P2: Design By Contract

Manuel Schuepbach
6 March 2020

Contents

- Feedback Exercise 2
    - JavaDoc
    - Git
- Design by Contract
    - Assertions
    - Exceptions
- UML
- Exercise 3

JavaDoc: Examples

```
/**
 * Square that enables the entering player to immediately roll the dice again.
 */
public class RollAgainSquare extends Square implements ISquare {
     // …
}
```

JavaDoc: Examples

```
/**
 * Square that enables the entering player to immediately roll the dice again.
 */
publi
      // …
}
```

Missing details

JavaDoc: Examples

```java
/**
 * The class RollAgainSquare contains methods enabling the
 * entering player to roll the dice again.
 */
public class RollAgainSquare extends Square implements ISquare {
	// …
}
```

JavaDoc: Examples

```
/**
 * The class RollAgainSquare contains methods enabling the
 * ent
 */
publi
      // …
}
```

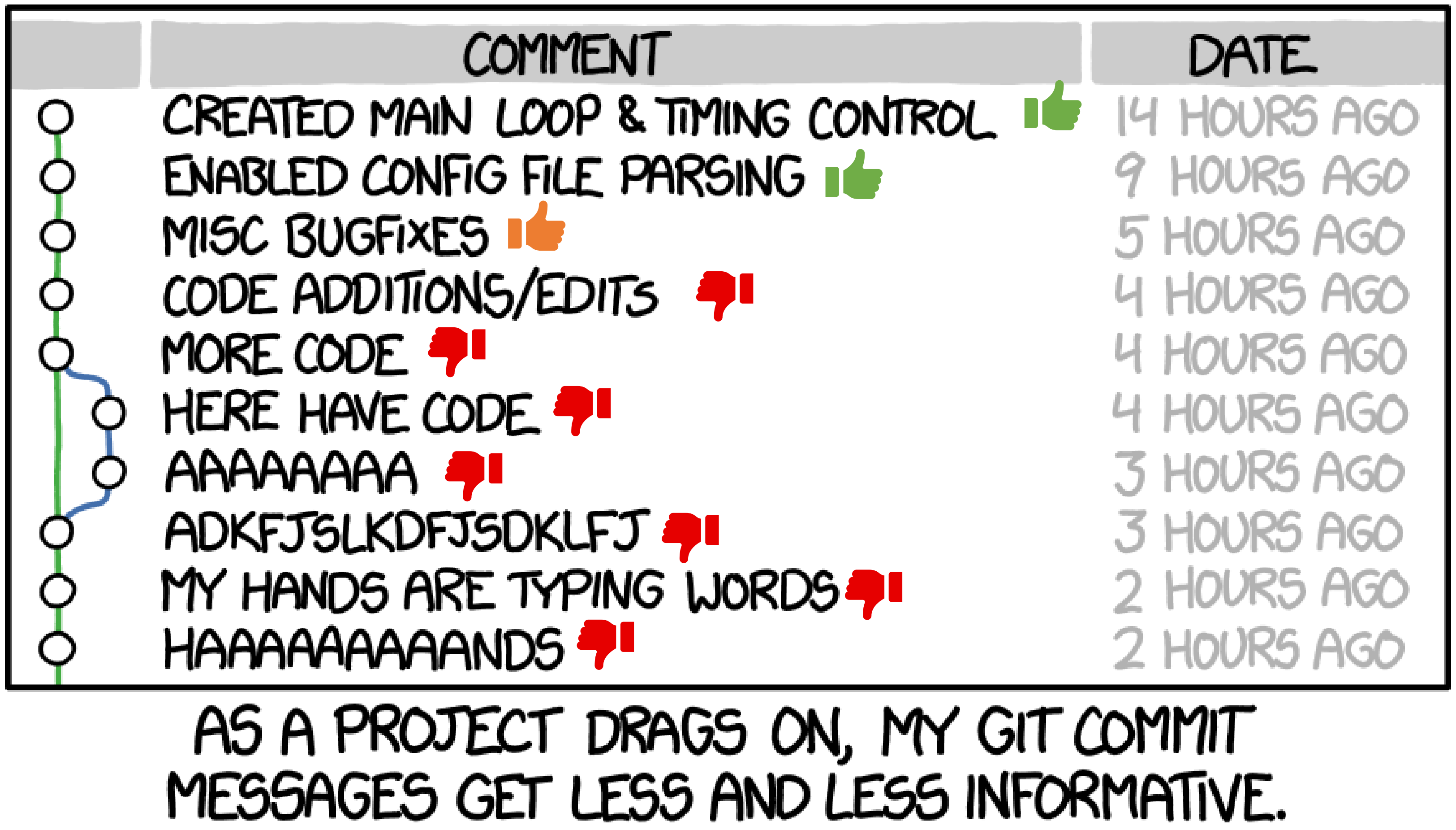
Filler words: The class RollAgainSquare

👎

JavaDoc: Examples

```java
/**
 * Entering player can immediately roll the dice again.
 *
 * Is created and called inside the {@link Game} class.
 * Extends {@link Square}.
 *
 */
public class RollAgainSquare extends Square implements ISquare {
        // …
}
```

Git-messages



https://xkcd.com/1296/

Git-messages

- `No more errors!`
- `I hate git`
- `FIRST TRY`
- `V3`
- `slooowly getting there`
- `Here have some code`
- `changes`

Git-messages

- `Implemented TikTokSquare`
- `Implemented RollAgainSquare enabling the entering player to immediately roll again.`
- `Added Player.toString() method.`

DBC - Example

```
/**
 * Sets the refresh rate for the current display.
 * @param rate new refresh rate
 */
public void setRefreshRate(int rate) {
       // what if rate < 0?
}
```

DBC - Assertion Example

```
/**
 * Sets the refresh rate for the current display.
 * @param rate new refresh rate, must be >= 0
 */
public void setRefreshRate(int rate) {
	assert rate >= 0;
}
```

DBC – Exception Example

```java
/**
 * Sets the refresh rate for the current display.
 *
 * @param rate new refresh rate
 * @throws IllegalArgumentException if rate is not valid
 */
public void setRefreshRate(int rate) throws IllegalArgumentException {
	if (rate < 0) {
		throw new IllegalArgumentException();
	}
}
```

DBC – When to use Assertions

- Use when you expect a property to hold
- Calls inside the program
- Use for contracts
  - Pre-/postconditions, invariants
  - Simplifies design
- Use inside complex code
  - For example to make sure an intermediate result holds

Assertions – Pre-, and Postconditions

```
/**
 * Draw a vertical line, starting from position,
 * with a length of steps + 1.
 *
 * @param position start location of the line, must not be null
 * @param steps length of the line
 */
public void drawVertical(Point position, int steps) {
       assert position != null;     // This is a precondition
       // Implementation here
       assert(invariant());         //This is a postcondition
}
```

DBC – When to use Exceptions

- Favor exceptions for checking method parameters in public/external API
  - Can’t trust user to read JavaDoc
- Always use exceptions to check user input!

Exceptions

- Error handling
- Expected behavior
  - Deal with it in try-catch blocks, or
  - throw it up to the caller

DBC – Checked Exceptions

- Declared Exception

```
public void matches(String filename) throws NotImplementedException {}
```

- Wrapped inside a try-catch block

```
public void fooBar() {
        try {
                // something that throws a TodoException
        } catch (TodoException e) {
                // handle exception
        }
}
```

- Always use checked exceptions unless there is a **very good** reason not to!

NullPointerException

- Very common unchecked exception
- Often hard to tell where it originated
  - Value may be passed around for a while before it is used
- Include **`null`** checks where appropriate

NullPointerException

```java
private void newGame() {
        setPlayer(null);
        execute();
}

private void setPlayer(Player player) {
        this.player = player;
}

private void execute() {
        this.player.move();
}
```

NullPointerException

```
private void newGame() {
        se
        ex
}

private v
        th
}

private v
        this.player.move();
}
```

```
Exception in thread "main" java.lang.NullPointerException
at exercise_03.SomeClass.execute(SomeClass.java:79)
at exercise_03.SomeClass.newGame(SomeClass.java:65)
at exercise_03.SomeClass.main(SomeClass.java:7)
...
Process finished with exit code 1
```

we do not know why player == null

xceptions

```
private void newGame() {
        setPlayer(null);
        execute();
}
/** @param player must not be null */
private void setPlayer(Player player) {
        assert player != null;
        this.player = player;
}
private void execute() {
        this.player.move();
}
```

xceptions

```
private void newGame() {
        setPlayer(null);
        ex
}
/** @para
private v
        as

}
private void execute() {
        this.player.move();
}
```

```
Exception in thread "main" java.lang.AssertionError
at exercise_03.SomeClass.setPlayer(SomeClass.java:74)
at exercise_03.SomeClass.newGame(SomeClass.java:64)
at exercise_03.SomeClass.main(SomeClass.java:7)
Process finished with exit code
```

Stacktrace shows where Nullpointer occured

DBC - Example

```java
/**
 * Look up the object at the top of
 * this stack and return it.
 *
 * @return the object at the top
 */
public E top() {
       return top.item;
}
```

DBC - Example

```
/**
 * Look up the object at the top of
 * this stack and return it.
 * Returns null if called on an empty stack.
 *
 * @return the object at the top
 */
public E top() {
        if (this.isEmpty()) {
                return null;
        }
        return top.item;
}
```

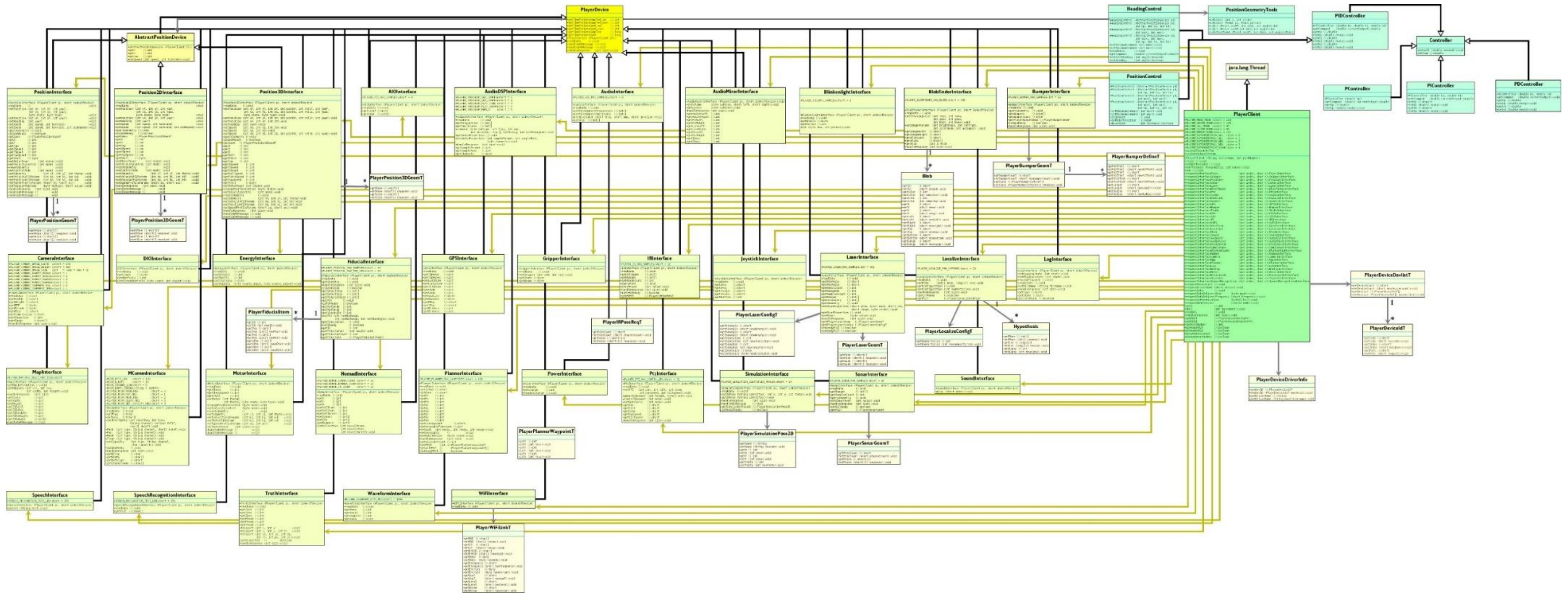
DBC - Example

```java
/**
 * Look up the object at the top of
 * this stack and return it.
 * @throws EmptyStackException if the stack is empty
 *
 * @return the object at the top
 */
public E top() throws EmptyStackException {
        if (this.isEmpty()) {
                throw new EmptyStackException();
        }
        return top.item;
}
```

UML

- Documentation
  - Can be done automatically
    - Can be an overkill (next slide)
- Drafts
  - Simplify reality
  - Understand 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

UML - Documentation



UML - Categories

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

UML - Categories

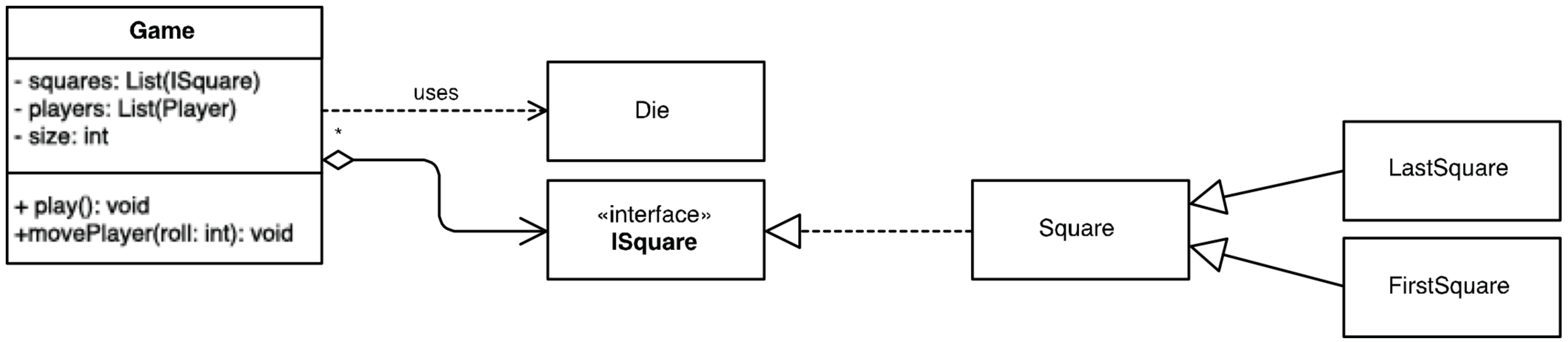
structure

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

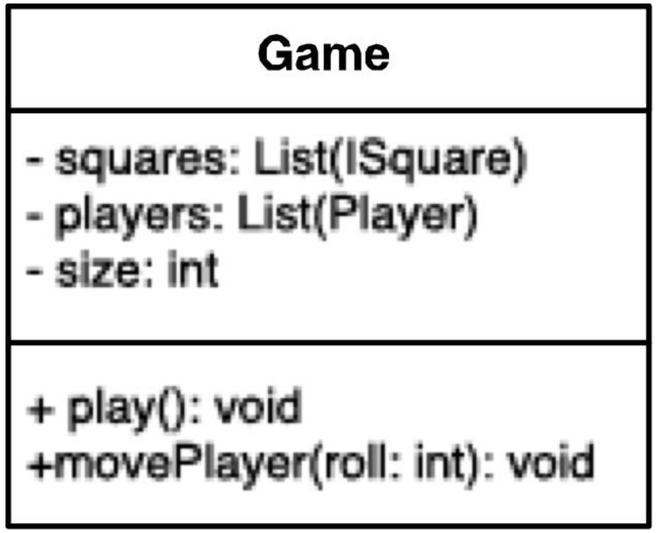
behaviour

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

UML - Example


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

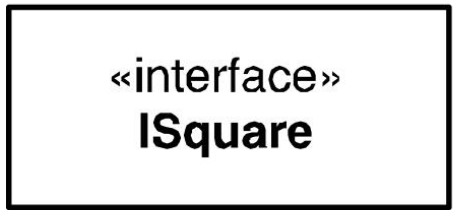
UML



Name

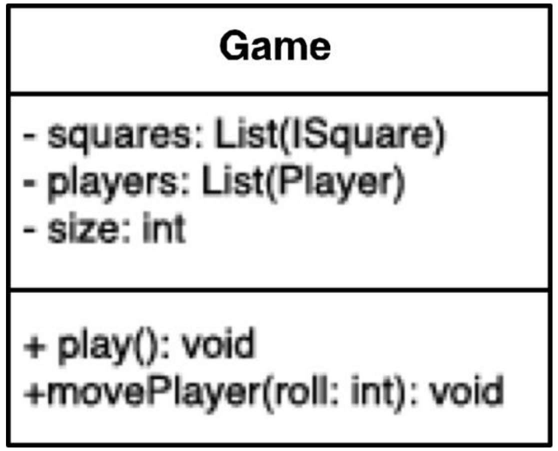
Attributes

Methods



Interface annotation

UML – Class annotation

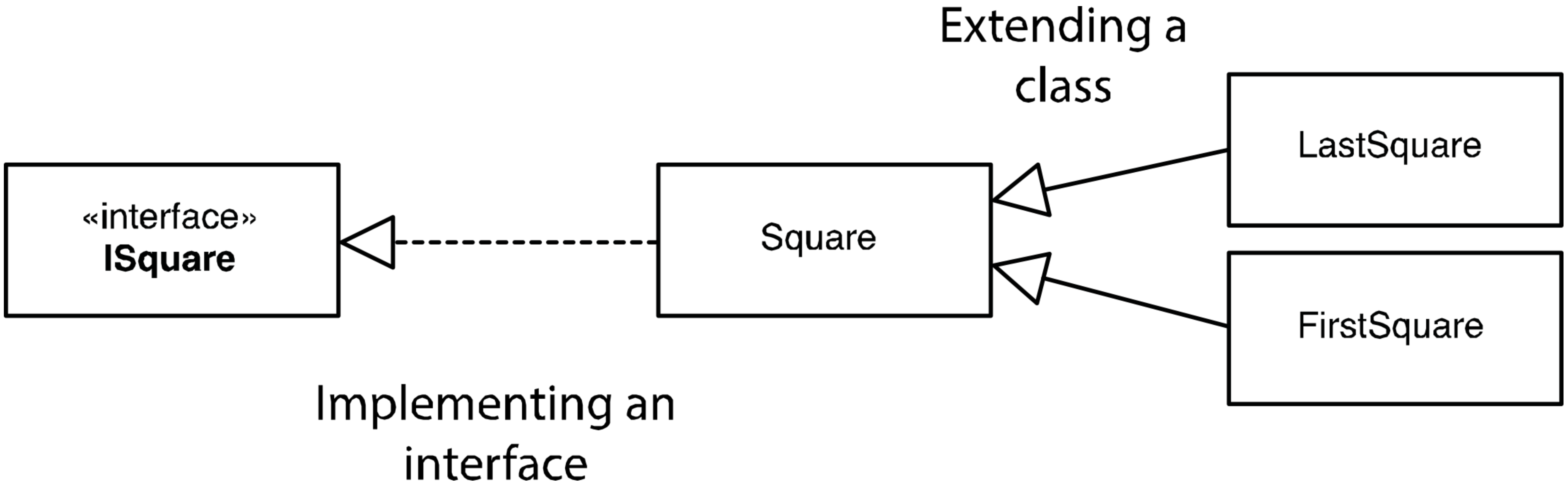


Access modifiers:
+ public, - private, # protected, static

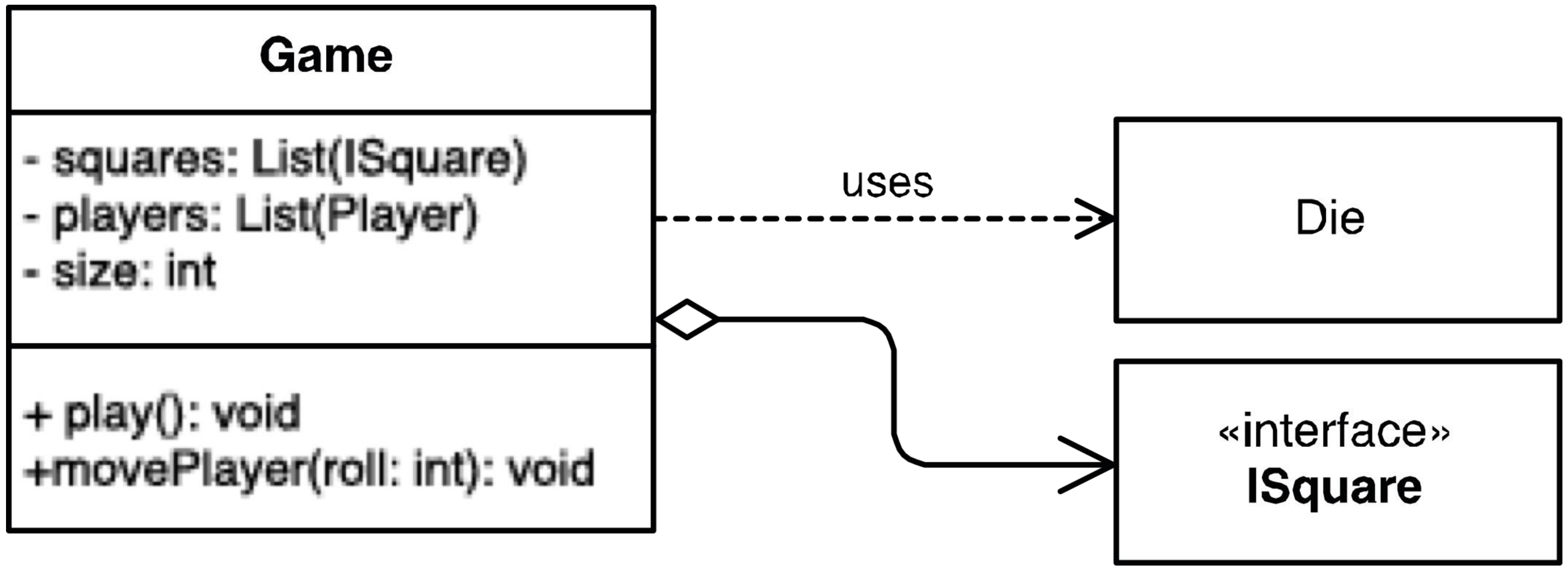
Attributes:
acessIdentifier: type
Example: - size: int

Methods:
accessIdentifier(parameter: type): returnType

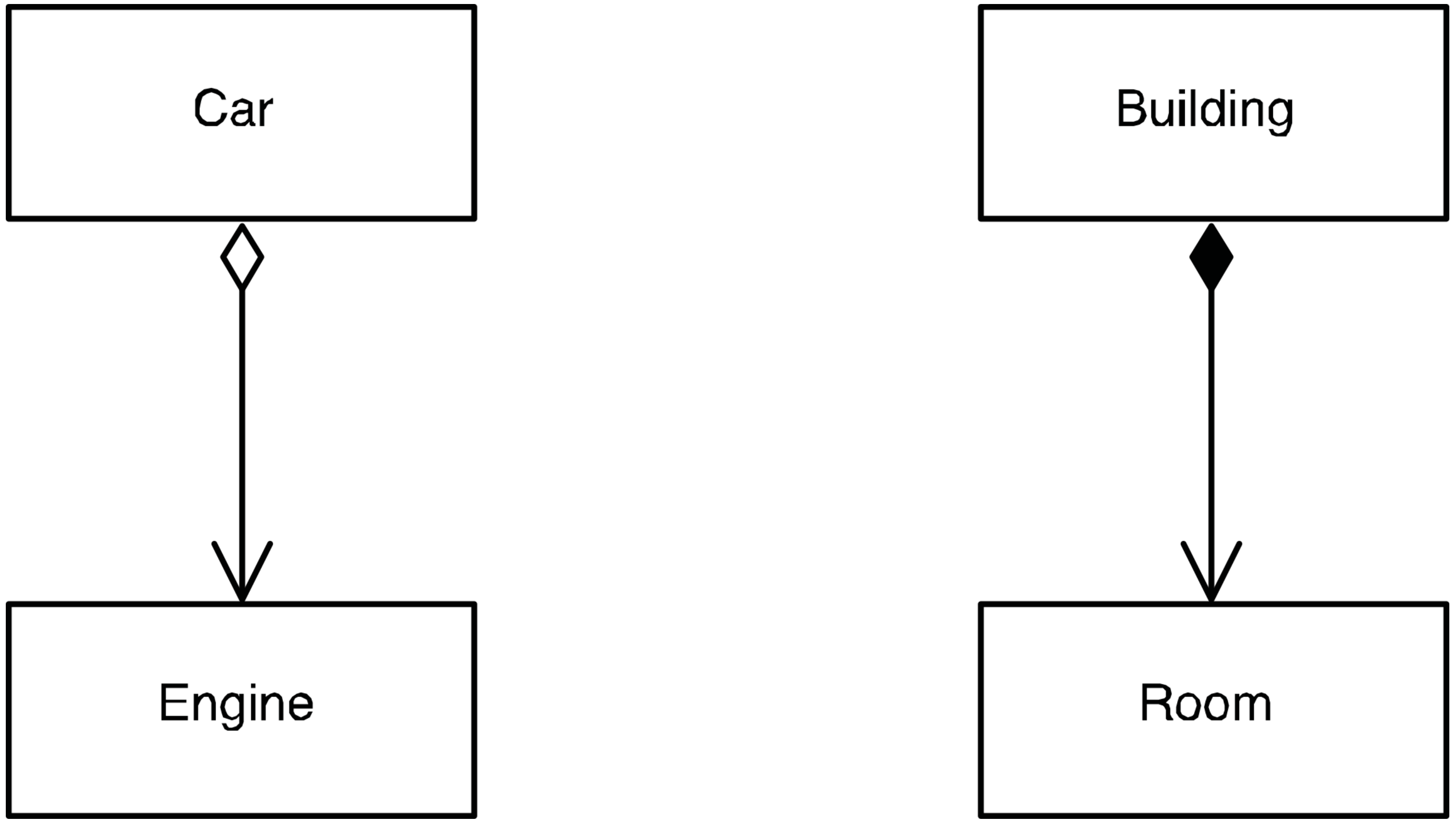
UML - Relationships



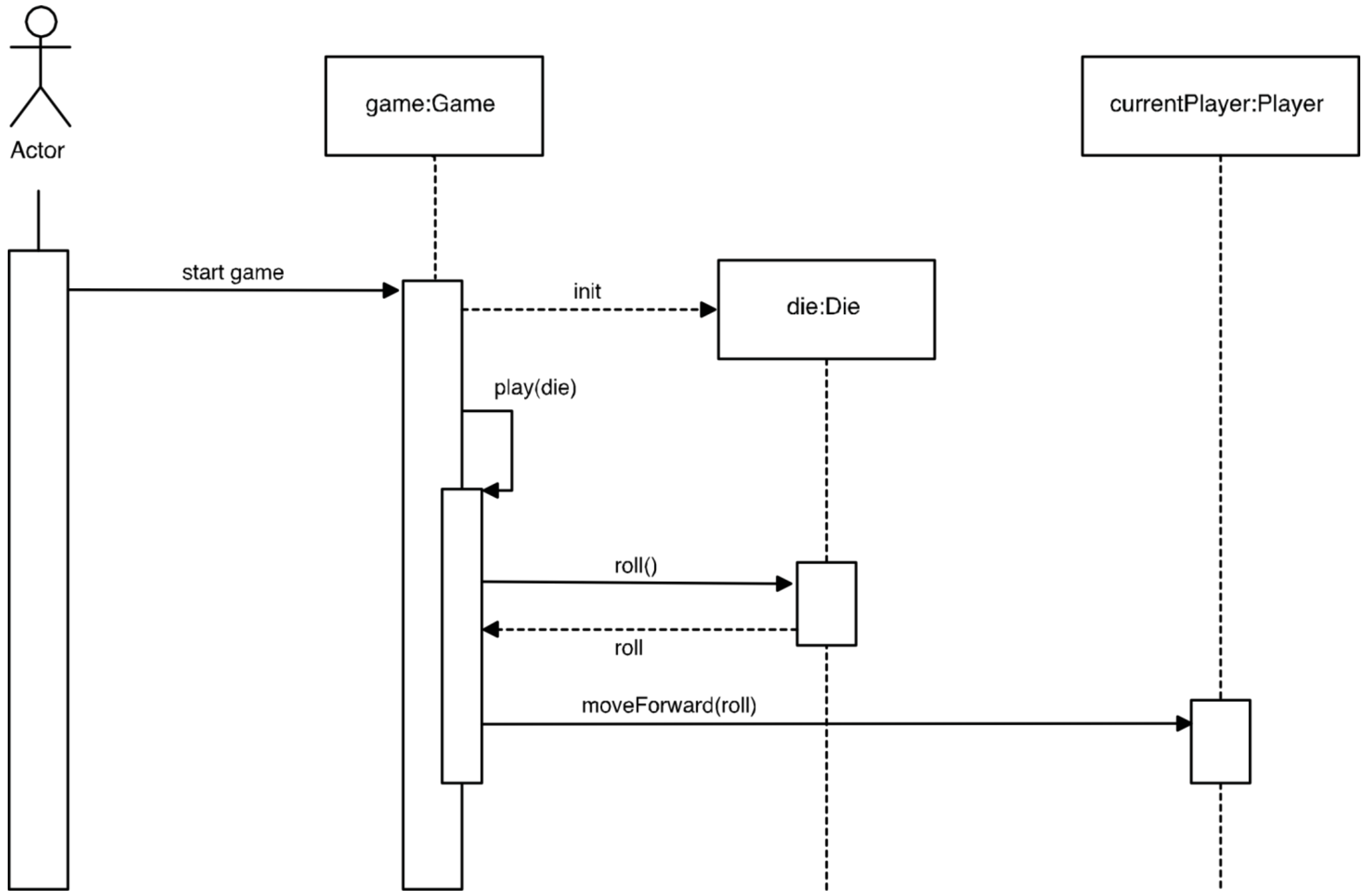
UML - Relationships



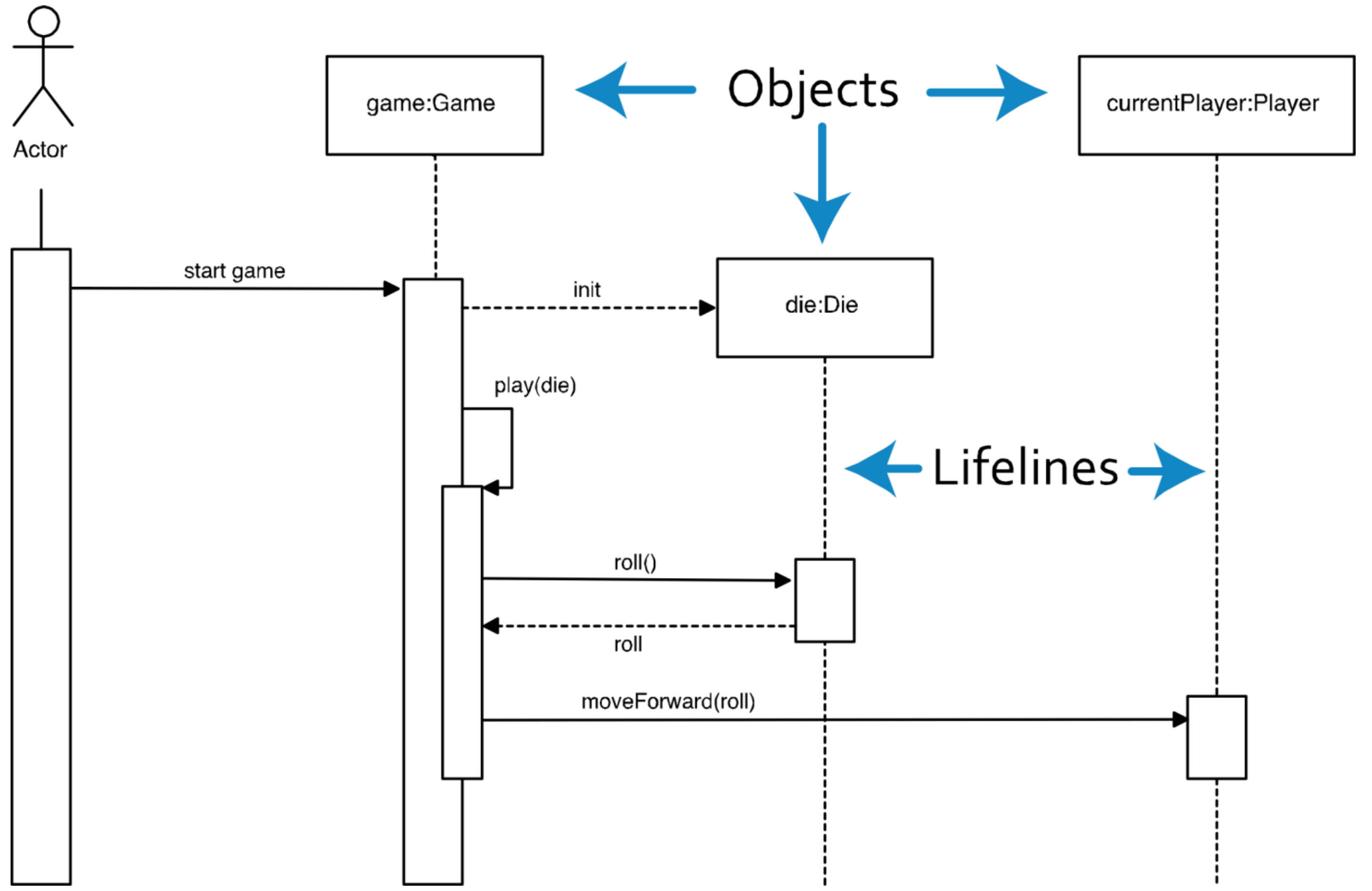
UML – Aggregation vs Composition



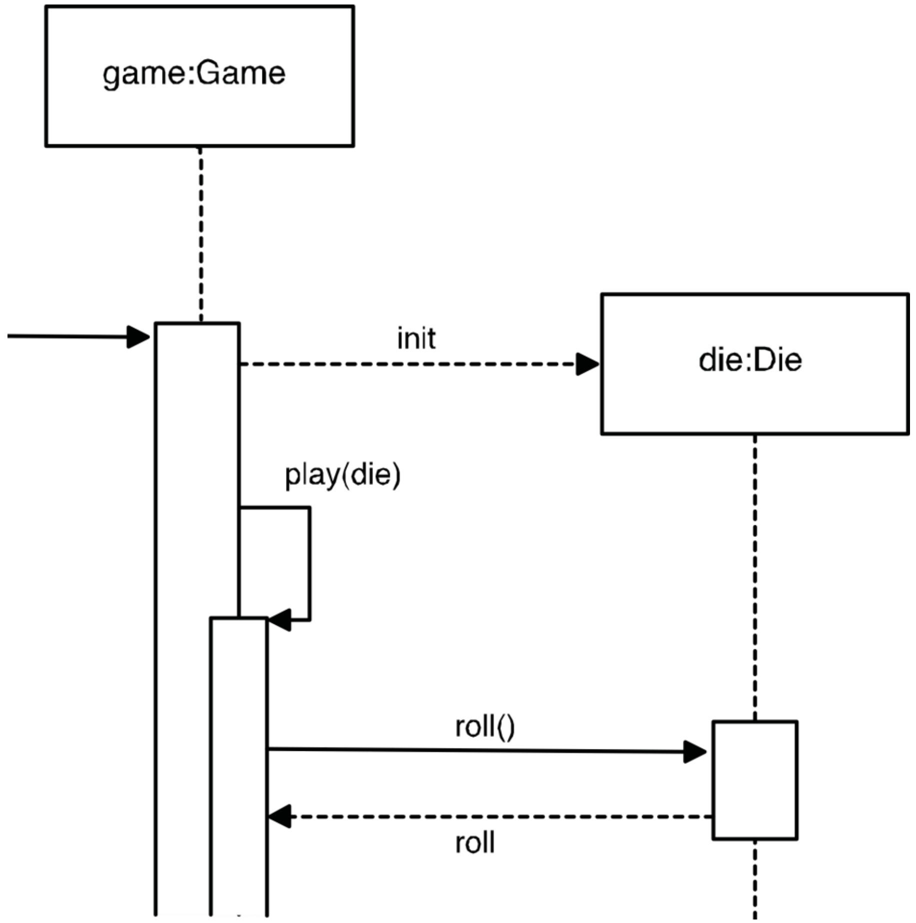
UML – Sequence Diagramm



UML – Sequence Diagramm



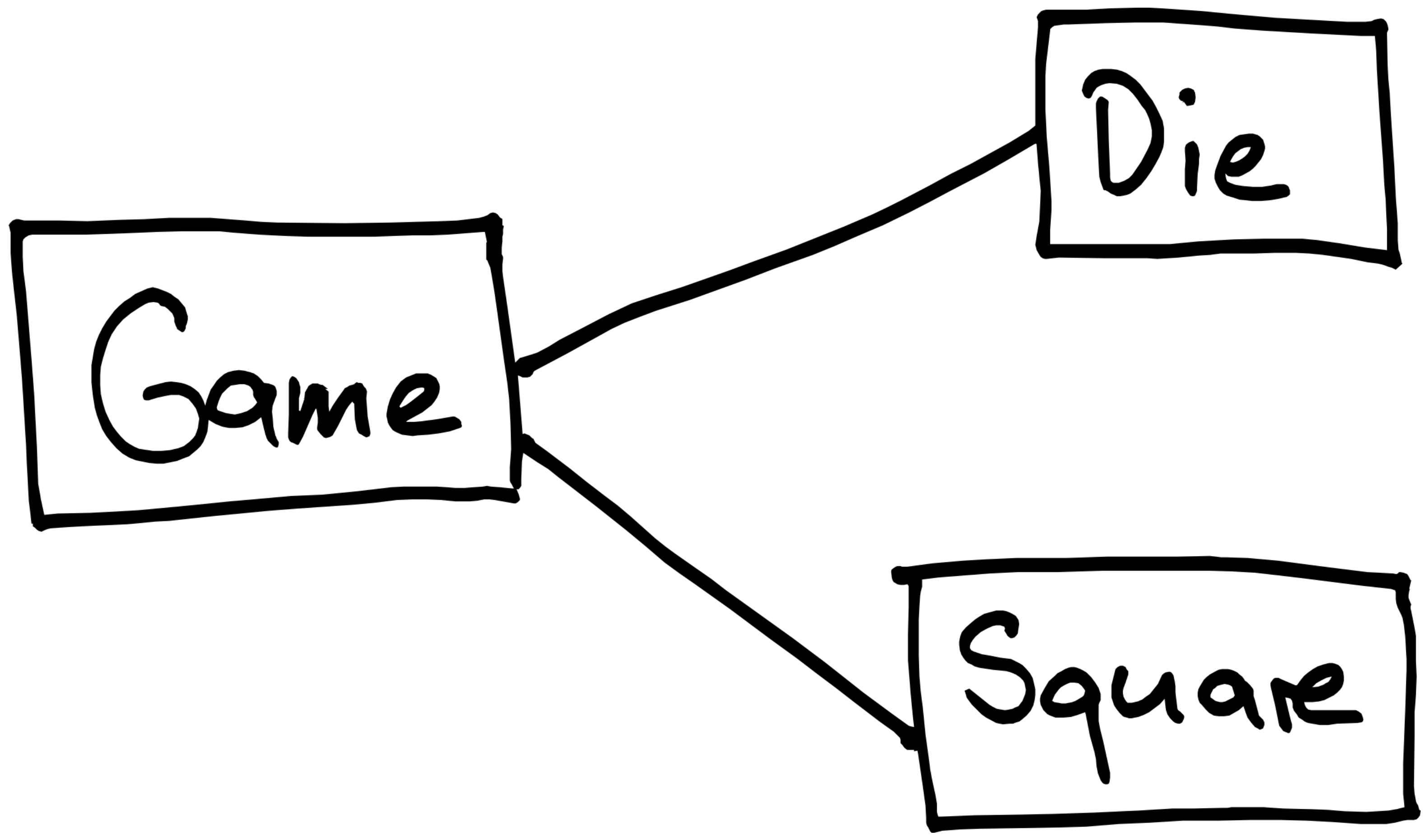
UML – Sequence Diagramm



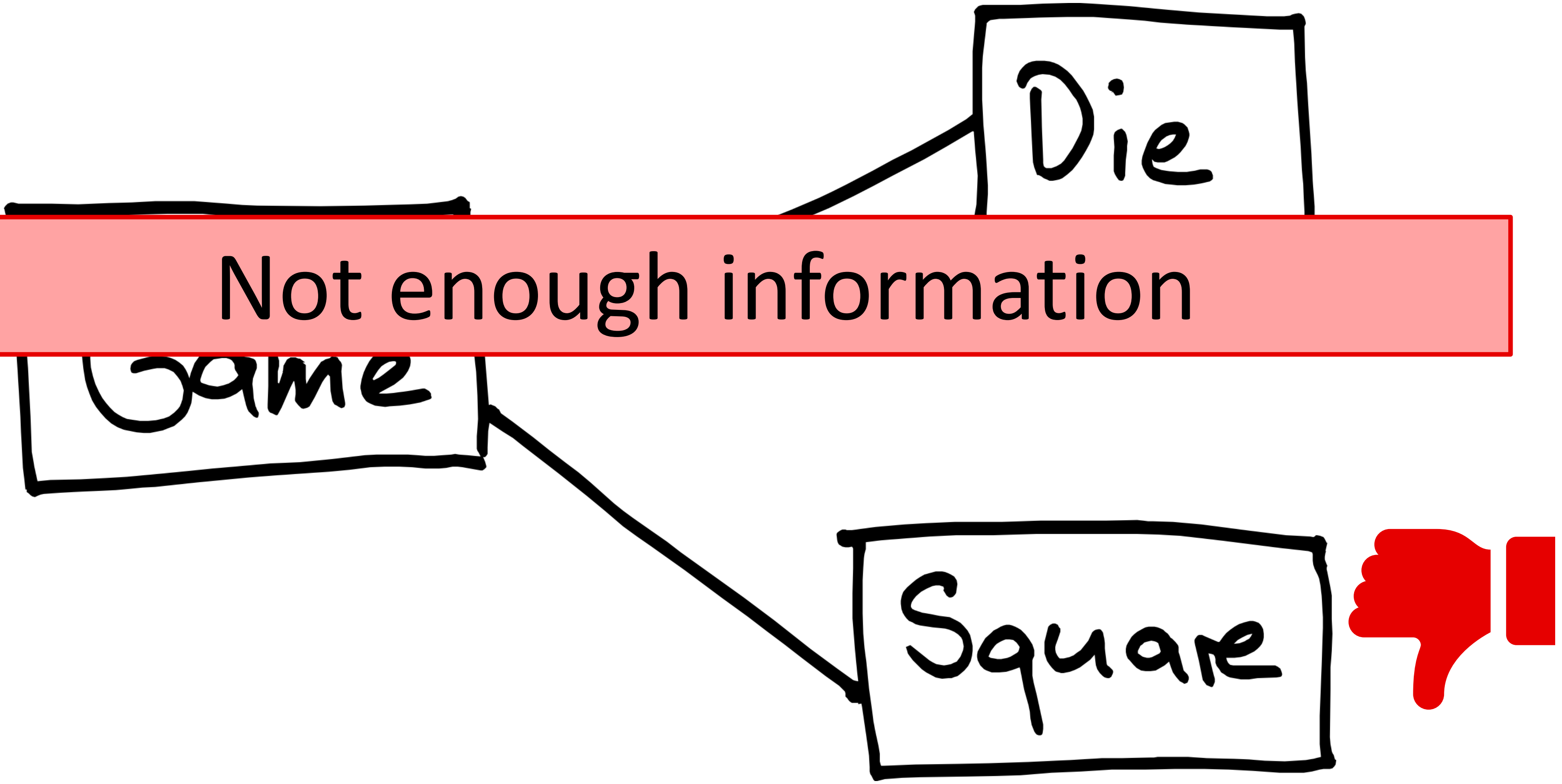
UML - Tips

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

UML - Tips

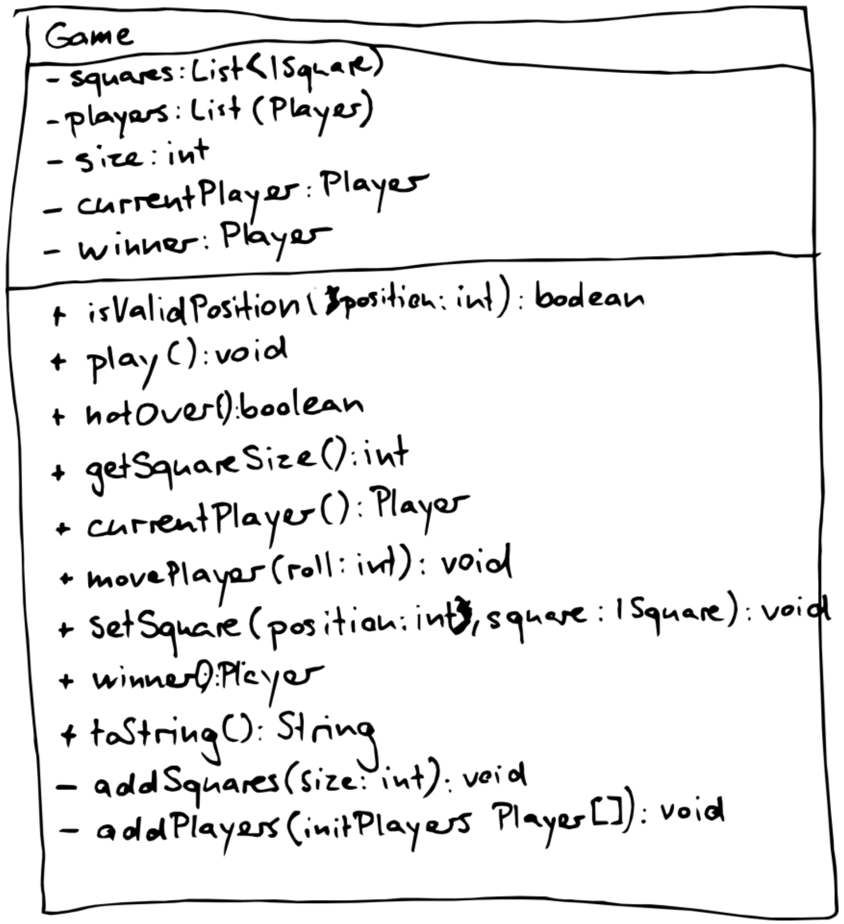


UML - Tips



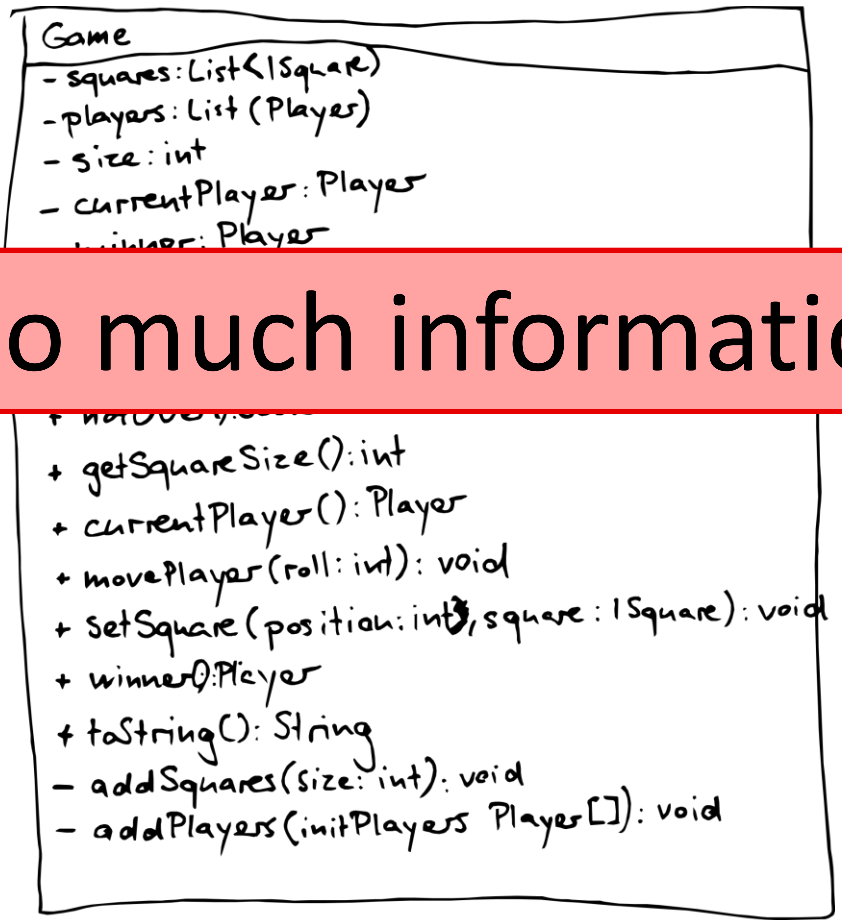
Not enough information

UML - Tips



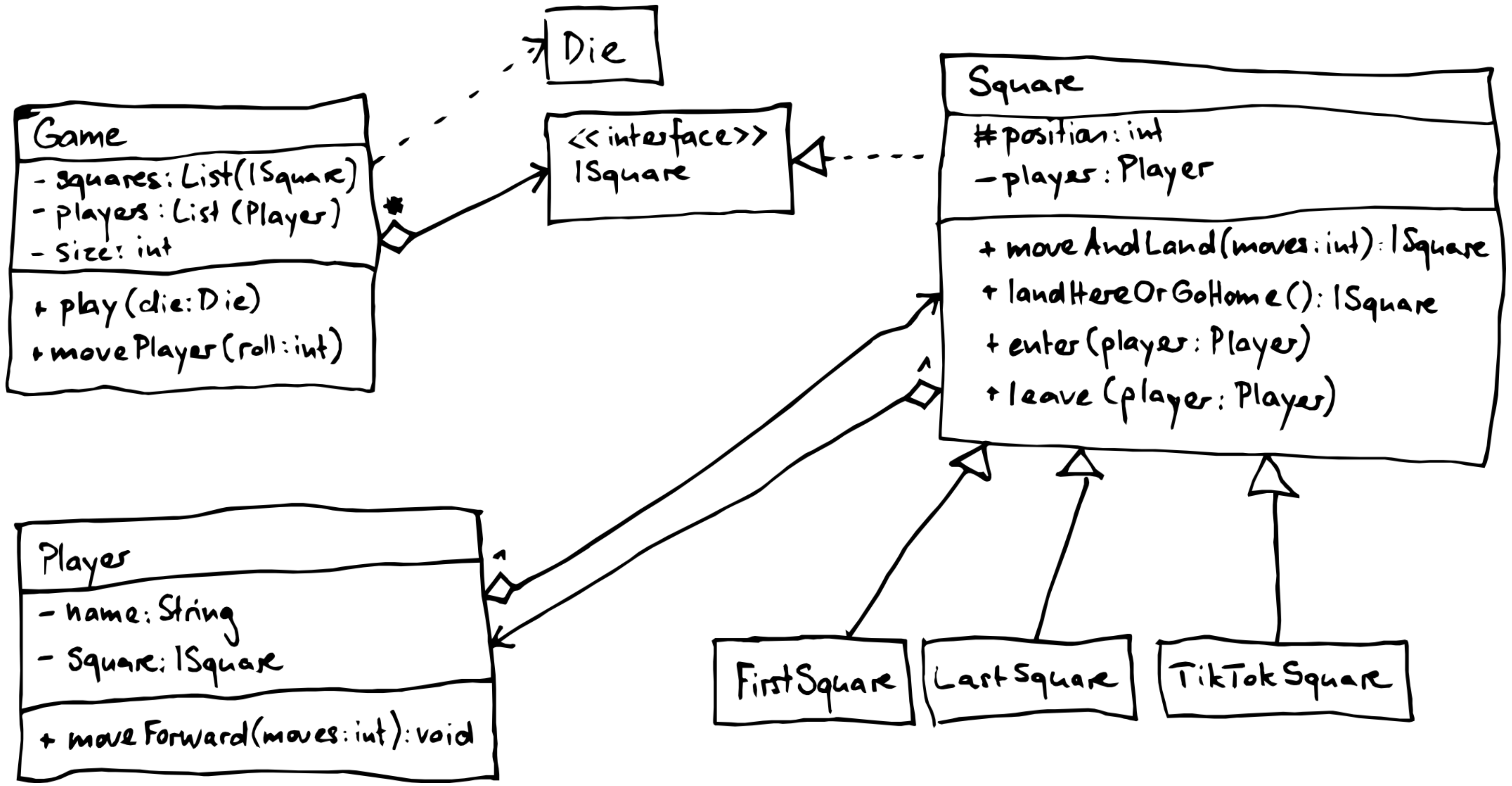
```
Game
- squares: List<ISquare)
- players: List (Player)
- size: int
- currentPlayer: Player
- winner: Player
+ isValidPosition(position: int): boolean
+ play(): void
+ notOver(): boolean
+ getSquareSize(): int
+ currentPlayer(): Player
+ movePlayer(roll: int): void
+ setSquare(position: int, square: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

UML - Tips


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

Too much information

UML - Tips


Die
Game
- squares: List(ISquare)
- players: List (Player)
- size: int
+ play (die: Die)
+ movePlayer (roll: int)
*
<<interface>>
ISquare
Square
# position: int
- player: 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

Additional Material

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

Exercise 3 - Demo

- A turtle that moves around a 100x100 board
  - Commands : `east, west, north, south` or `goto`
  - Leave a red trail
- Input: String representing a turtle program
- Example:

```
east 5
west 4
north 3
goto 20 20
south 10
```

Exercise 3 - Tips

- You start with
  - TurtleRenderer: GUI
  - BoardMaker: Class that gets text from GUI and returns a boolean array of size 50x50
- You have to
  - Parse input program (split lines into commands)
  - Execute turtle actions
  - Keep track of trail
- Use the information from the lecture and form these slides to make the UML diagrams
- Scan the UML or take a picture and add them both to your repository as a .png or .jpg

Exercise 3 - Tips

- You start with
  - TurtleRenderer: GUI
  - BoardMaker: Class that gets text from GUI and returns a boolean array of size 50x50
- You hav
  - Pars
  - Exec
  - Keep
- Use the rams
- Scan the UML or take a picture and add them both to your repository as a .png or .jpg

- **git pull p2-exercises master**
- Read exercise_03.md
- Happy Coding!