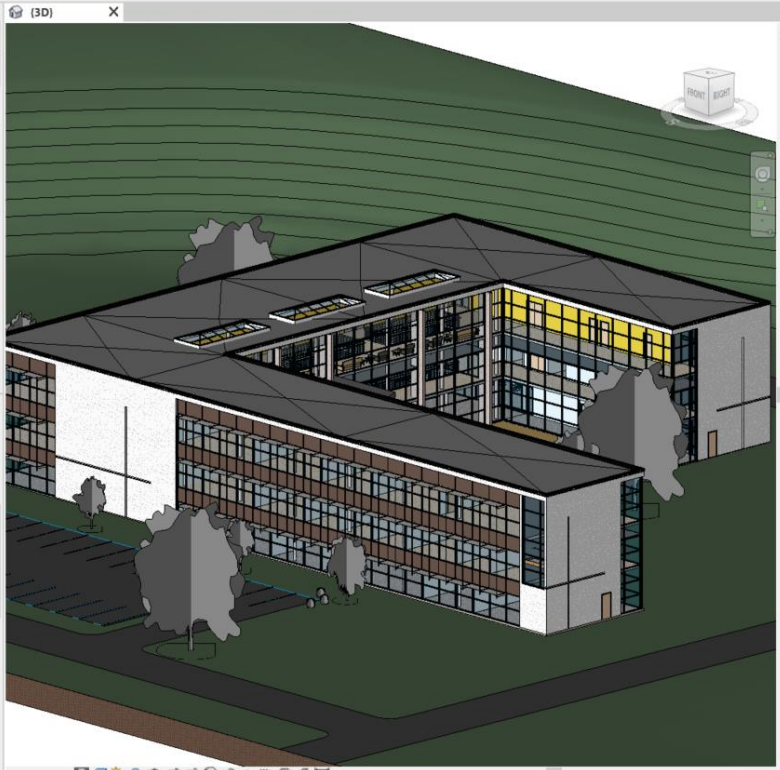


# Demonstrator-Anwendung



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

## Automatische Erstellung der Raumliste aus der Modellierungssoftware Autodesk Revit



The screenshot displays the Autodesk Revit interface. On the left, a 3D perspective view of a modern building with a glass facade and a flat roof is shown. The building is situated on a green landscape with some trees. On the right, the 'Room Schedule' table is open, showing a list of rooms and their areas. The table has four columns: A (Level), B (Number), C (Name), and D (Area). The rooms are listed in ascending order of level and number.

A	B	C	D
Level	Number	Name	Area
01 - Entry Level	101	Vest.	41 m²
01 - Entry Level	102	Lobby	327 m²
01 - Entry Level	103	Conference	47 m²
01 - Entry Level	104	Instruction	48 m²
01 - Entry Level	105	Instruction	97 m²
01 - Entry Level	106	Instruction	48 m²
01 - Entry Level	107	Comdor	137 m²
01 - Entry Level	108	Instruction	89 m²
01 - Entry Level	109	Women	13 m²
01 - Entry Level	110	Men	14 m²
01 - Entry Level	111	Lounge	38 m²
01 - Entry Level	112	Electrical	7 m²
01 - Entry Level	114	Stair	19 m²
01 - Entry Level	115	Instruction	127 m²
01 - Entry Level	116	Conference	32 m²
01 - Entry Level	117	Instruction	49 m²
01 - Entry Level	118	Electrical	17 m²
01 - Entry Level	119	Sprinkler	9 m²
01 - Entry Level	120	Lounge	41 m²
01 - Entry Level	121	Cafeteria	147 m²
01 - Entry Level	122	Prep/Dish	22 m²
01 - Entry Level	123	Conference	42 m²
01 - Entry Level	124	Dry Storage	8 m²
01 - Entry Level	125	Electrical	6 m²
01 - Entry Level	126	Admin	16 m²
01 - Entry Level	127	Office	15 m²
01 - Entry Level	128	Storage	10 m²
01 - Entry Level	129	Toilet	6 m²
01 - Entry Level	130	Stair	19 m²
01 - Entry Level	131	Comdor	55 m²
01 - Entry Level	132	Stair	19 m²
02 - Floor	201	Stair	19 m²
02 - Floor	202	Instruction	31 m²
02 - Floor	203	Computer Lab	32 m²
02 - Floor	204	Instruction	48 m²
02 - Floor	205	Instruction	32 m²
02 - Floor	206	Lounge	32 m²
02 - Floor	207	Copy/Print	32 m²
02 - Floor	208	Drafting	48 m²
02 - Floor	209	Computer Lab	73 m²
02 - Floor	210	Men	13 m²
02 - Floor	211	Women	14 m²
02 - Floor	212	Lounge	38 m²
02 - Floor	214	Electrical	7 m²
02 - Floor	216	Lobby	285 m²
02 - Floor	217	Stair	10 m²

1. Anforderungen analysieren
2. Erstellen von UML Diagrammen zur Konzeptionierung der Software
3. Implementierung der Klassen und Logikfunktionen
4. Implementierung eines Benutzerinterfaces zum Einsehen und Bearbeiten der Ergebnisse
5. Anbindung an die Modellierungssoftware Autodesk Revit 2019 zum Austauschen relevanter Daten

- Anwender: Architekt
- Ablegen von Bauwerken und Stockwerken
- Ablegen von Räumen
  - Verschiedene Raumarten (Büro, Wohnen, ...)
  - Raumbezeichnung &-fläche als Attribute
- Automatische Erstellung der Raumliste
- Berechnen der gesamten Raumfläche (pro Stockwerk und pro Bauwerk)

(Nur Stichwörter, für Abgabe nicht ausreichend!! )

# Nicht-funktionale Anforderungen

- Antwortzeit  $< 0,5$  s
- Das System muss eine hohe Verfügbarkeit aufweisen
- Das System soll auf einer Windows-Plattform laufen
- Das System soll für Gelegenheitsbenutzer einfach zu bedienen sein
- Das System soll bei Problemen sinnvolle Fehlermeldung den Nutzern zurückgeben

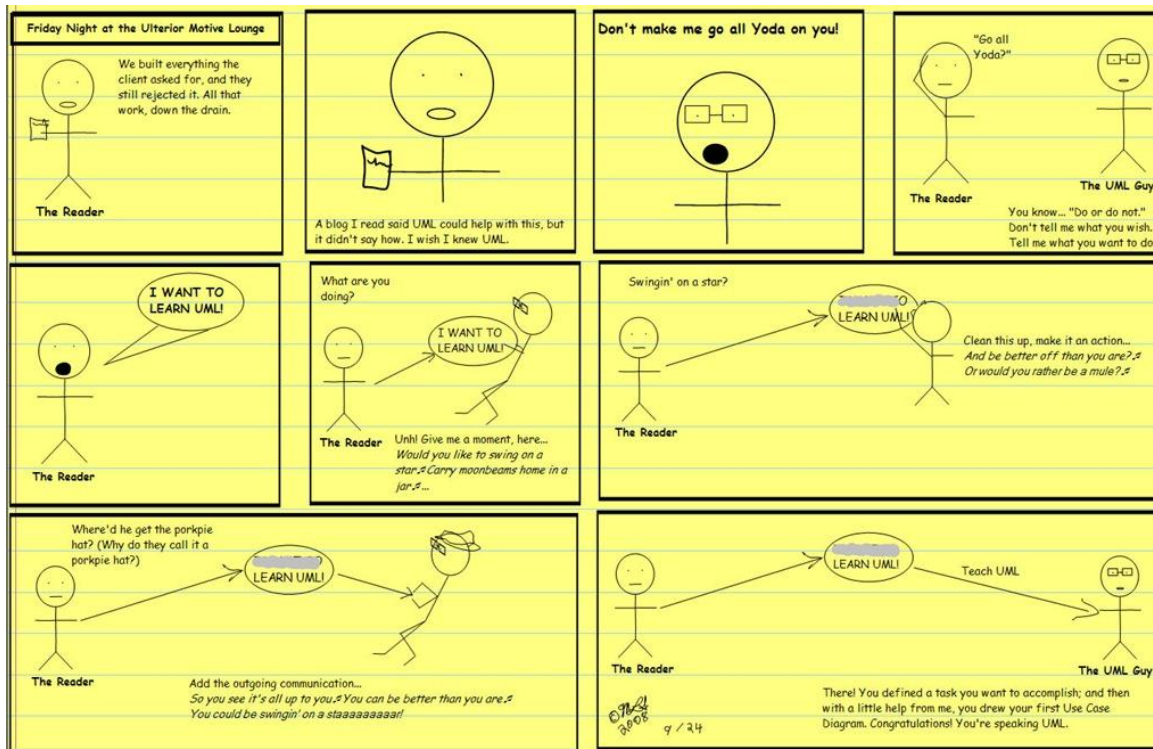
**Beobachtungsauftrag  
für die nächsten Folien  
– Wie könnten Use-  
Case und Klassen-  
diagramm aussehen??**

# Gruppenaufgabe



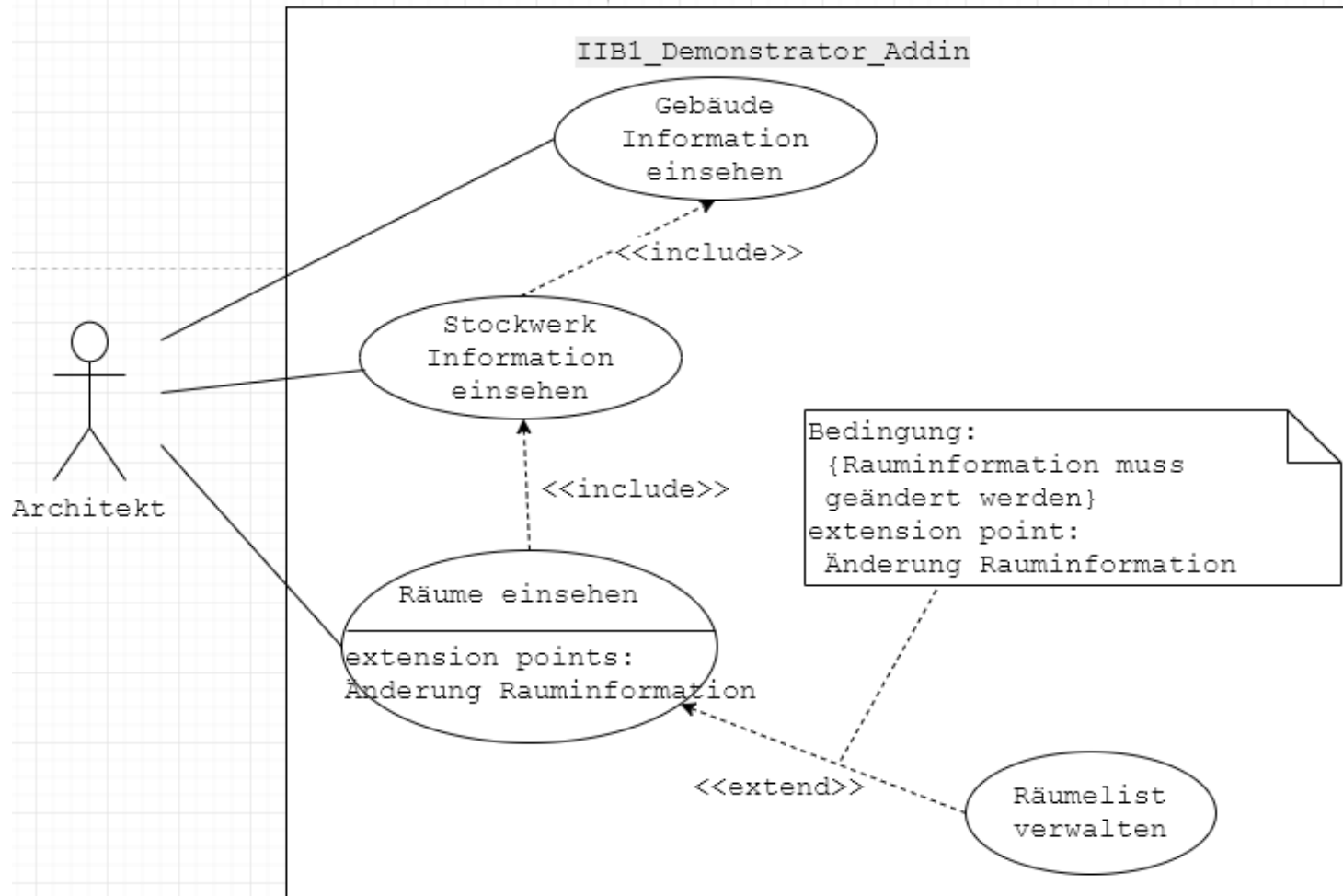
TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

## Erstellt ein Use-Case und ein Klassendiagramm für die Demonstrator-Anwendung

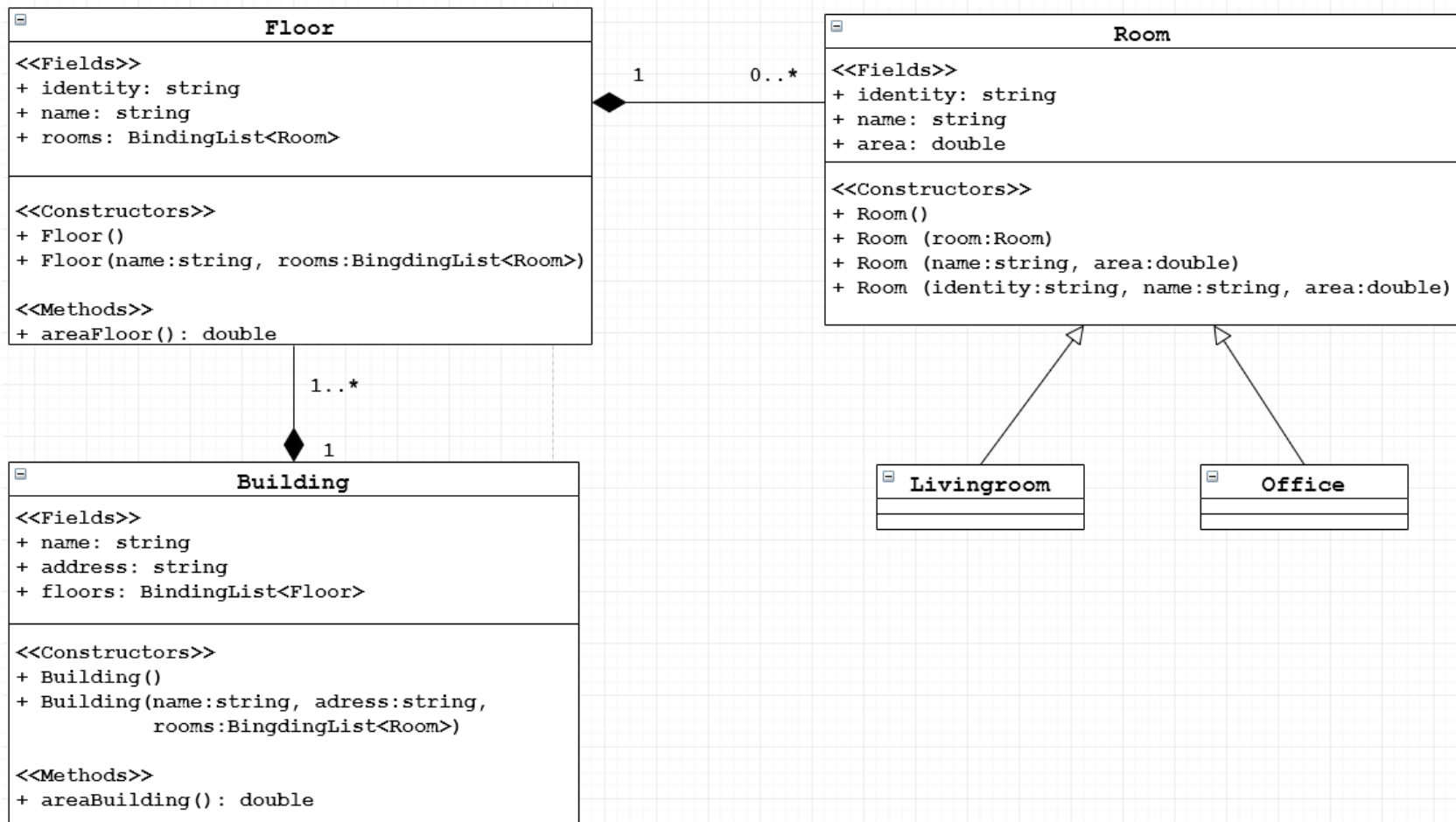


Dauer: 5 Minuten

# Lösung: Use-Case Diagramm

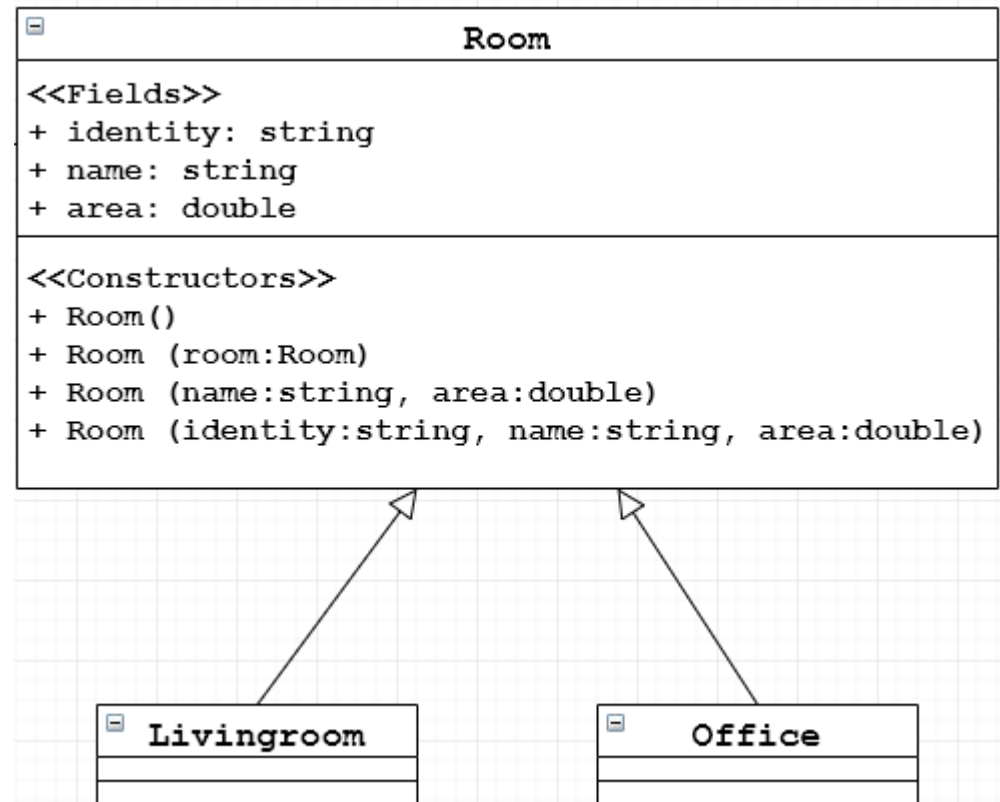


# Lösung: Klassendiagramm



Erzeugt eine Demonstrator-Klassenbibliothek und Klasse im Quellcode

- Klassenbibliothek erstellen „IIB1\_Demonstrator\_ClassLibrary“
- Ordner „Classes“ erstellen.
- „Room“ Class im Ordner hinzufügen. Attribut, Property und Konstruktor(en) erstellen
- Klassen „Livingroom“ & „Office“ erstellen (ohne Attribute bzw. Properties)
- Relation „Vererbung“ einzeichnen





# Lösung (1/2)



```
public class Room
{
    #region Fields and Properties
    // fields
    private string identity;
    private string name;
    private double area;

    // Properties
    public string Name { get => name; set => name = value; }
    public double Area { get => area; set => area = value; }
    public string Identity { get => identity; set => identity = value; }
    #endregion

    #region constructors
    /// Default Constructor
    public Room()
    {
    }
}
```

```
/// Custom Constructors

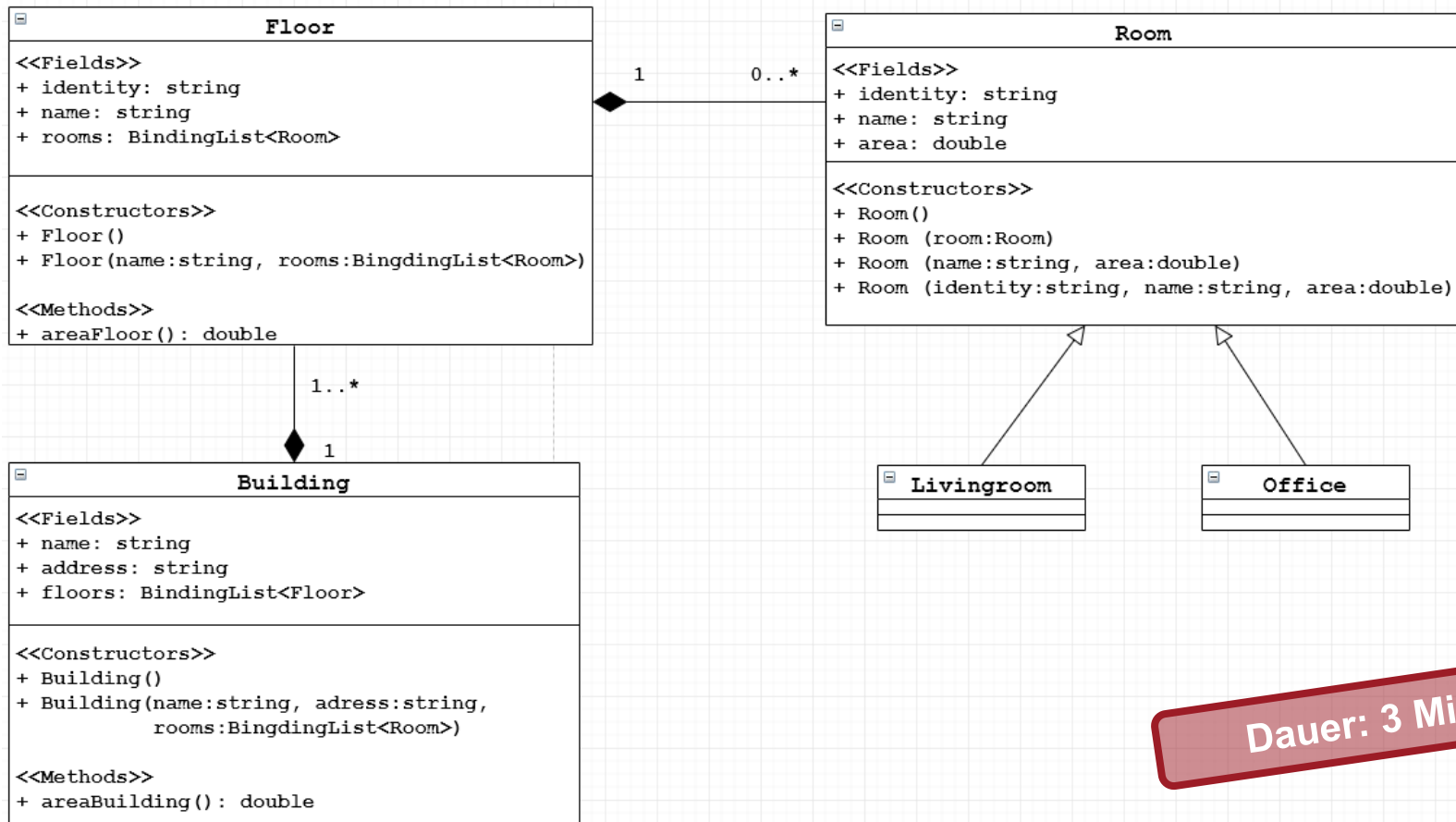
public Room ( string identity, string name, double area )
{
    this.Identity = identity;
    this.Name = name;
    this.Area = area;
}

public Room(Room room)
{
    this.Identity = room.Identity;
    this.Name = room.Name;
    this.Area = room.Area;
}

public Room (string name, double area)
{
    this.Name = name;
    this.Area = area;
}

#endregion
}
```

## Erzeugt Klasse „Building“ und „Floor“ im Klassendiagramm



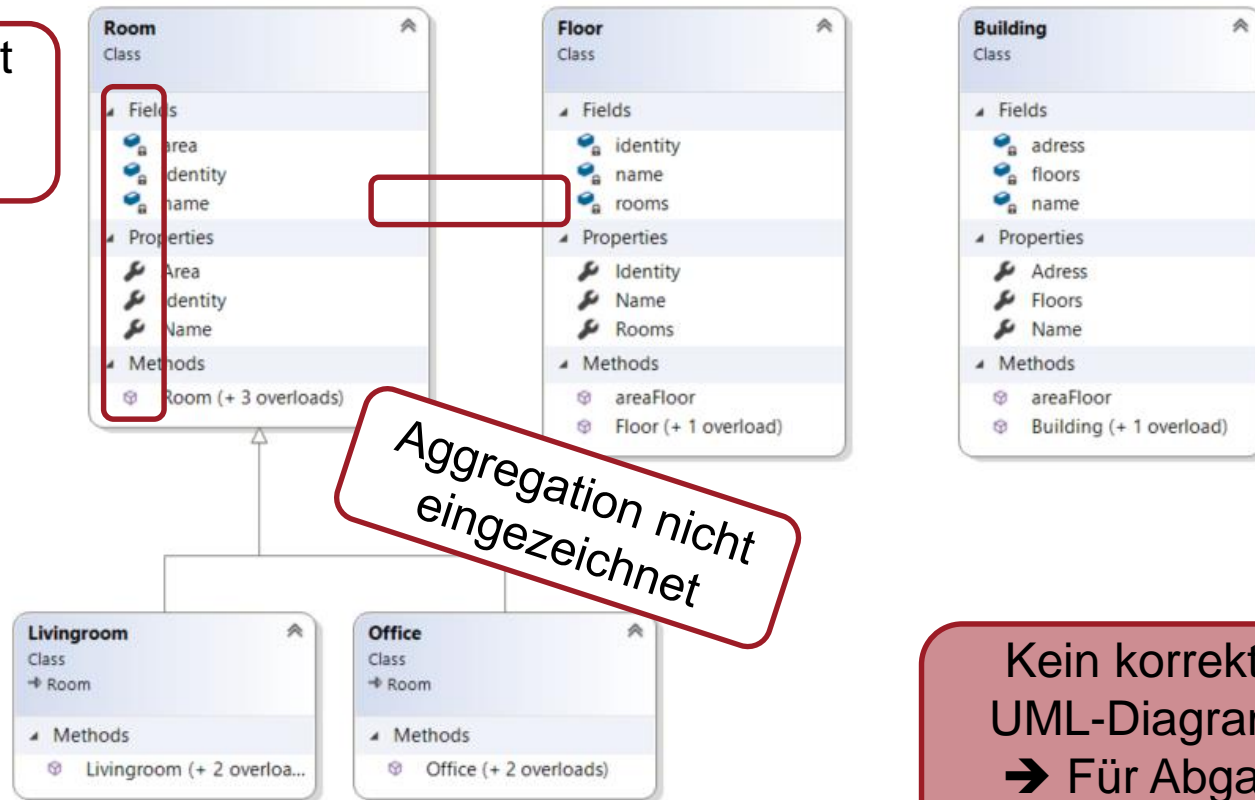
Dauer: 3 Minuten

# Lösung:



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Sichtbarkeit nicht  
aus Diagramm  
erkennlich



Keine Umlaute  
verwenden!!

Aggregation nicht  
eingezeichnet

Kein korrektes  
UML-Diagramm  
→ Für Abgabe  
nicht ausreichend!

## Zu implementierende Methoden

- (1) `areaFloor()` : double
  - Iteriert über alle Räume auf der gleichen Etage
  - Summiert Raumflächen auf
- (2) `areaBuilding()` : double
  - Iteriert über alle Stockwerke im gleichen Gebäude
  - Ruft die Methode `areaFloor()` auf und summiert Stockwerkflächen auf

Arbeiten mit  
einer Liste

Schleifen

Aufbau einer  
Methode

Kontroll-  
strukturen

# Lösung: Methode areaFloor()

```
public double areaFloor ()
{
    double floorArea = 0;
    // Ckeck if Field "rooms" is already initialized
    if (rooms != null)
    {
        //Interate all Rooms on one Floor
        foreach (Room r in rooms)
        {
            // Add up room area; Short for a = a + b
            floorArea += r.Area;
        }
    }
    else
    {
        Console.WriteLine("Field rooms is not initialized! ");
    }
    return floorArea;
}
```

Aufbau einer  
Methode

Arbeiten mit  
einer Liste

Schleifen

If-else Kontroll-  
strukturen

# Lösung: Methode areaBuilding()



Ähnliche Struktur wie areaFloor():

```
public double areaBuilding()
{
    double buildingArea = 0;

    // Check if Field "floors" is already initialized
    if (floors != null)
    {
        //Iterate all Floors in one Building
        foreach (Floor f in floors)
        {
            // Add up floor area; Short for a = a + b
            buildingArea += f.areaFloor();
        }
    }
    else
    {
        Console.WriteLine("Field floors is not initialized! ");
    }
    return buildingArea;
}
```