Introduction to UML

Mathias Stocker

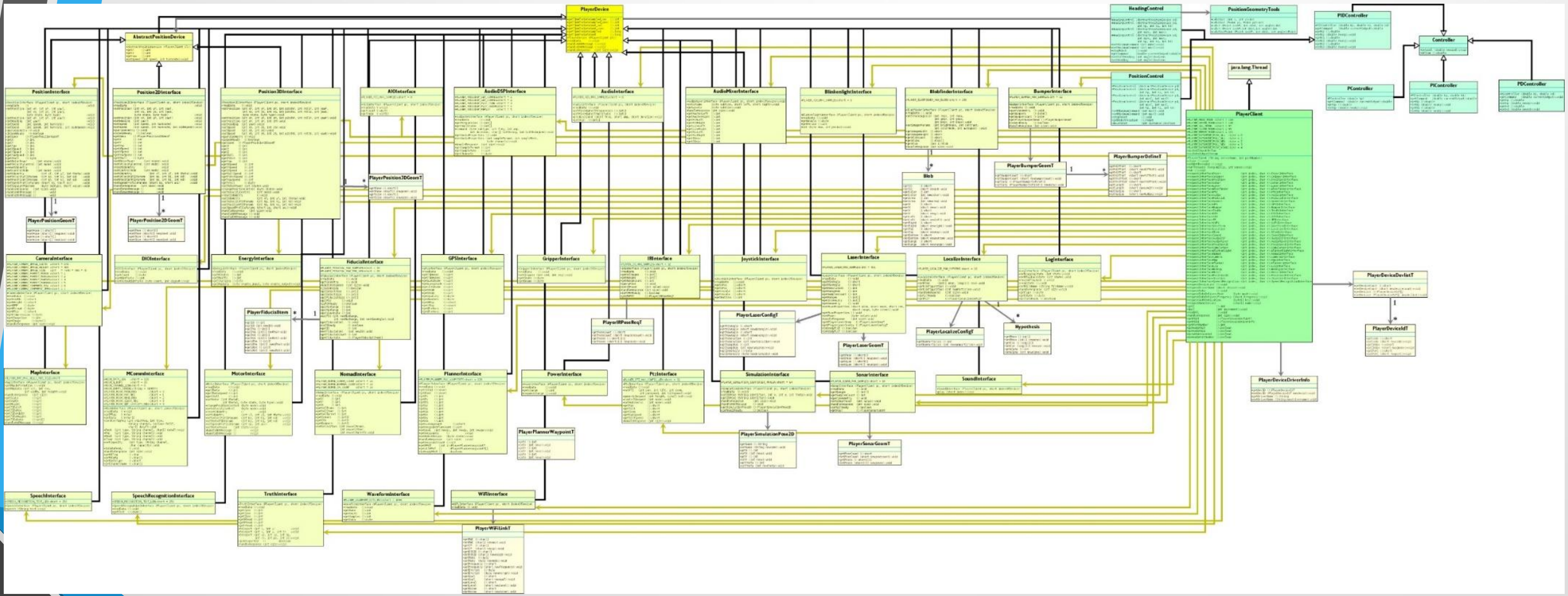
Main areas of application

- Documentation
- Drafts

Documentation

- Can be done automatically
- Can be an «overkill»

Documentation



source: java-player.sourceforge.net

A draft helps you to…

… simplify reality

… unterstanding an existing solution

… deciding how to build something from scratch

… capture requirements and discuss your idea with others

… reduce your effort to test different approaches

Modeling your system…

structure

- class diagram
- component diagram
- composite structure diagram
- object diagram
- package diagram
- profile diagram

behaviour

- activity diagram
- comunication diagram
- interaction overview diagram
- sequence diagram
- state machine diagram
- timing diagram

Modeling your system...

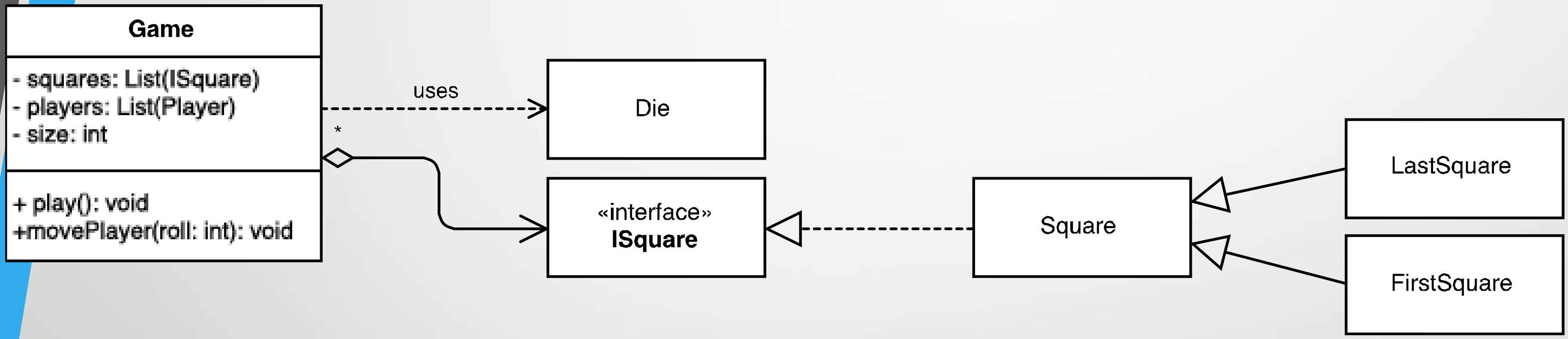
structure

- class diagram
- component diagram
- composite structure diagram
- object diagram
- package diagram
- profile diagram

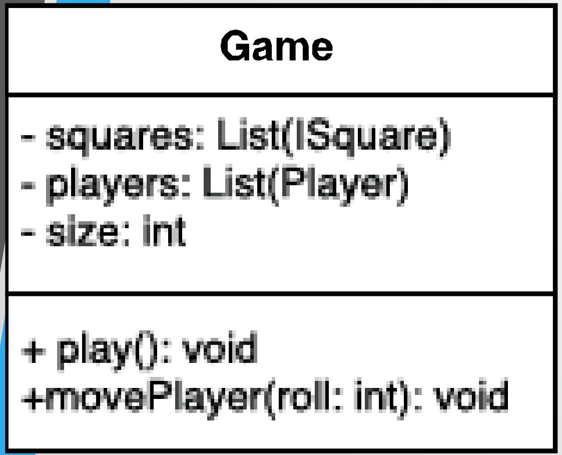
behaviour

- activity diagram
- comunication diagram
- interaction overview diagram
- sequence diagram
- state machine diagram
- timing diagram

Class diagram



Classes and Interfaces



Game

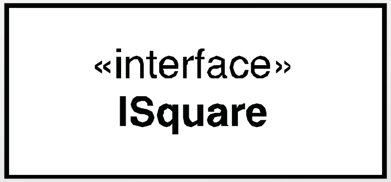
- squares: List(ISquare)
- players: List(Player)
- size: int

+ play(): void
+movePlayer(roll: int): void

Name

Attributes

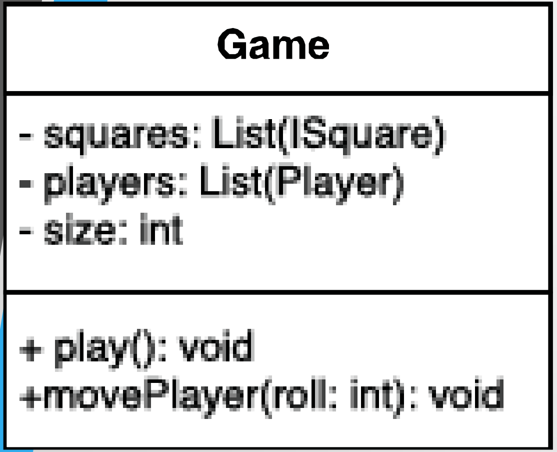
Methods



«interface»
ISquare

Interface annotation

Classes and Interfaces



Game

- squares: List(ISquare)
- players: List(Player)
- size: int

+ play(): void
+movePlayer(roll: int): void

Access modifiers

+ public, - private, # protected, static

Attributes

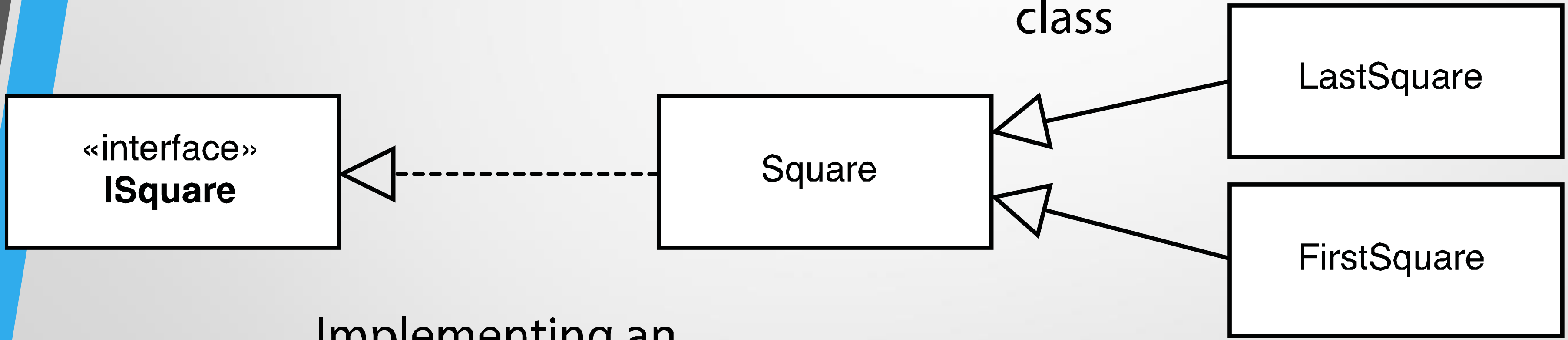
acessIdentifier: type

Example: - size: int

Methods

accessIdentifier(parameter: type): returnType

Implementation and extension



Extending a class

«interface»
ISquare

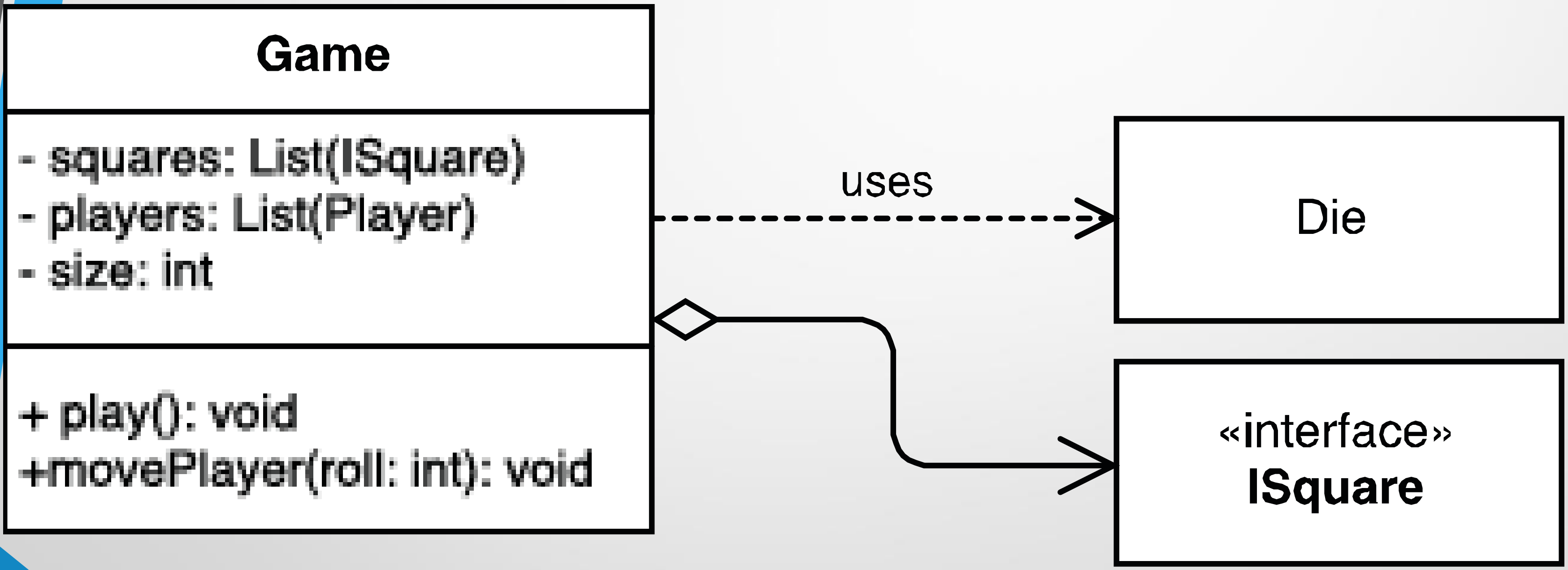
Square

LastSquare

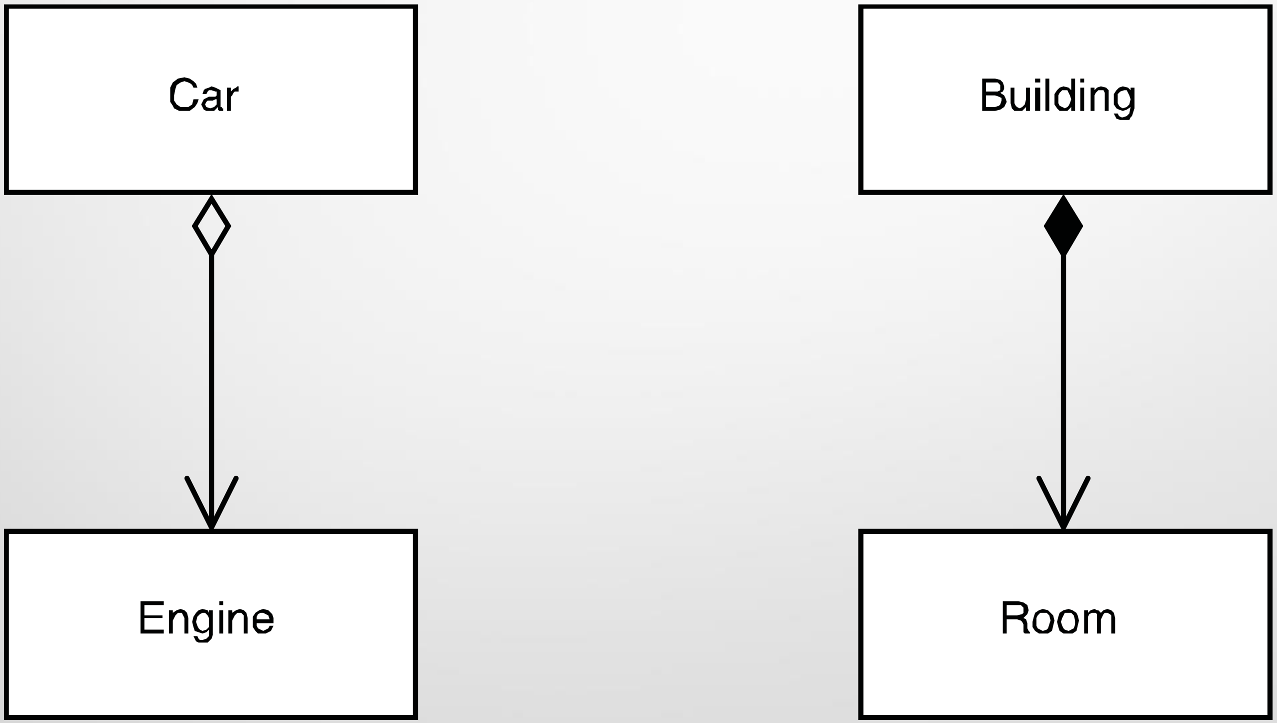
FirstSquare

Implementing an interface

Dependency


Game
- squares: List(ISquare)
- players: List(Player)
- size: int
+ play(): void
+movePlayer(roll: int): void
uses
Die
«interface»
ISquare

Aggregation vs. Composition



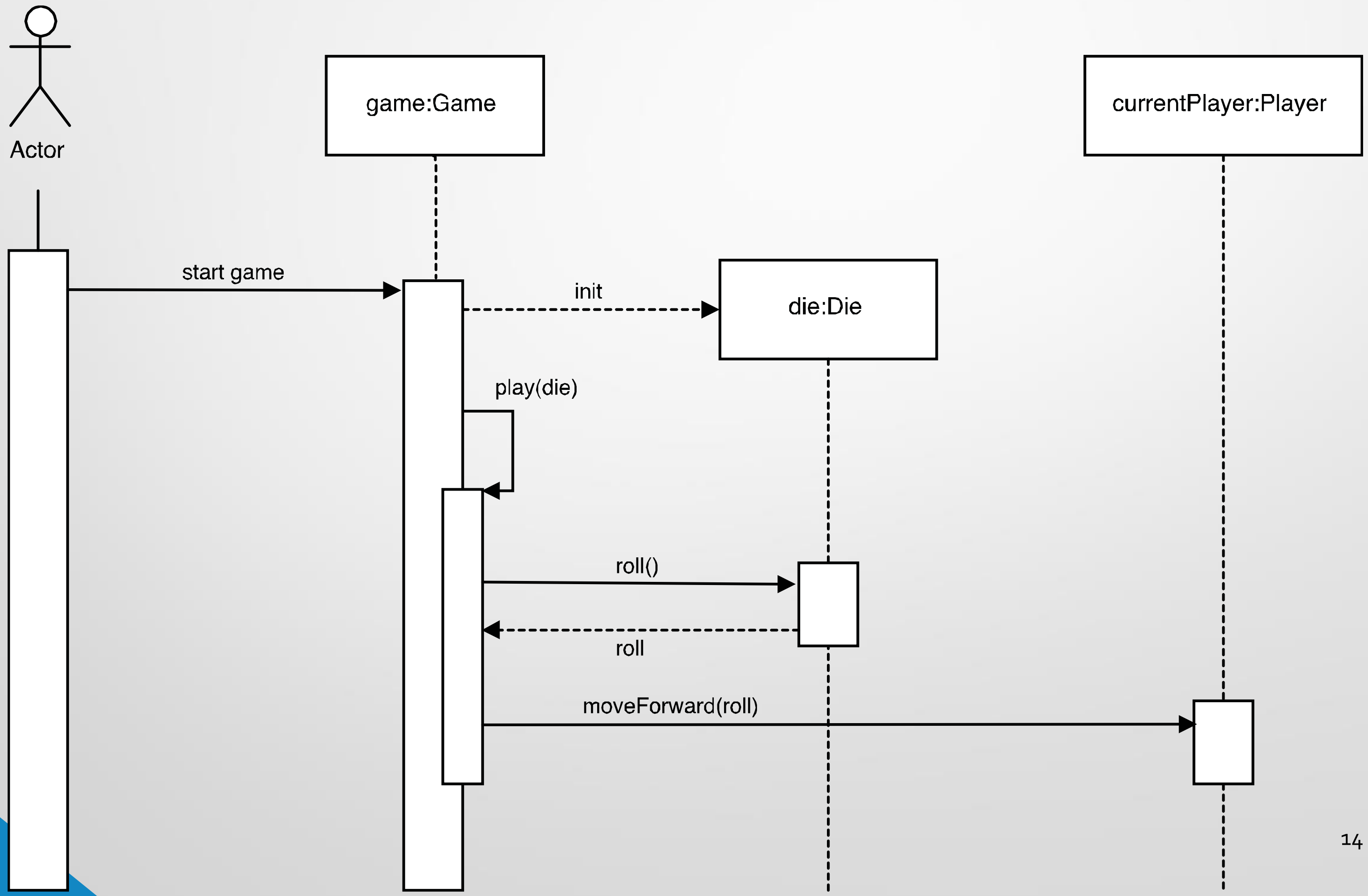
Car

Engine

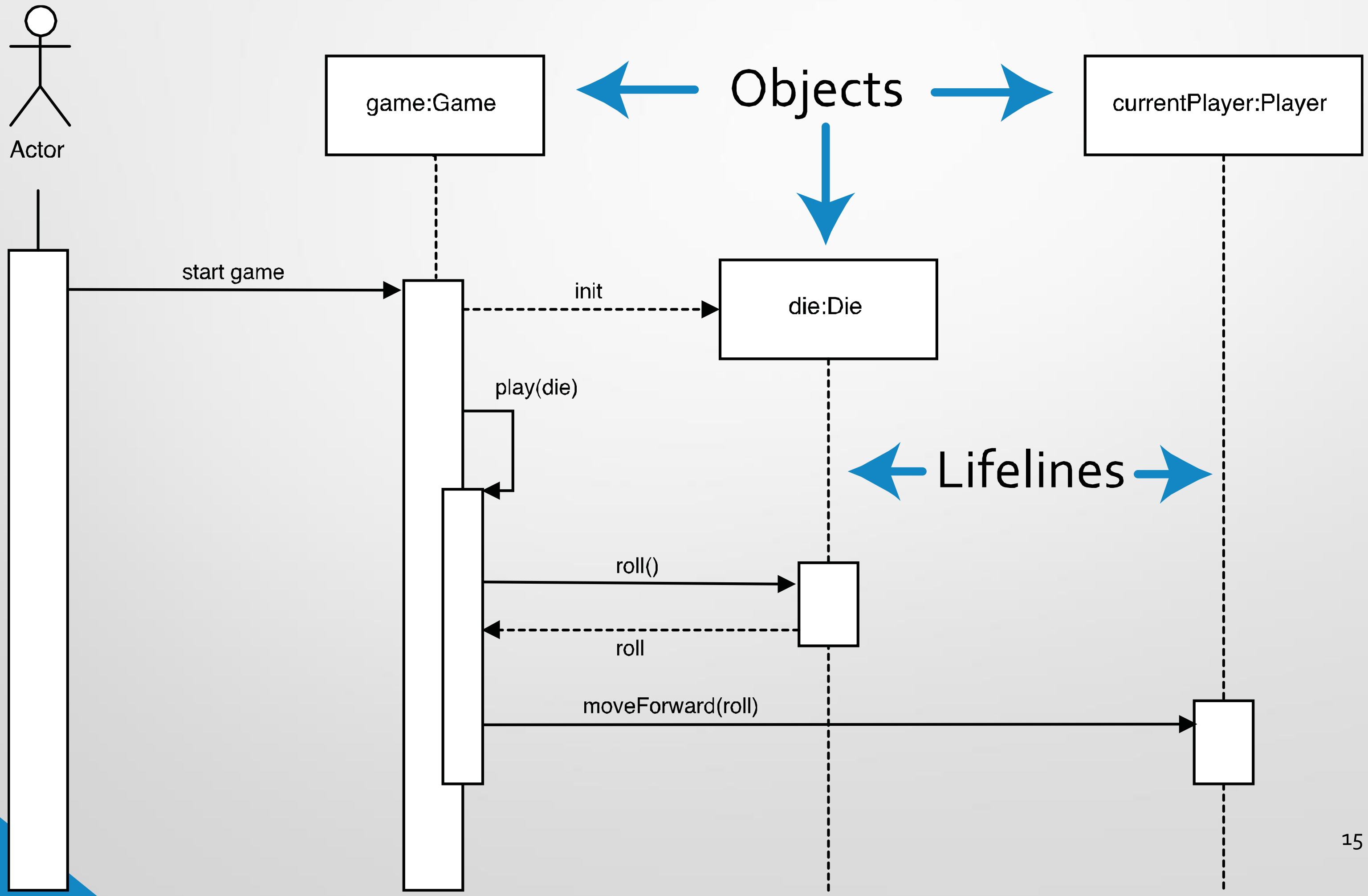
Building

Room

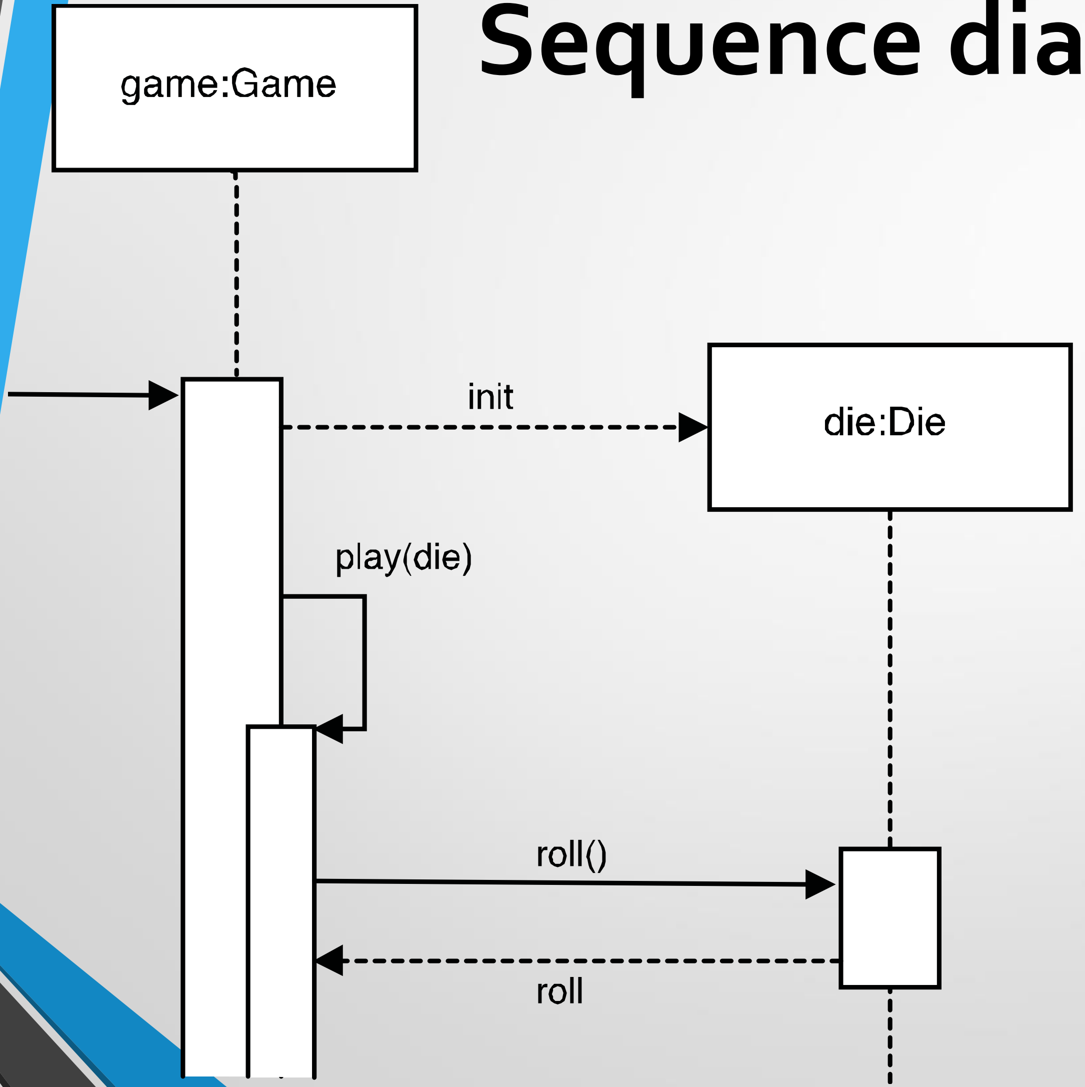
Sequence diagram



Sequence diagram



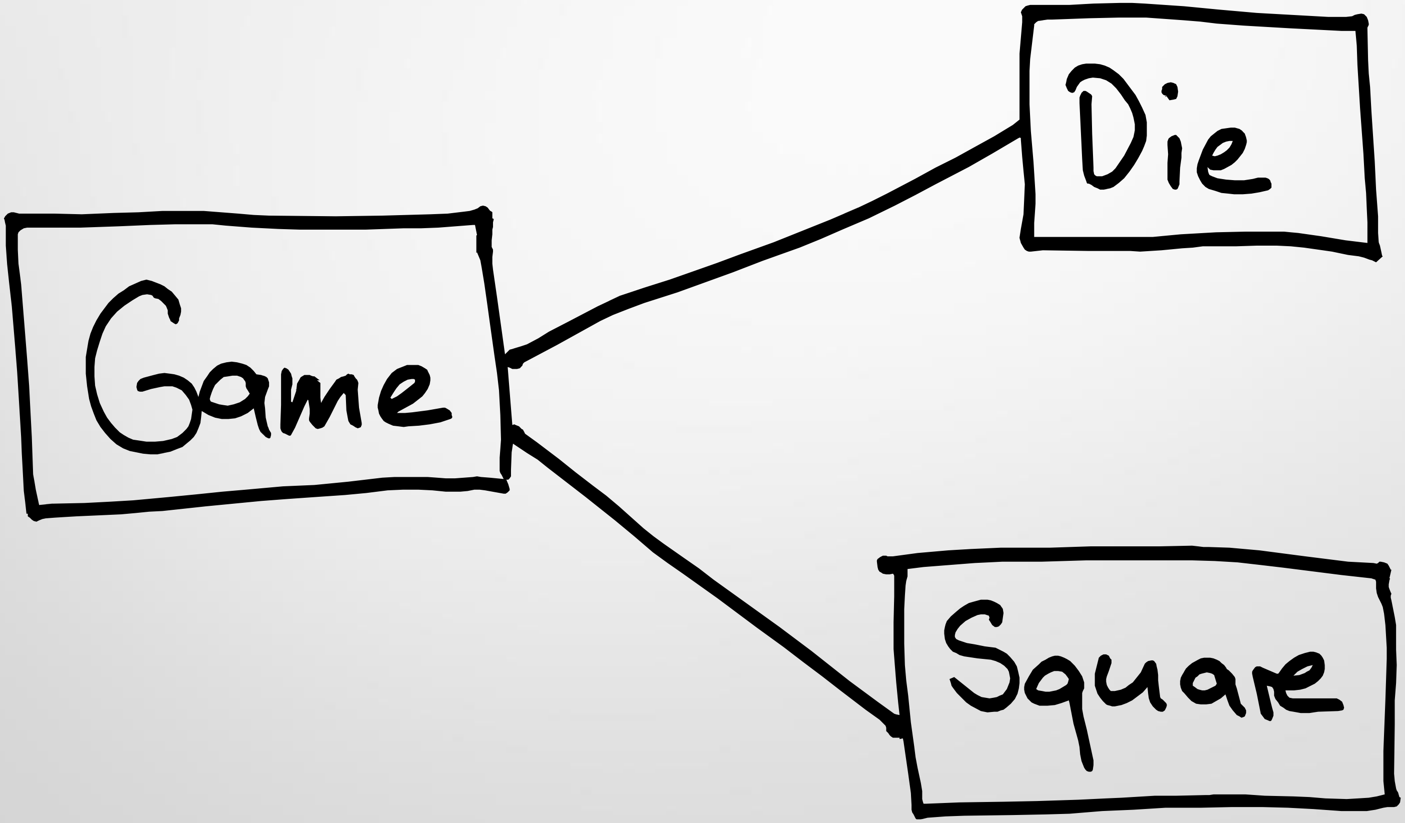
Sequence diagram



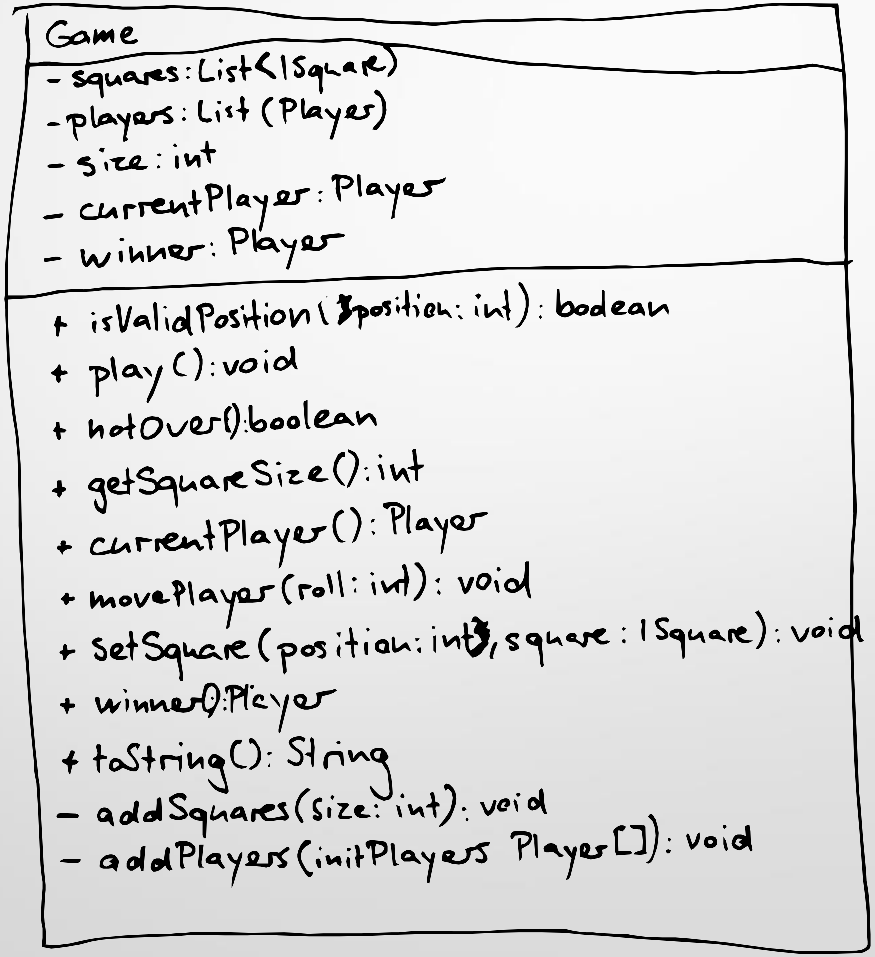
Keep in mind

- Different aspects, different diagram type
- Keep it simple
- Focus on what you want to communicate, forget the rest

On paper: Not enough information



On paper: Too much information

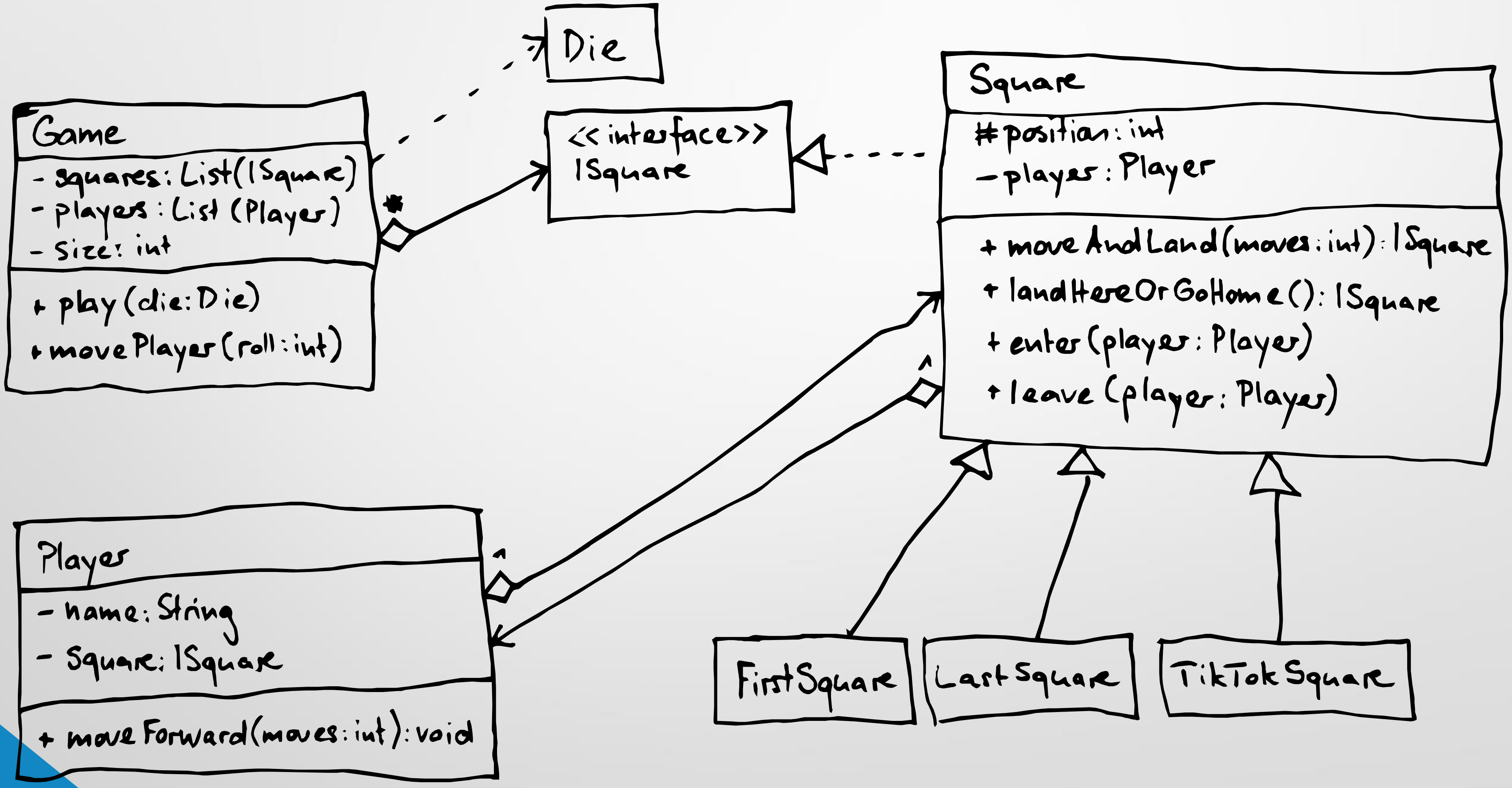


Game

- squares: List<ISquare)
- players: List (Player)
- size : int
- currentPlayer: Player
- winner: Player

+ isValidPosition(position: int): bodean
+ play(): void
+ notOver(): boolean
+ getSquareSize(): int
+ currentPlayer(): Player
+ movePlayer(roll: int): void
+ setSquare(position: int, square: ISquare): void
+ winner(): Player
+ toString(): String
- addSquares(size: int): void
- addPlayers(initPlayers Player[]): void

On paper


Die
Game
- squares: List(ISquare)
- players : List (Player)
- size: int
+ play (die: Die)
+ movePlayer (roll : int)
*
<< interface>>
ISquare
Square
# position: int
- players : Player
+ moveAndLand(moves: int): ISquare
+ landHereOrGoHome(): ISquare
+ enter (player: Player)
+ leave (player: Player)
Player
- name: String
- Square: ISquare
+ moveForward(moves: int): void
FirstSquare
LastSquare
TikTokSquare

Exercise 3

Use the information from the lecture and from this presentation to solve the UML related tasks in Exercise 3

Add both diagrams in a common format (e.g. JPG, PDF) to the exercise root in your group folder.

If you do not have a scanner, you can just take a photo of the UML diagrams with a smartphone.

To learn more

- http://scg.unibe.ch/teaching/p2/ (P2 reading material, UML Reference)
- Book: UML Distilled, Martin Fowler