A graphical model of an operation could be useful...

What do operations consist of?

A machine performs a certain operation upon a certain product, but it expects the product to be prepared

Does the machine itself has to check whether a product fulfills the conditions?

OrderManager::dispatchOrder(order: Order)

Can we dispatch any order?

An operation is like a service: the conditions of its realization are regulated by a contract

But this contract is usually not expressed explicitly...

How to express a contract?

Contract conditions

Preconditions

Postconditions

Invariants

A commitment:

Given the preconditions,
and if a service is provided,
postconditions will be achieved.

Invariants will be preserved.

Object Constraint Language (OCL)

```
context Game
  inv maxNumberOfPlayers:
    -- 10 players at most
    numberOfPlayers <= 10

  inv minNumberOfPlayers:
    -- 1 player at least
    numberOfPlayers >= 1
```

```
context Game::addPlayer(player : Person) : void
  pre addingAPlayerToGame:
    -- if the game is ongoing, it must be paused
    ongoing implies paused
```

Advanced OCL usage

- Model well-formedness rules (the UML specification)

- The QVT language for defining model transformations in the MDA approach (Model Driven Architecture)

Operations can be expressed by a grahical model on the final software realization level

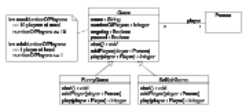
Not even a complete expressing of an operation by a model or code makes its intent and realization conditions directly readable

To express conditions, a formal language is necessary: in UML modeling, OCL is used for this

In overriding, operation preconditions must not be stronger, while postconditions and invariants must not be weaker (we must not ask for more, nor give less)

Can an overriding operation change the realization conditions?

Unless substitution principle



Weakening a precondition

Strengthening an (implicit) postcondition

Lecture 6:

Conditions and Constraints: OCL

Valentino Vranić

Ústav informatiky, informačných systémov a softvérového inžinierstva

STU FIIT

vranic@stuba.sk www2.fiit.stuba.sk/~vranic

MSOFT 2019/20

29. 10. 2019

A graphical model of an operation could be useful...

What do operations consist of?

Operations can be expressed by a grahical model on the final software realization level

A machine performs a certain operation upon a certain product, but it expects the product to be prepared

Does the machine itself has to check whether a product fulfills the conditions?

OrderManager::dispatchOrder(order: Order)

Can we dispatch any order?

An operation is like a service: the condtions of its realization are regulated by a contract

But this contract is usually not expressed explicitly...

How to express a contract?

Contract conditions

How to express a contract?

Contract conditions

- Preconditions
- Postconditions
- Invariants

Preconditions

Postconditions

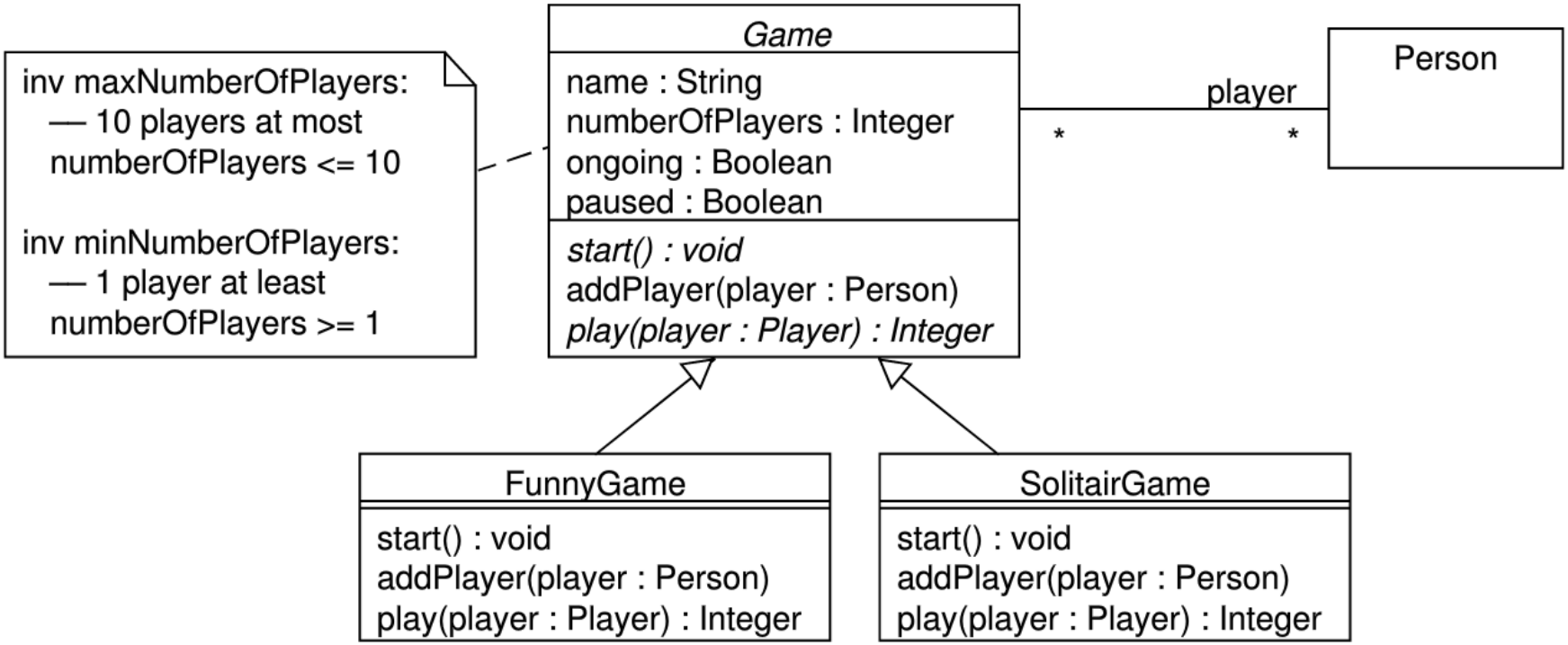
Invariants

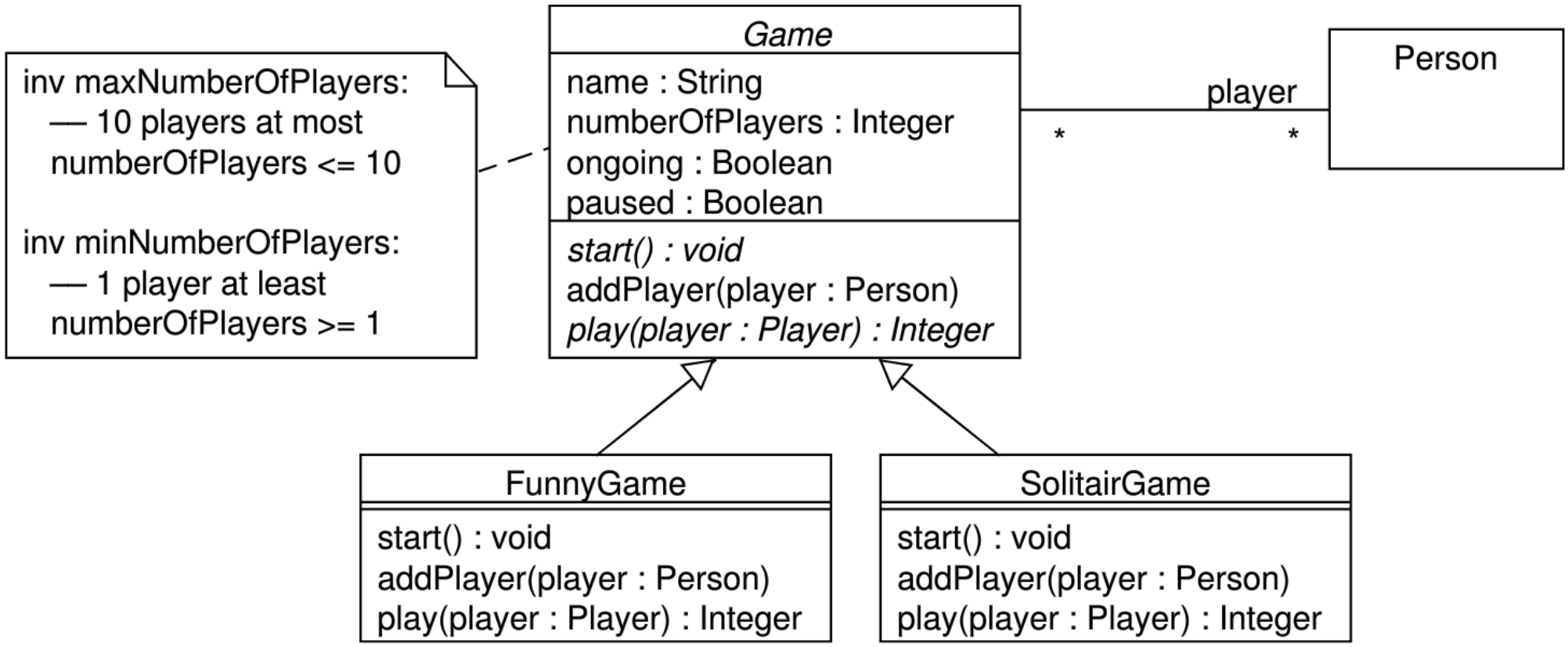
A commitment:

> Given the preconditions,
> and if a service is provided,
> postconditions will be achieved.
>
> Invariants will be preserved.

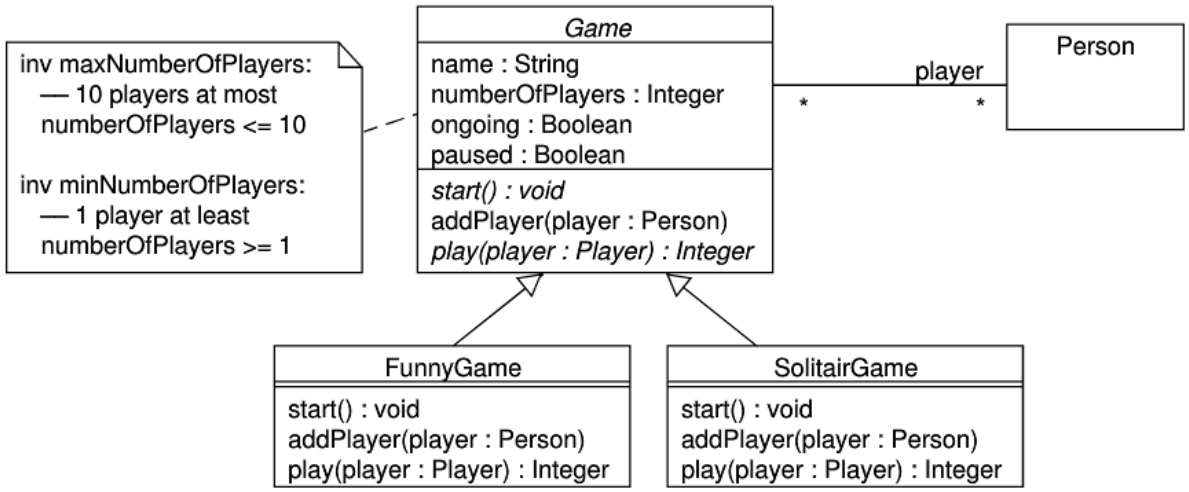
Not even a complete expressing of an operation by a model or code makes its intent and realization conditions directly readable

Object Constraint Language (OCL)

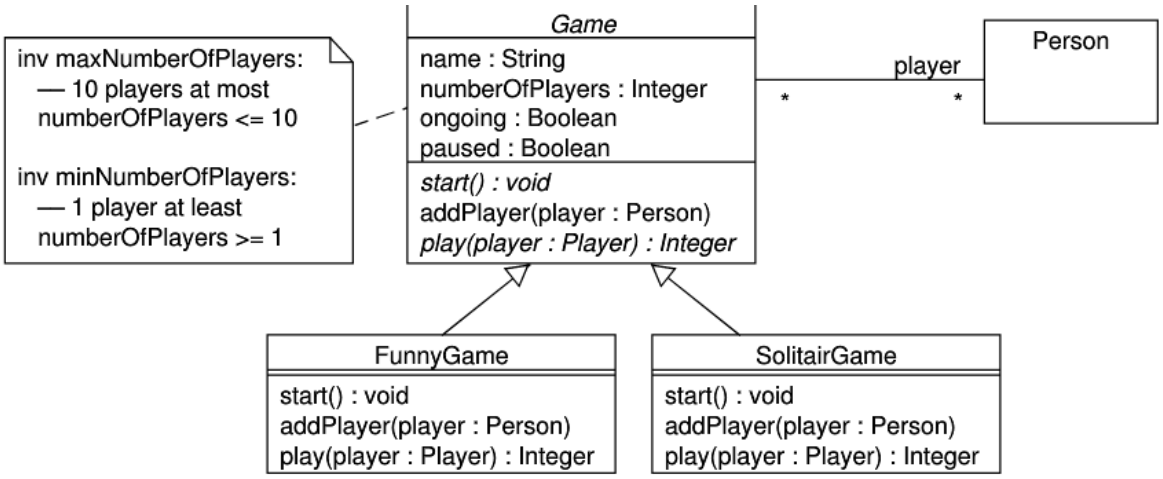



inv maxNumberOfPlayers:
— 10 players at most
numberOfPlayers <= 10
inv minNumberOfPlayers:
— 1 player at least
numberOfPlayers >= 1
Game
name : String
numberOfPlayers : Integer
ongoing : Boolean
paused : Boolean
start() : void
addPlayer(player : Person)
play(player : Player) : Integer
player
*
*
Person
FunnyGame
start() : void
addPlayer(player : Person)
play(player : Player) : Integer
SolitairGame
start() : void
addPlayer(player : Person)
play(player : Player) : Integer

Object Constraint Language (OCL)




```
context Game
  inv maxNumberOfPlayers:
    -- 10 players at most
    numberOfPlayers <= 10

  inv minNumberOfPlayers:
    -- 1 player at least
    numberOfPlayers >= 1
```



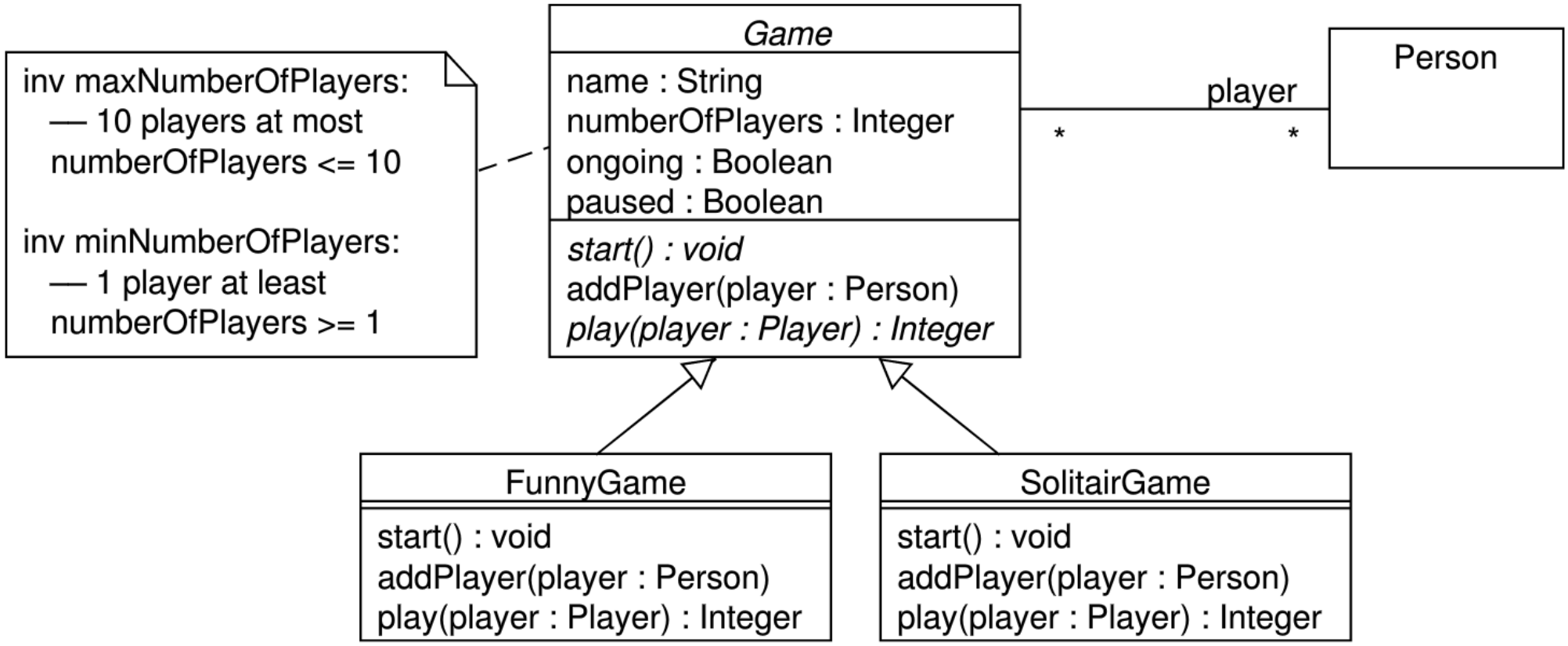

```
context Game
  inv maxNumberOfPlayers:
    -- 10 players at most
    numberOfPlayers <= 10

  inv minNumberOfPlayers:
    -- 1 player at least
    numberOfPlayers >= 1
```

```
context Game::addPlayer(player : Person) : void
  pre addingAPlayerToGame:
    -- if the game is ongoing, it must be paused
    ongoing implies paused
```

To express conditions, a formal language is necessary: in UML modeling, OCL is used for this

Can an overriding operation change the realization conditions?

Liskov substitution principle


inv maxNumberOfPlayers:
— 10 players at most
numberOfPlayers <= 10
inv minNumberOfPlayers:
— 1 player at least
numberOfPlayers >= 1
Game
name : String
numberOfPlayers : Integer
ongoing : Boolean
paused : Boolean
start() : void
addPlayer(player : Person)
play(player : Player) : Integer
player
*
*
Person
FunnyGame
start() : void
addPlayer(player : Person)
play(player : Player) : Integer
SolitairGame
start() : void
addPlayer(player : Person)
play(player : Player) : Integer

```
context Game::addPlayer(player : Person) : void
  pre addingAPlayerToAGame:
    -- if the game is ongoing, it must be paused
    ongoing implies paused
```

```
context FunnyGame::addPlayer(player : Person) : void
  pre addingAPlayerToAGame:
      true
```

Weakening a precodition

```
context Game::addPlayer(player : Person) : void
  pre addingAPlayerToGame:
    -- if the game is ongoing, it must be paused
    ongoing implies paused
```

```
context FunnyGame::addPlayer(player : Person) : void
    pre addingAPlayerToGame:
    -- a game must not be ongoing
    not ongoing
```

Strengthening a precondition

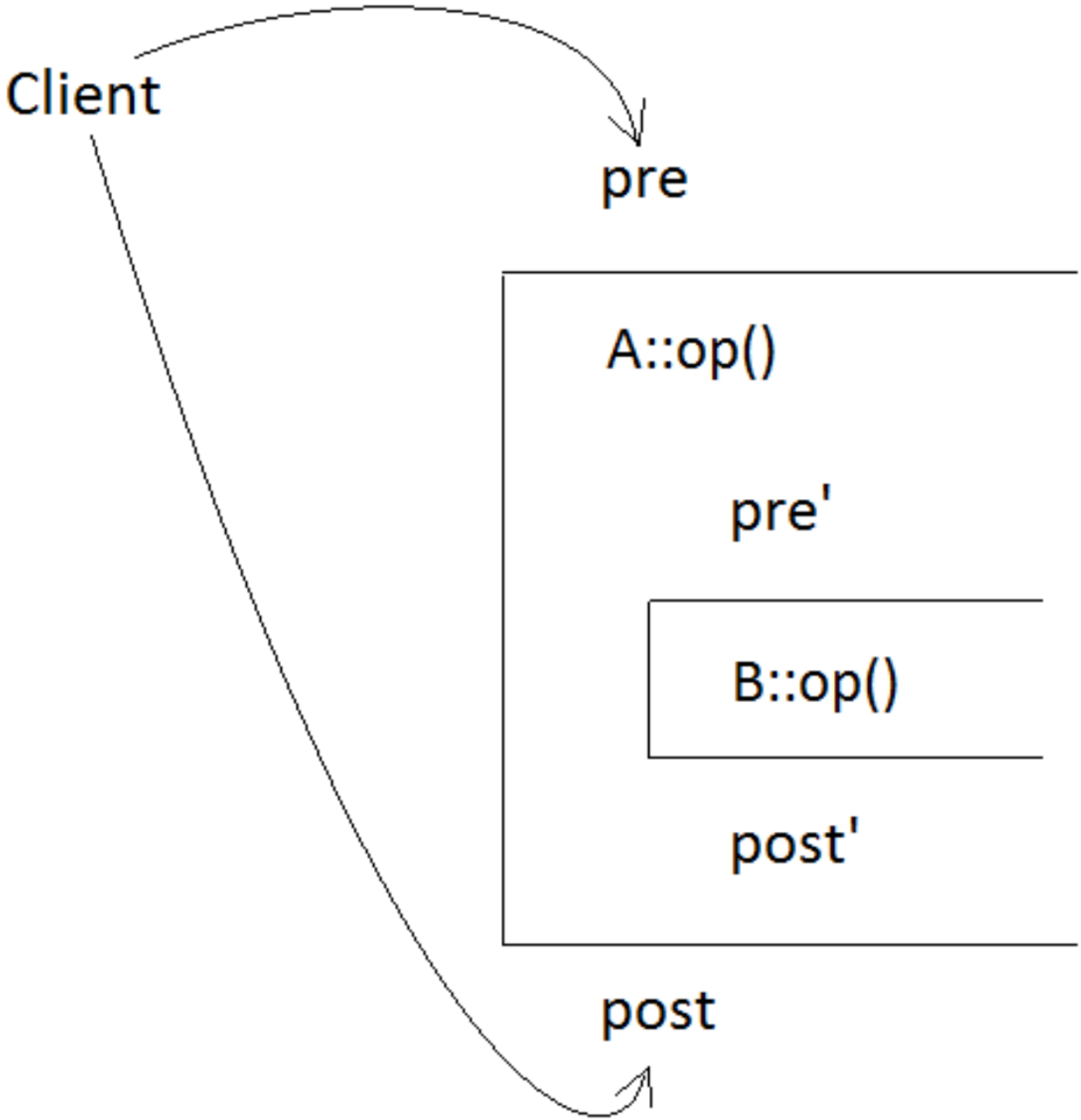
```
for (Game game : allGames)
    if (!game.ongoing || game.paused) // ongoing => paused
        game.addPlayer(player)
```

A problem for a client!

```
context FunnyGame::addPlayer(player : Person) : void
    post addingAPlayerRaisesTheirNumber:
    -- the recorded number of players will be increased by 1
    numberOfPlayers = numberOfPlayers@pre + 1
```

Stregthening an (implicit) postcondition

```
numberOfPlayers >= numberOfPlayers@pre
```


Client
pre
A::op()
pre'
B::op()
post'
post

In overriding, operation preconditions must not be stronger, while postconditions and invariants must not be weaker (we must not ask for more, nor give less)

Advanced OCL usage

- Model well-formedness rules (the UML specification)

- The QVT language for defining model transformations in the MDA approach (Model Driven Architecture)

Operations can be expressed by a grahical model on the final software realization level

Not even a complete expressing of an operation by a model or code makes its intent and realization conditions directly readable

To express conditions, a formal language is necessary: in UML modeling, OCL is used for this

In overriding, operation preconditions must not be stronger, while postconditions and invariants must not be weaker (we must not ask for more, nor give less)