UNIVERSITY OF TWENTE.

Class design

Topic of Software Systems (TCS module 2)

Lecturer: Tom van Dijk

CLASS DESIGN

- A lot of “designing your classes” is experience
- Beginner’s rule: distinguish nouns and verbs
  - nouns => classes
  - verbs => methods
- Rule of thumb: keep classes simple and with few responsibilities
- Apply appropriate design patterns (later topics)

- Think before you code, or you have to fix a lot of mistakes

NOUNS VS VERBS

- Start with a system specification
- Nouns often are concepts; verbs are often actions
- “The customer books a room in the hotel”
  - Nouns: customer, room, hotel
  - Verbs: books
- “The player hits the monster with a crossbow”
  - Nouns: player, monster, crossbow
  - Verbs: hits

EXAMPLE: HOTEL INFORMATION SYSTEM

Initial requirements:

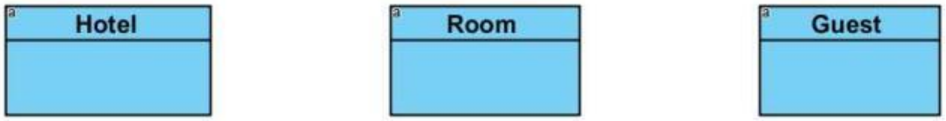
- System to record guests of a hotel, including their name and in which room they stay

What concepts do we need?

- Guest
- Hotel
- Name
- Room

EXAMPLE: HOTEL INFORMATION SYSTEM

First design step: class diagram



Ultimately, program manipulates objects

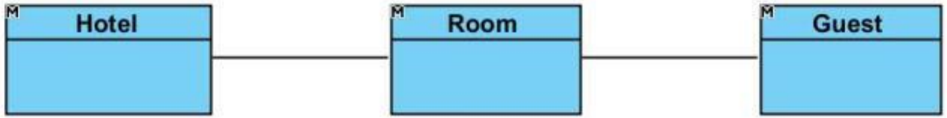
- Objects represent specific hotels, rooms and guests
- Examples
  - ‘Hotel Fawlty Towers’
  - ‘Room 101’, ‘Room 102’, etc.
  - ‘Major Gowen’, ‘Miss Tibbs’, etc.

EXAMPLE: HOTEL INFORMATION SYSTEM

What relations can be defined between these concepts?

- Hotel *has* Rooms, Room *belongs* to a Hotel
- Guest *occupies* a Room, Room *has* Guest

Second design step: extend class diagram with associations

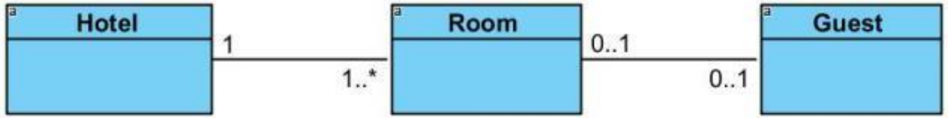


EXAMPLE: HOTEL INFORMATION SYSTEM

Multiplicities: how many of these are there?

- Hotel → Room: many; Room → Hotel: exactly one
- Guest → Room: zero or one, Room → Guest: zero or one

Third design step: extend class diagram with multiplicities

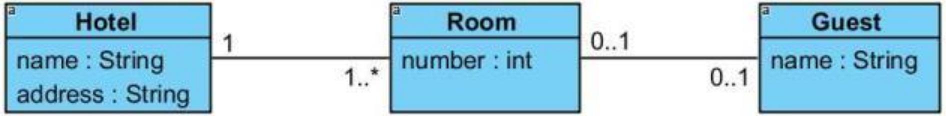


EXAMPLE: HOTEL INFORMATION SYSTEM

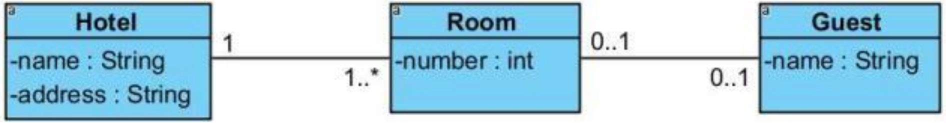
What properties do our concepts have?

- Hotel: name (a String), address (a String)
- Room: number (an int)
- Guest: name (a String)

Fourth design step: extend class diagram with attributes



And extend attributes with visibility indicators (private, of course)

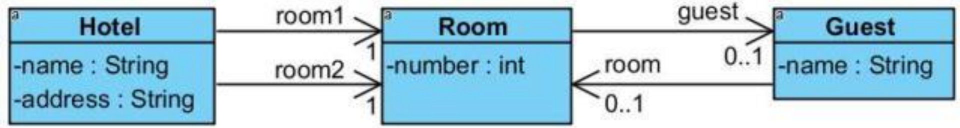


EXAMPLE: HOTEL INFORMATION SYSTEM

Make a choice which associations to code up

- Does a hotel “know” its rooms?
- Does a room “know” its hotel?
- Does a room “know” its (optional) guest?
- Does a guest “know” his room?

Fifth design step: named & directed associations



For simplification, our `Hotel` now has exactly 2 `Rooms`

These are *our* answers here, but not the only or (necessarily) best ones

In fact, there is hardly ever a single or absolutely best choice

UNIVERSITY OF TWENTE.

EXAMPLE: HOTEL INFORMATION SYSTEM

Every class is responsible for part of the action

- For this purpose, classes have methods
- Queries: reveal some of the internal state
- Commands: change the internal state

Examples

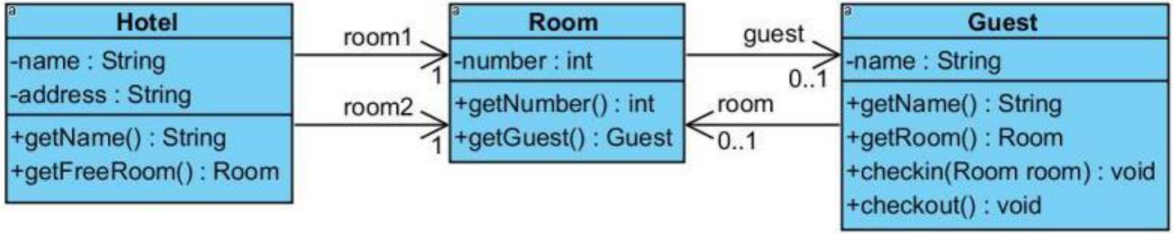
- For Hotel: what’s is its name? Is there a free Room? (queries)
- For Room: what’s is its number, etc. (queries)
- For Guest: check into a Room (command)

EXAMPLE: HOTEL INFORMATION SYSTEM

Examples

- For Hotel: what’s is its name? Is there a free Room? (queries)
- For Room: what’s is its number, etc. (queries)
- For Guest: check into a Room (command)

Sixth design step: show (public) operations


Hotel
-name : String
-address : String
+getName() : String
+getFreeRoom() : Room
room1
1
room2
1
Room
-number : int
+getNumber() : int
+getGuest() : Guest
guest
0..1
room
0..1
Guest
-name : String
+getName() : String
+getRoom() : Room
+checkin(Room room) : void
+checkout() : void