Object-oriented design heuristics

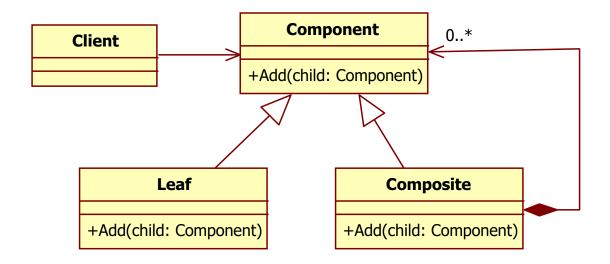
Object-oriented software design

Dr. Balázs Simon BME, IIT



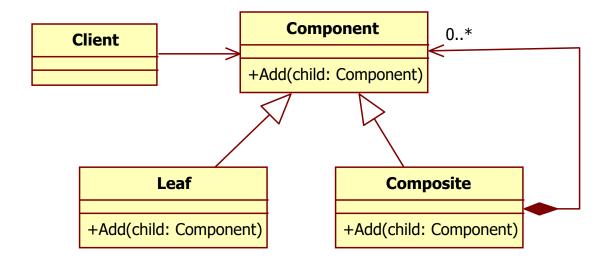
Which of the following design principles does the Composite pattern below violate?

LSP and/or OCP



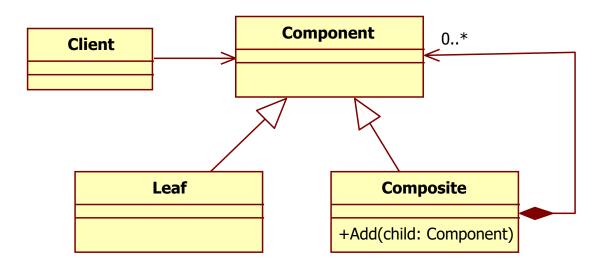
Which of the following design principles does the Composite pattern below violate?

Solution: LSP is violated, OCP is satisfied



Which of the following design principles does the Composite pattern below violate?

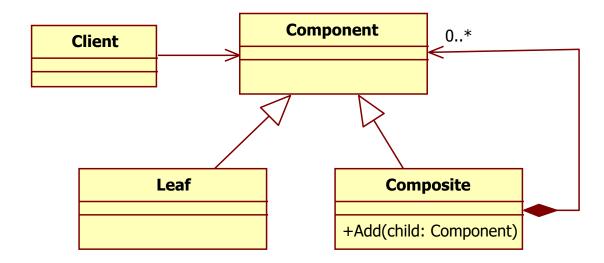
LSP and/or OCP





Which of the following design principles does the Composite pattern below violate?

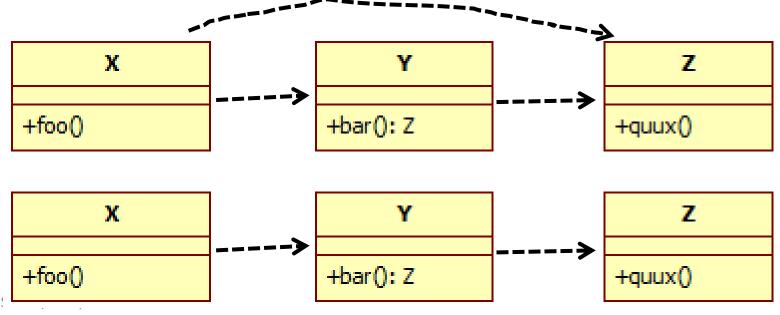
Solution: LSP is satisfied, OCP is violated



LoD

LoD

Which diagram may denote the violation of the Law of Demeter?



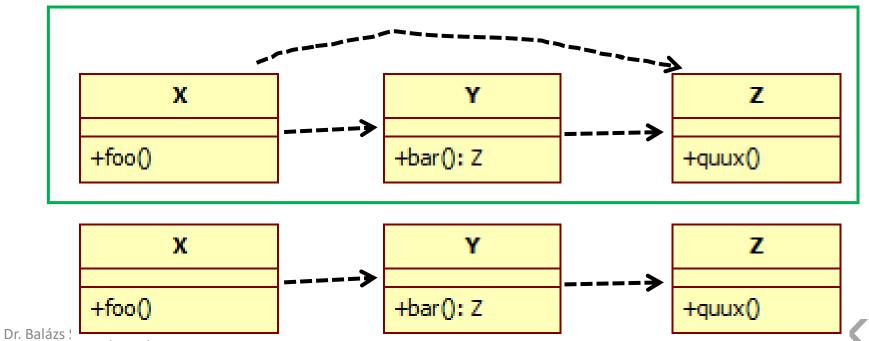
Dr. Balázs !

8

LoD

Which diagram may denote the violation of the Law of Demeter?

Solution:



9

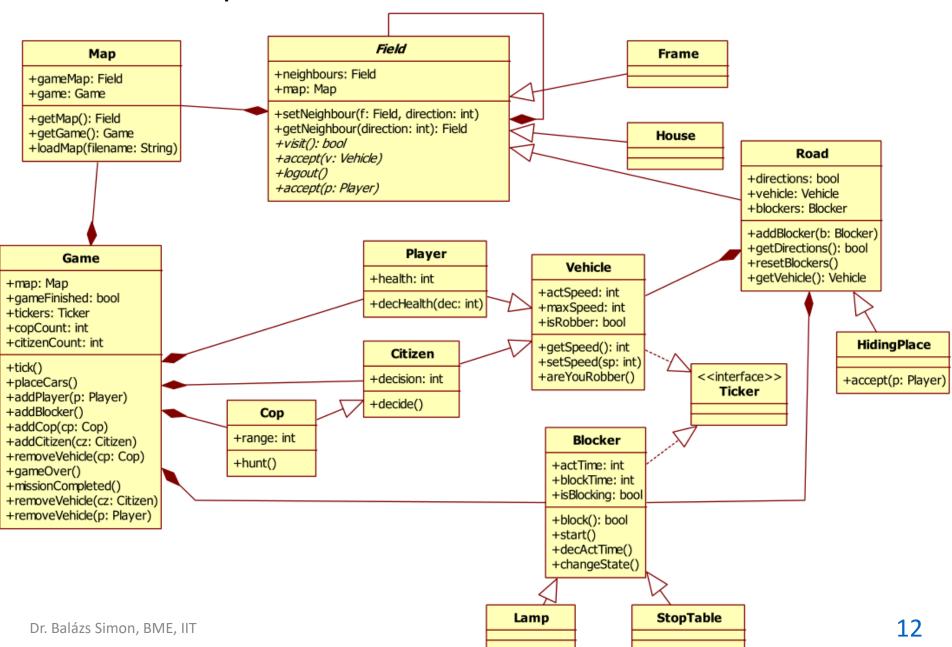
Heuristics

Design task

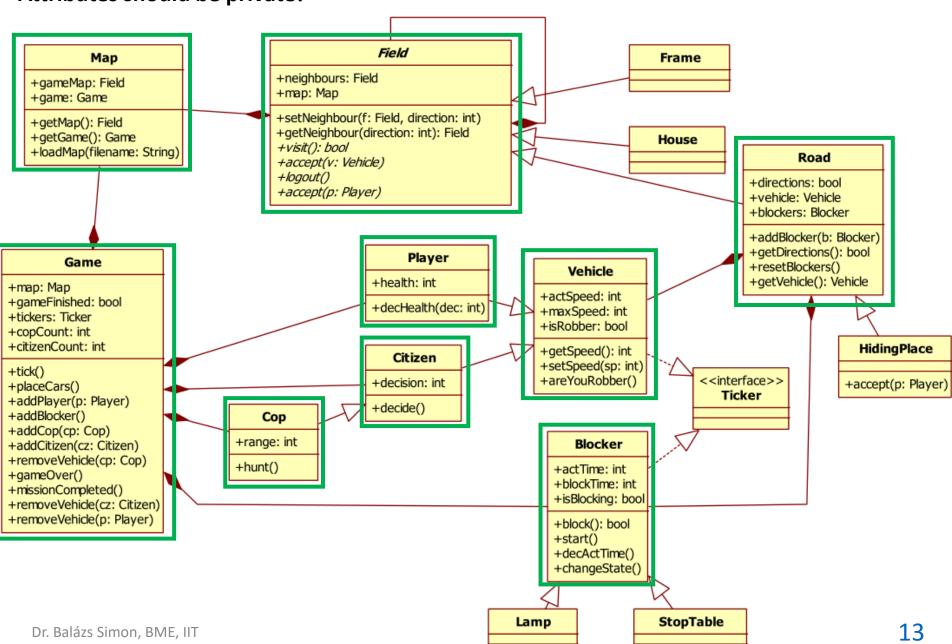
- In a small town there has been a bank robbery. The robbers try to flee using a car, however, there are other cars on the road and also the police is after them.
- The roads in the town are all one-way roads, and all the rules are signaled by signs (there are no right-handed junctions). All cars follow the roads and keep the rules. They stop at red lights. They also stop at STOP signs and give way to other cars if they are near enough. The cars go in a random direction when they arrive at a junction. Cars do not bump into each other, and they do not take over each other. If a faster car catches up with a slower car, it also slows down, but when it can go freely, it speeds up again. The cars arrive from the edges of the city, and they vanish when they leave the city.
- The robbers can break the rules (go through red light and stop signs). But they have to be careful not to hit another car and not to be caught by the police, before they reach their hiding place.

Dr. Balázs Simon, BME, IIT

"Attributes should be private!"

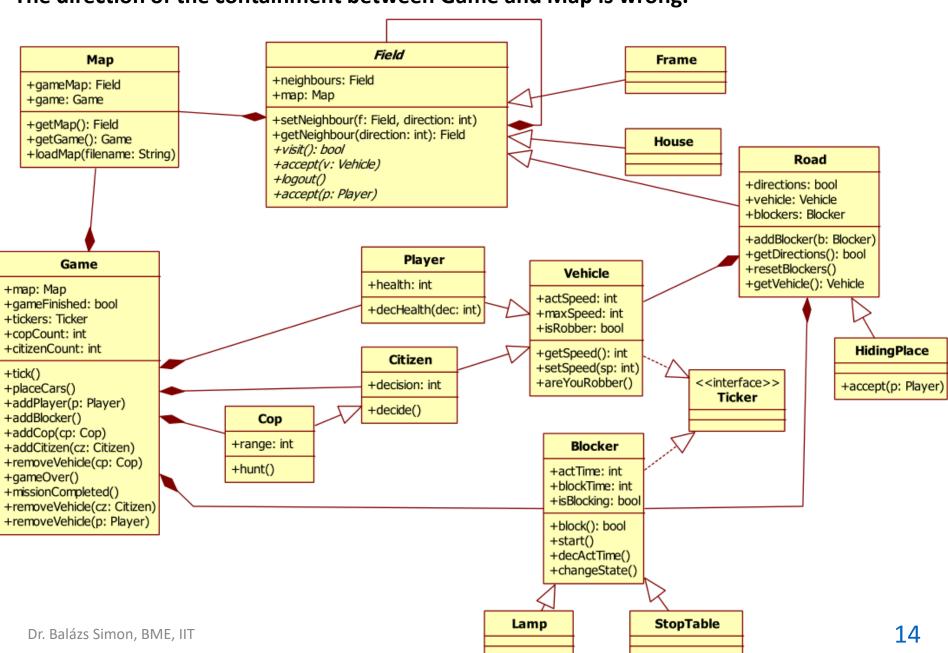


"Attributes should be private!"



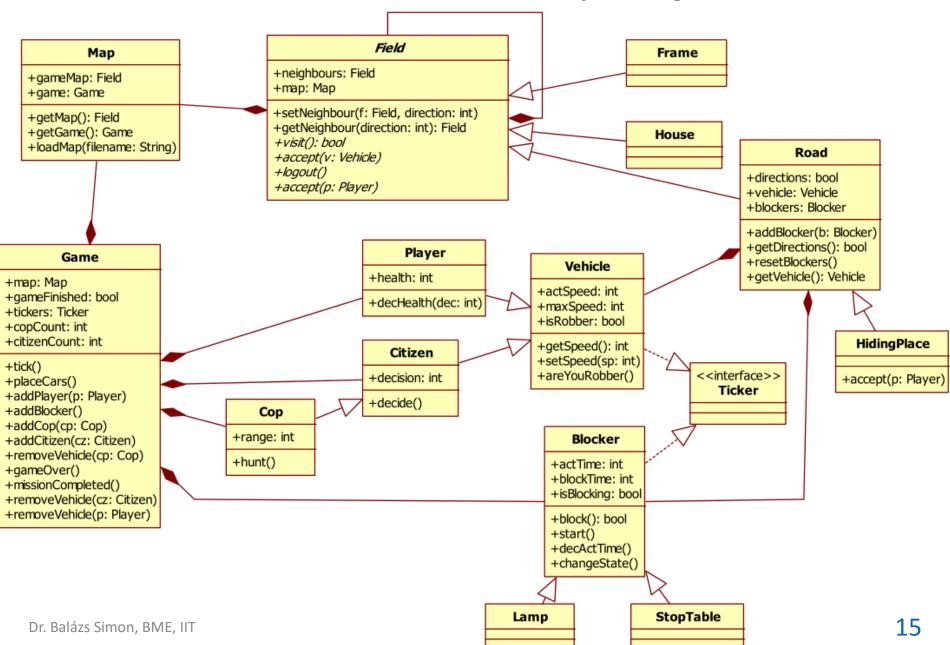
Is the following statement true or false?

"The direction of the containment between Game and Map is wrong."



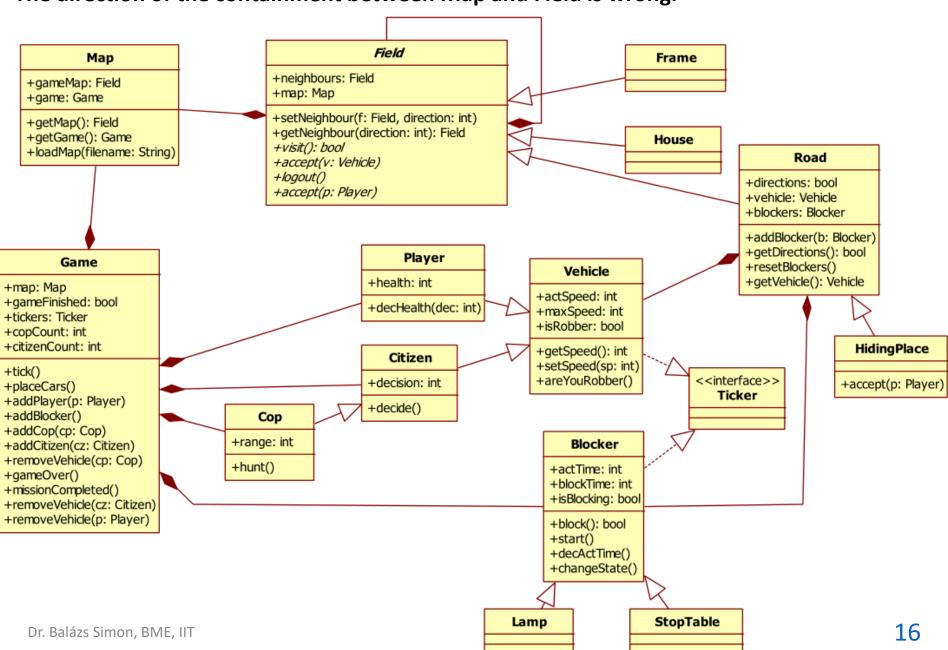
Solution: False

"The direction of the containment between Game and Map is wrong."



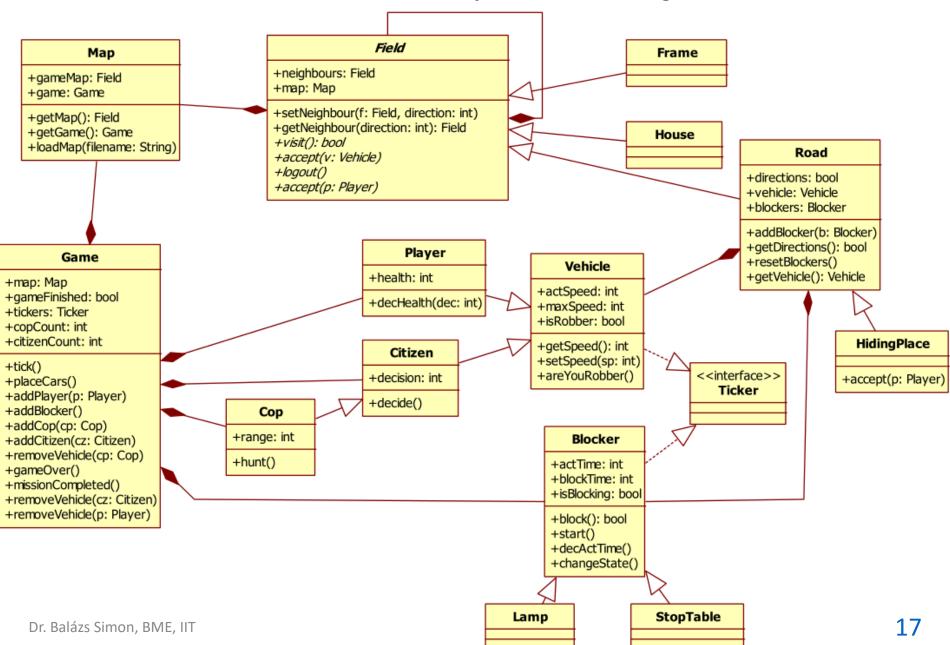
Is the following statement true or false?

"The direction of the containment between Map and Field is wrong."

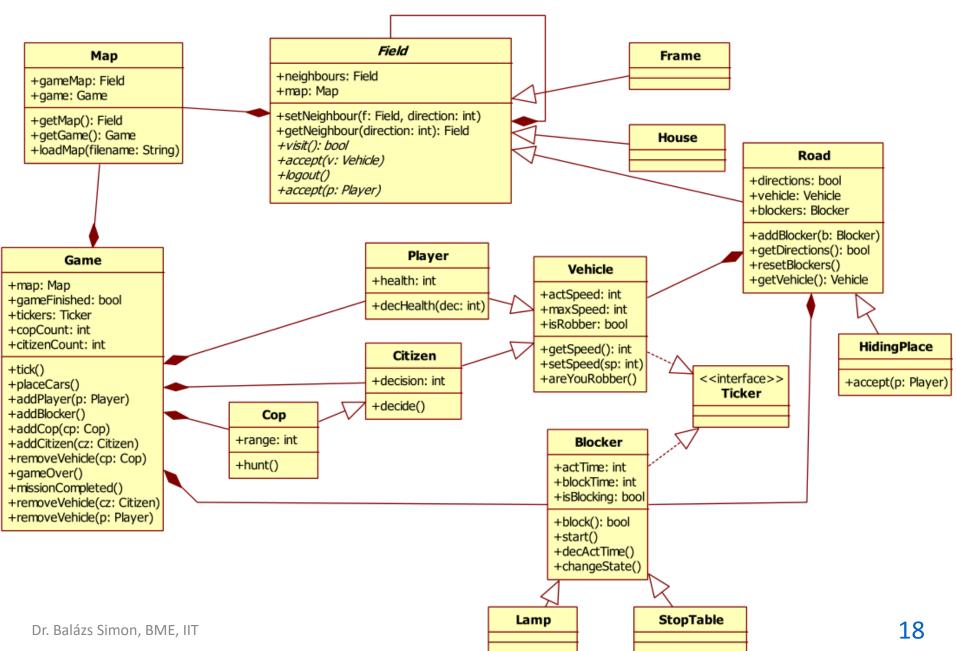


Solution: True

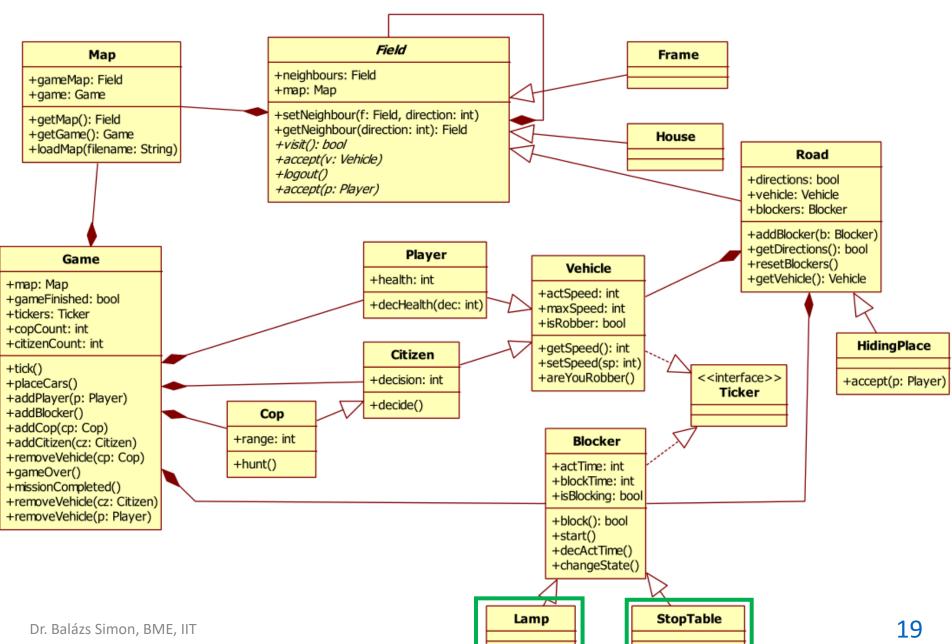
"The direction of the containment between Map and Field is wrong."



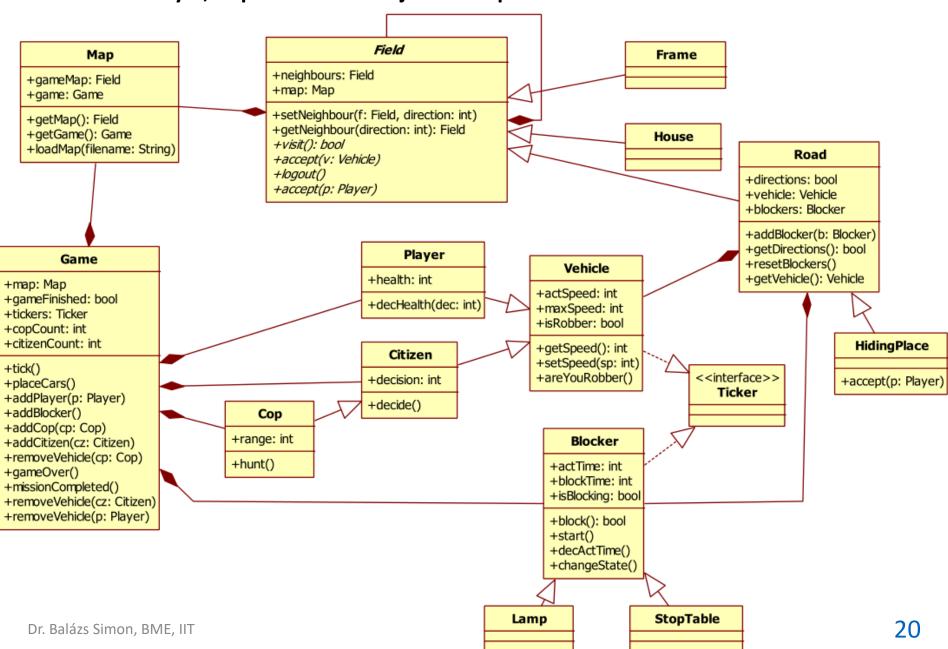
"Model for behavior not for roles!"



"Model for behavior not for roles!"

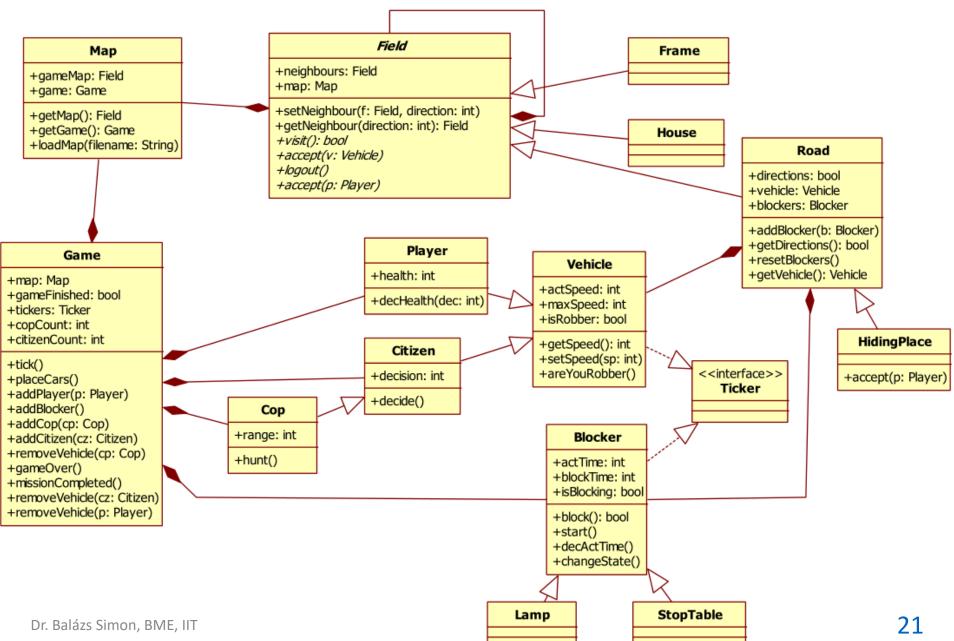


"Game stores Player, Cop and Citizen objects in separate collections."

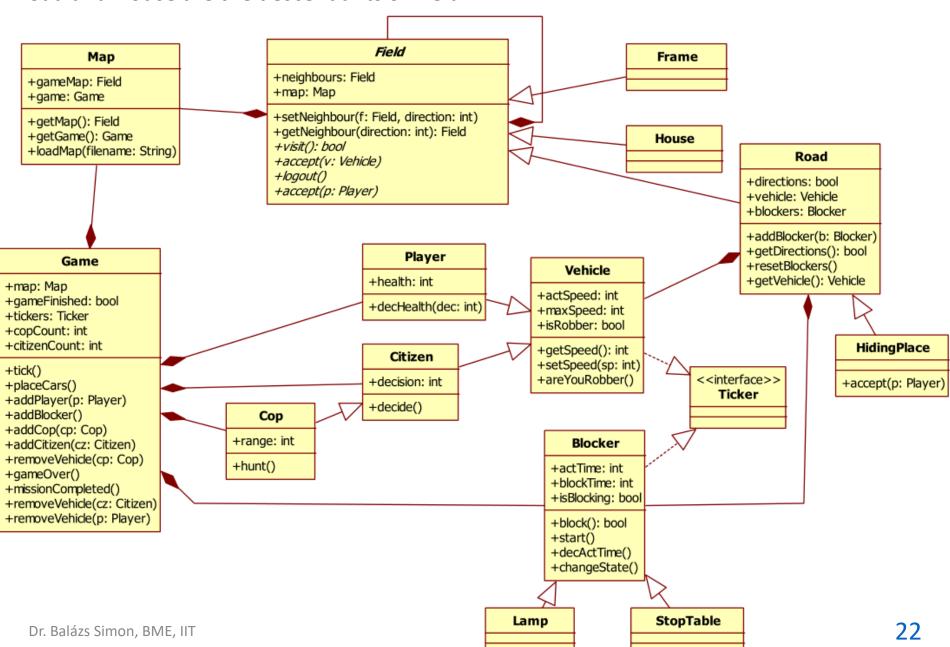


Solution: Incorrect

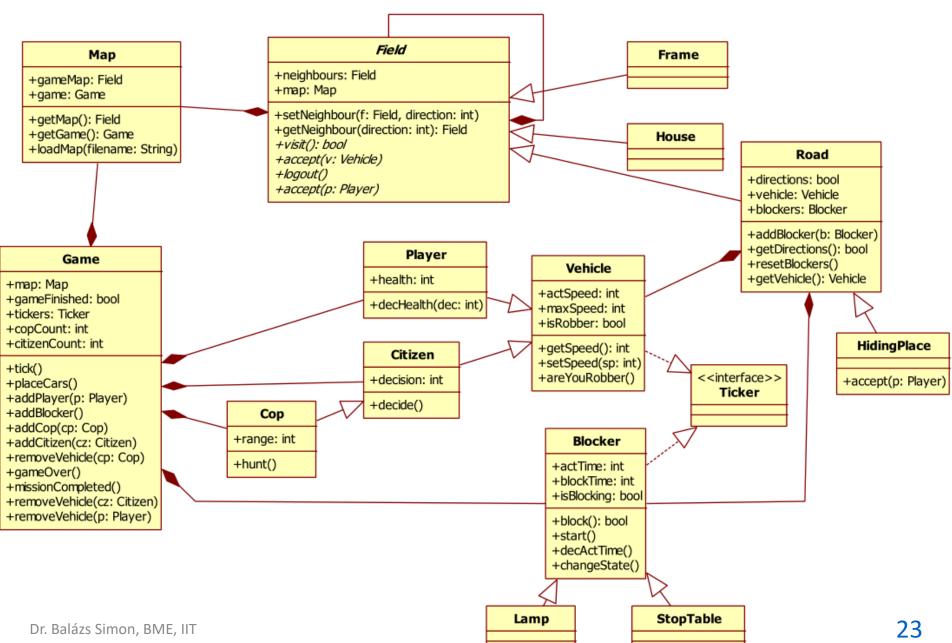
"Game stores Player, Cop and Citizen objects in separate collections."



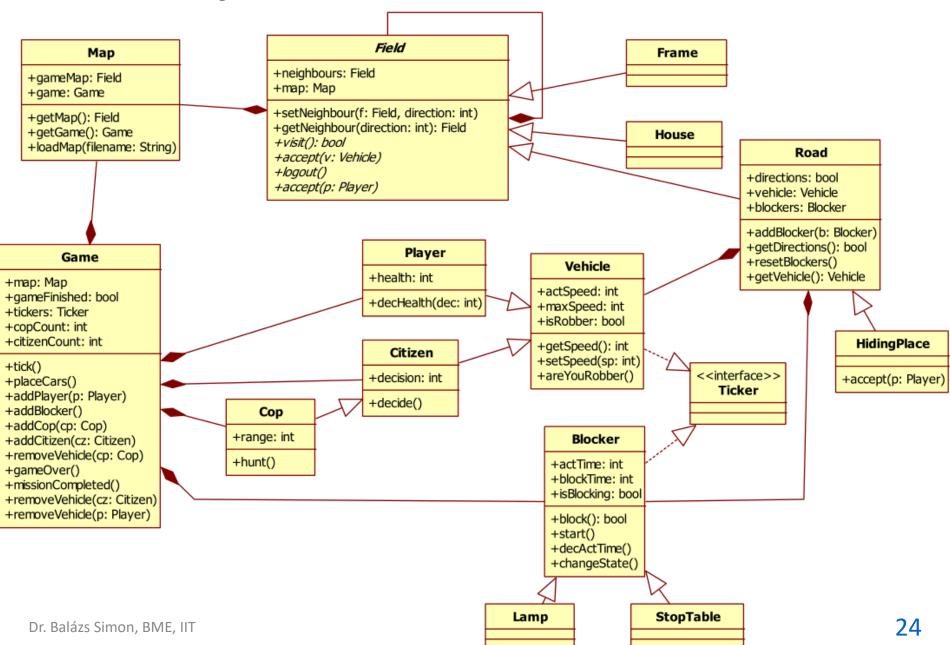
"Road and House are the descendants of Field."



"Road and House are the descendants of Field."

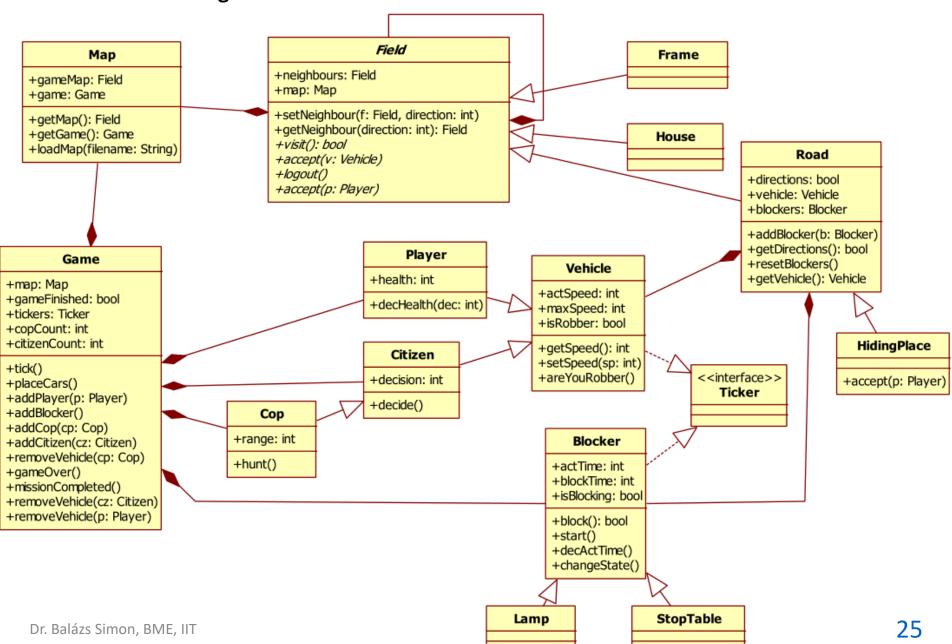


"Field contains its neighbors."

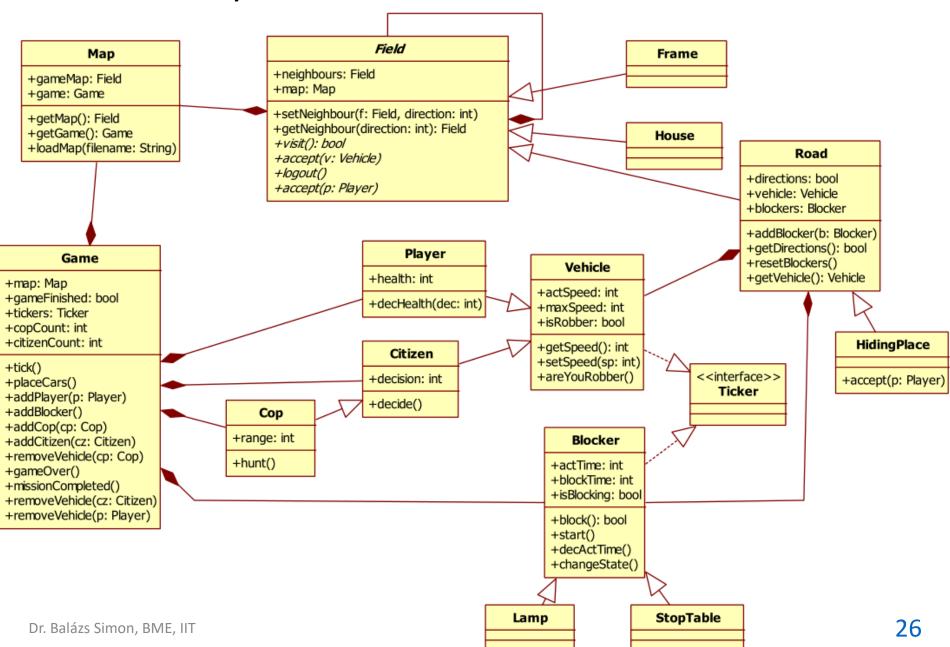


Solution: Incorrect

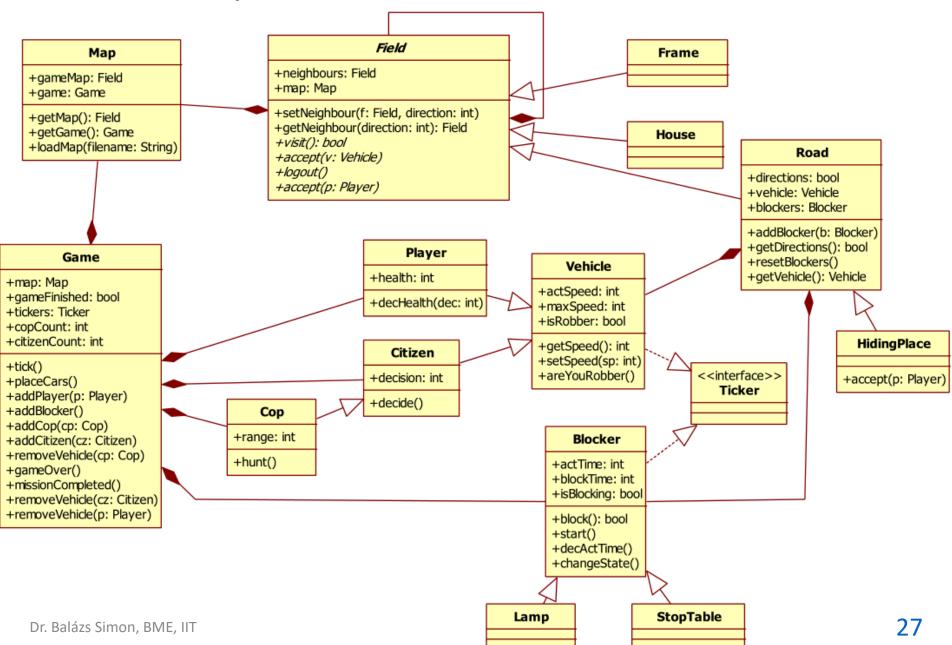
"Field contains its neighbors."



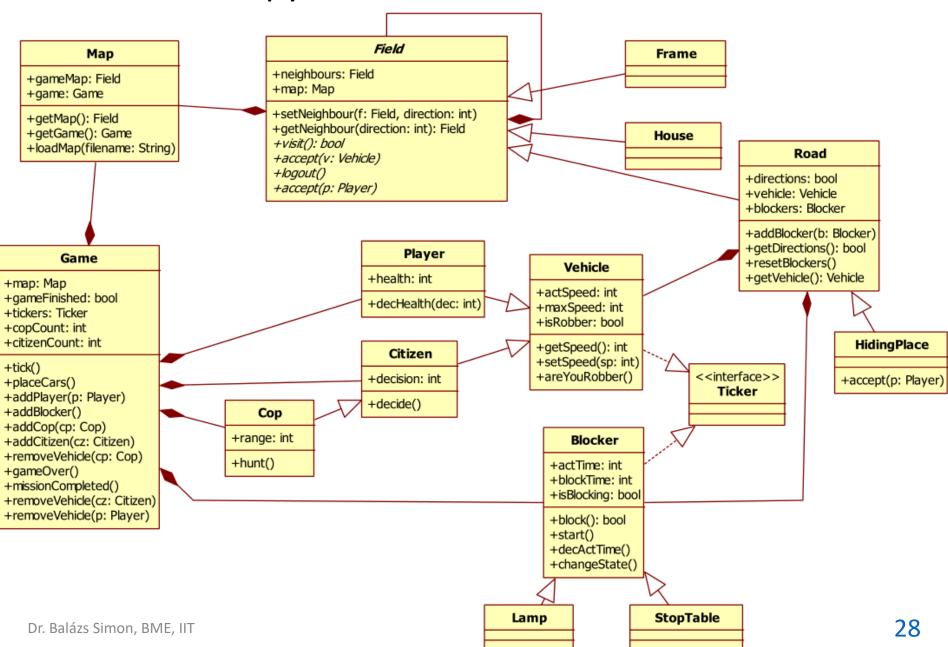
"Blocker is contained by Game and Road."



"Blocker is contained by Game and Road."

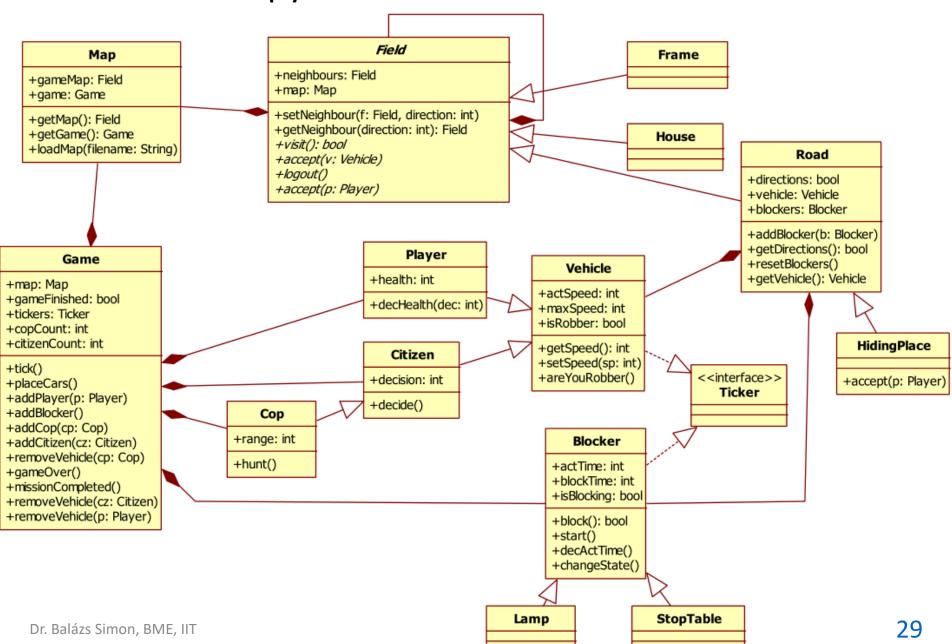


"The Ticker interface is empty."

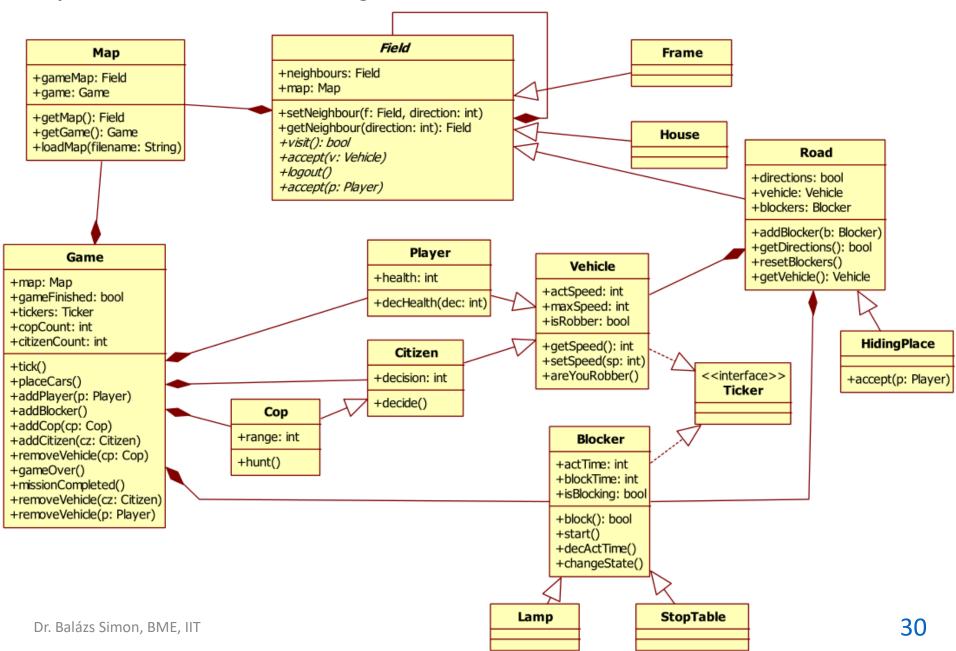


Solution: Incorrect

"The Ticker interface is empty."

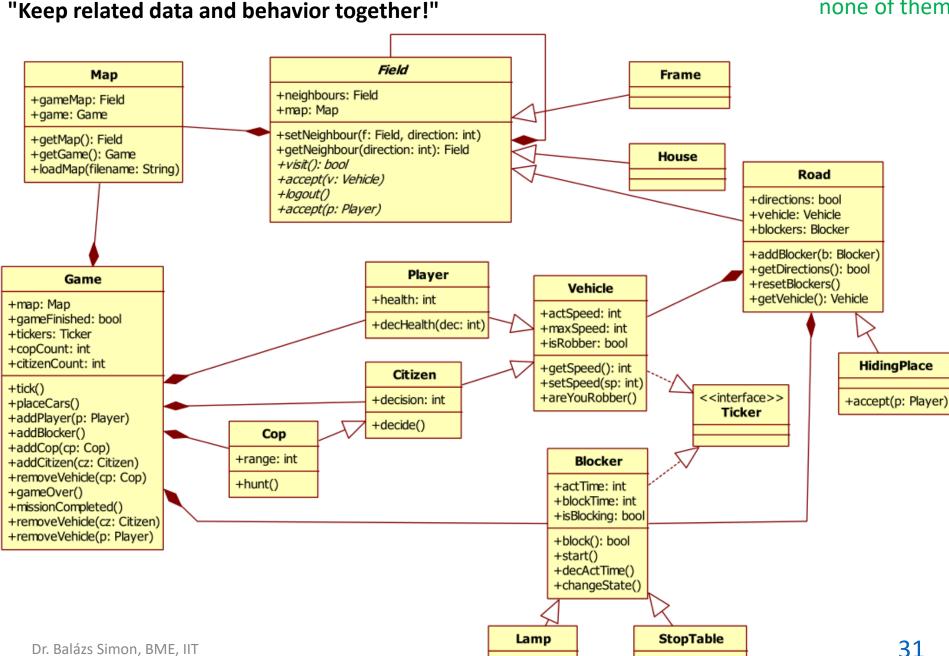


"Keep related data and behavior together!"

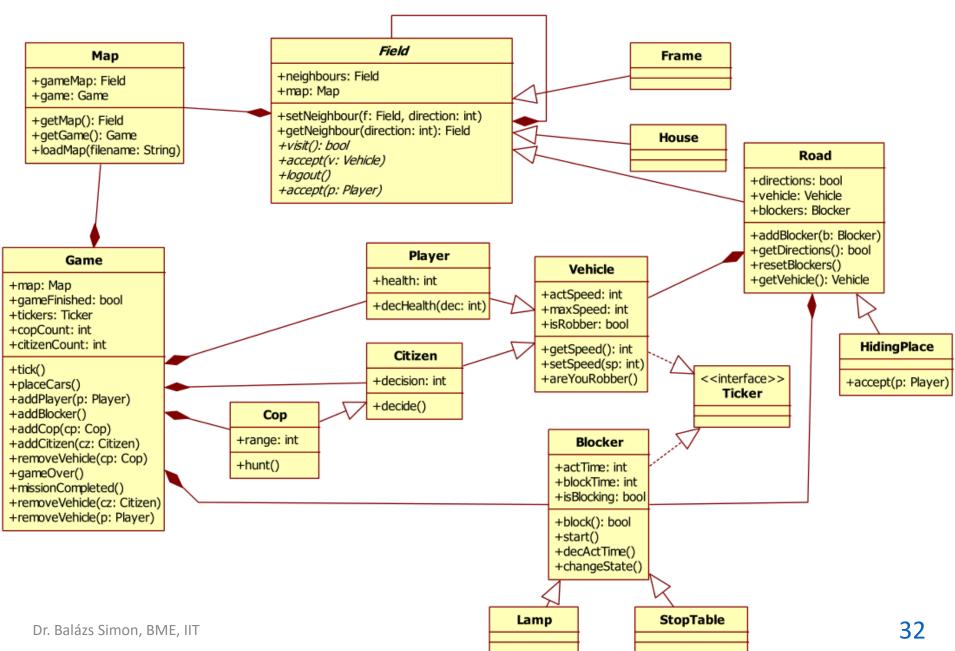


Solution:

none of them

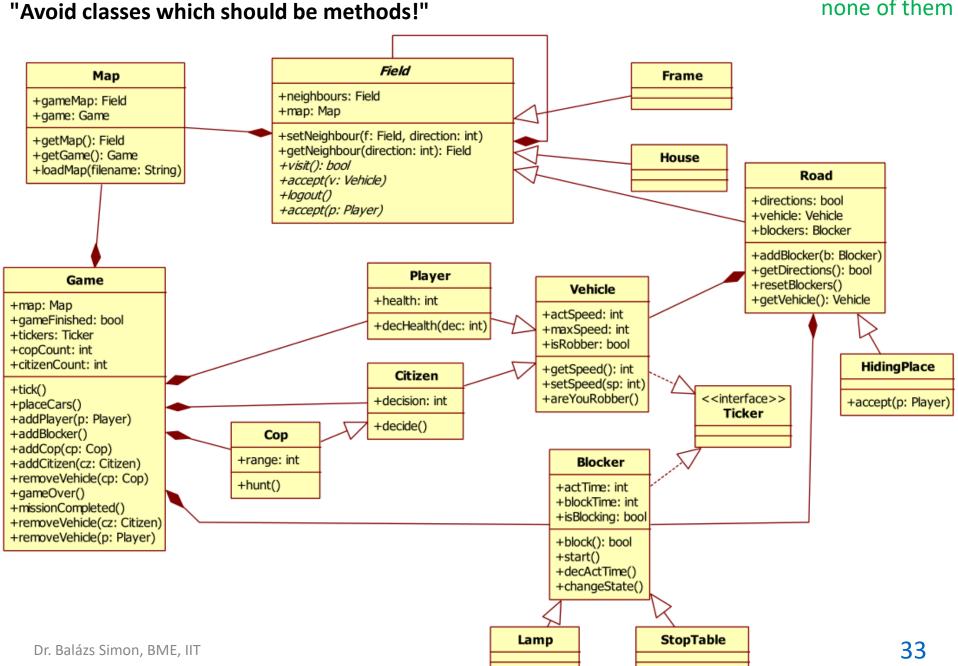


"Avoid classes which should be methods!"

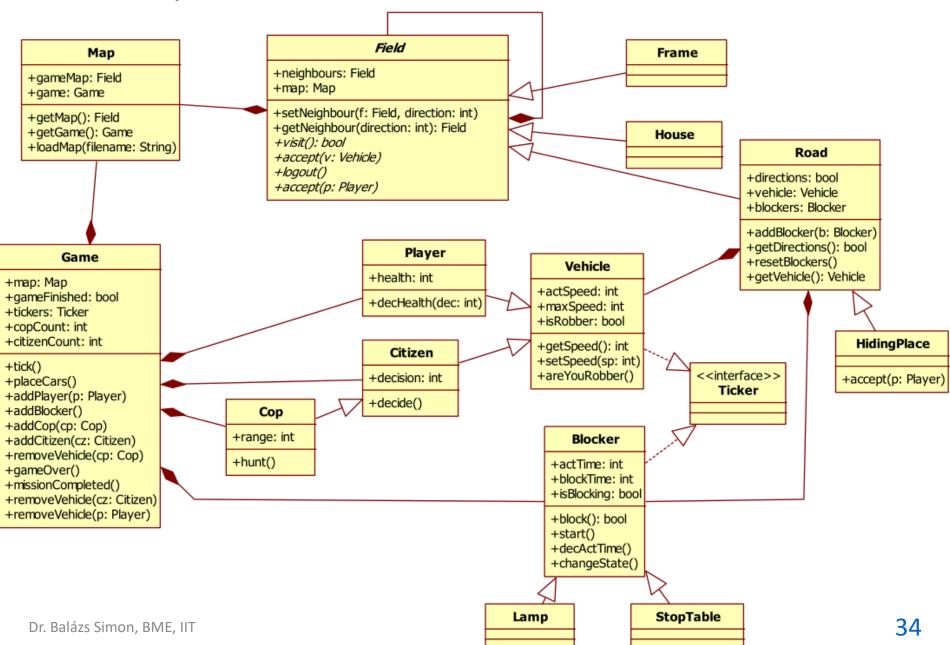


istic?

Solution:

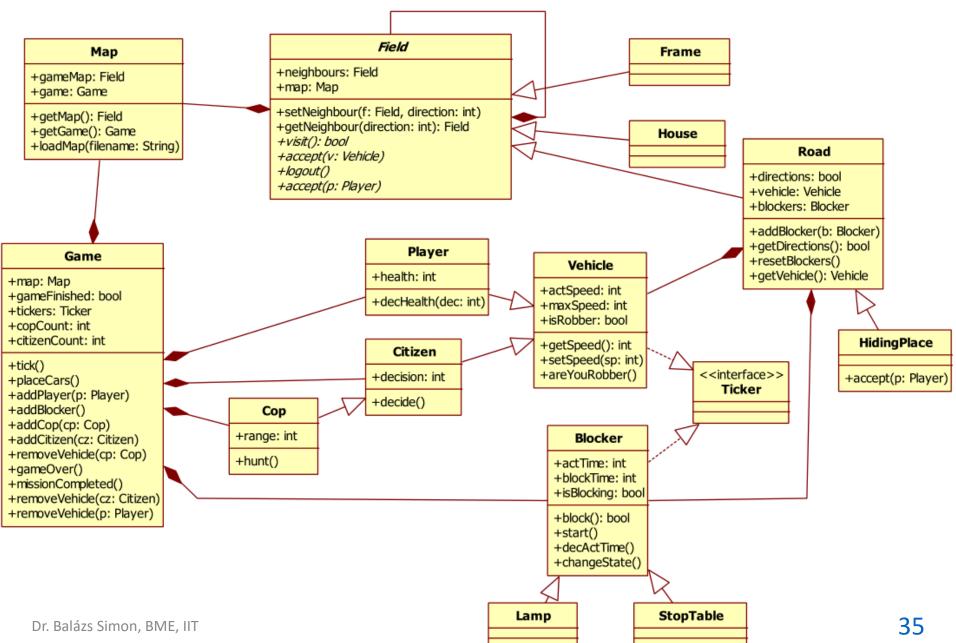


"Do not use non-public members of another class!"

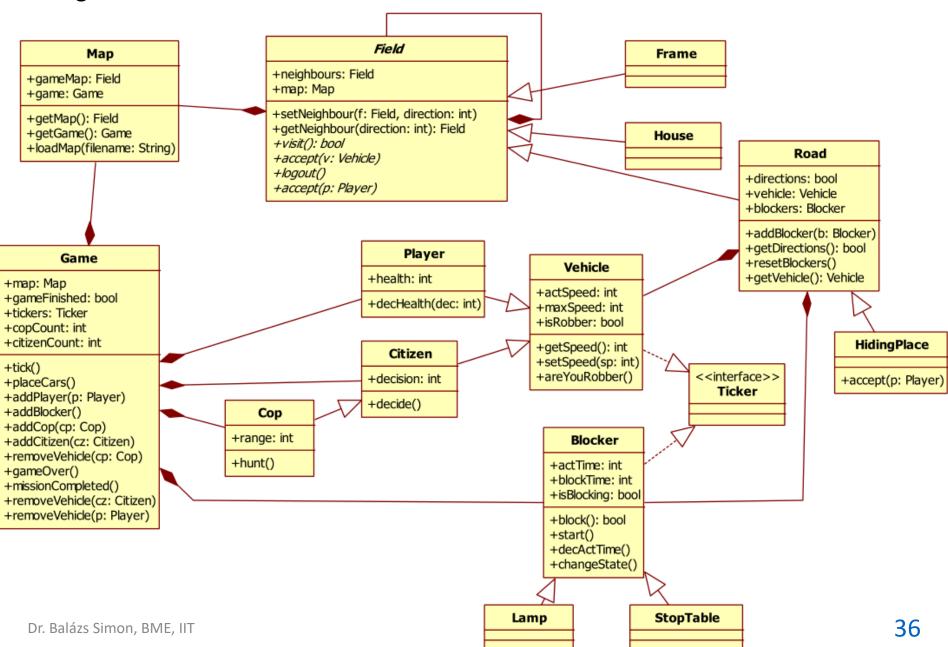


"Do not use non-public members of another class!"

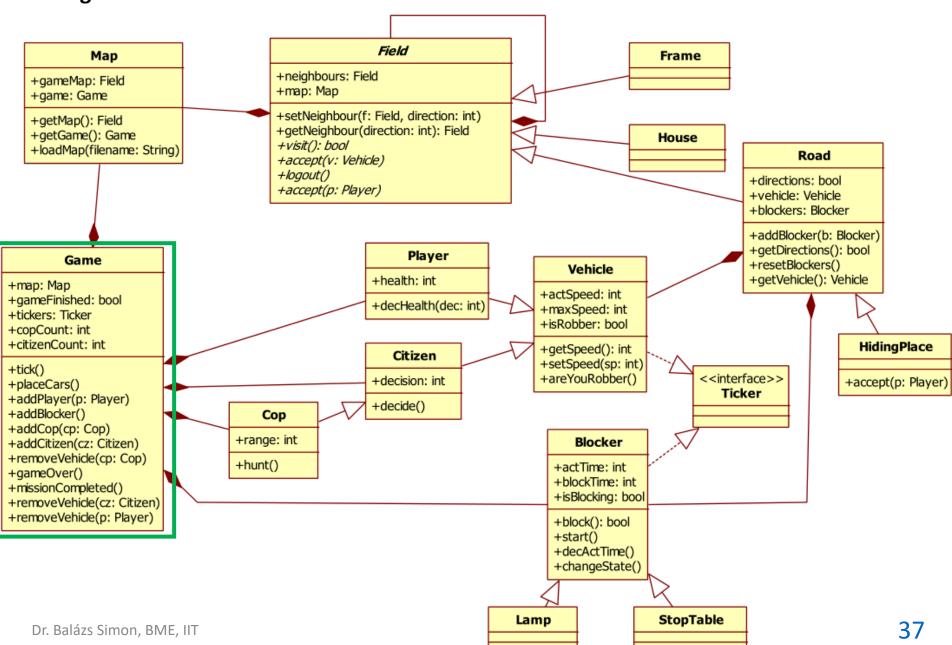
Solution: none of them



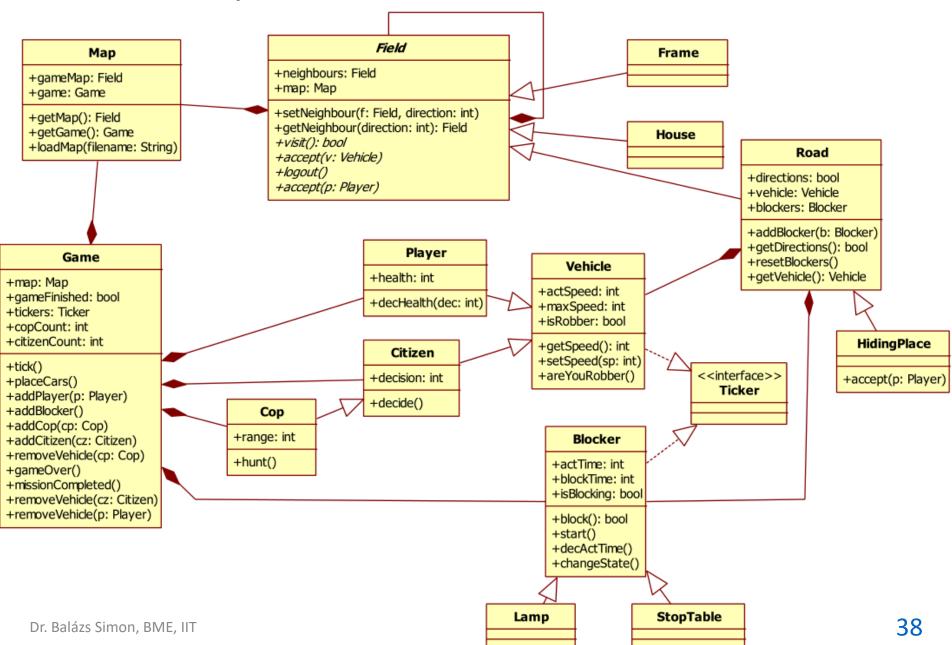
"Avoid god classes!"



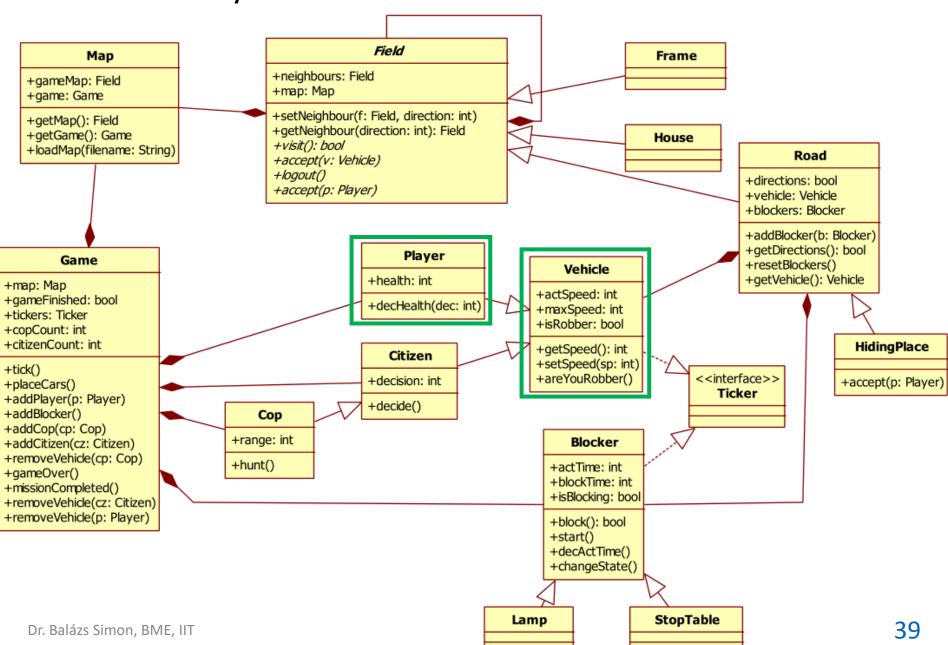
"Avoid god classes!"



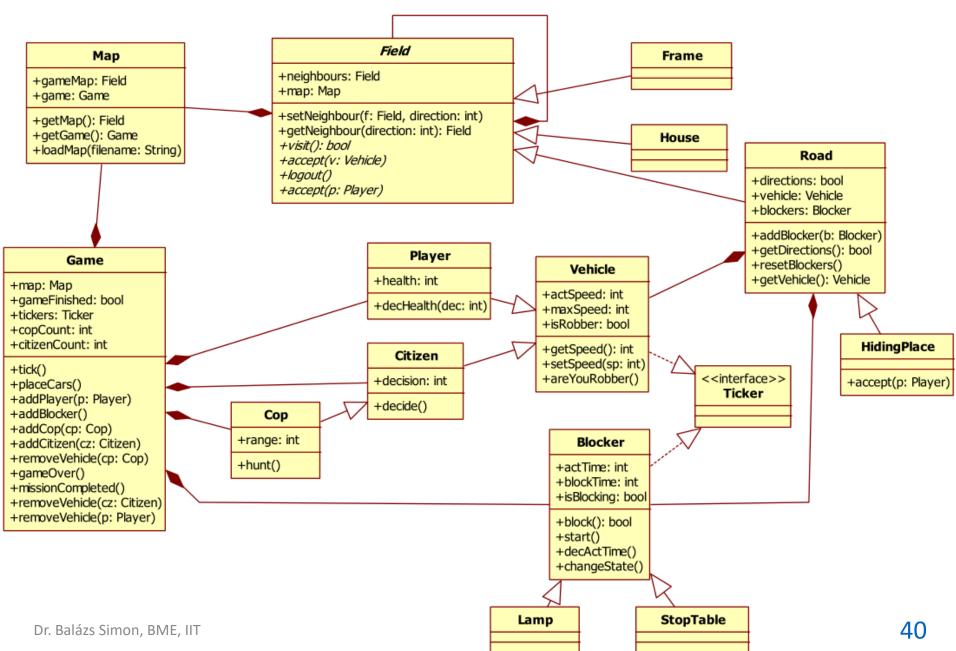
"Avoid classes with only accessor methods!"



"Avoid classes with only accessor methods!"

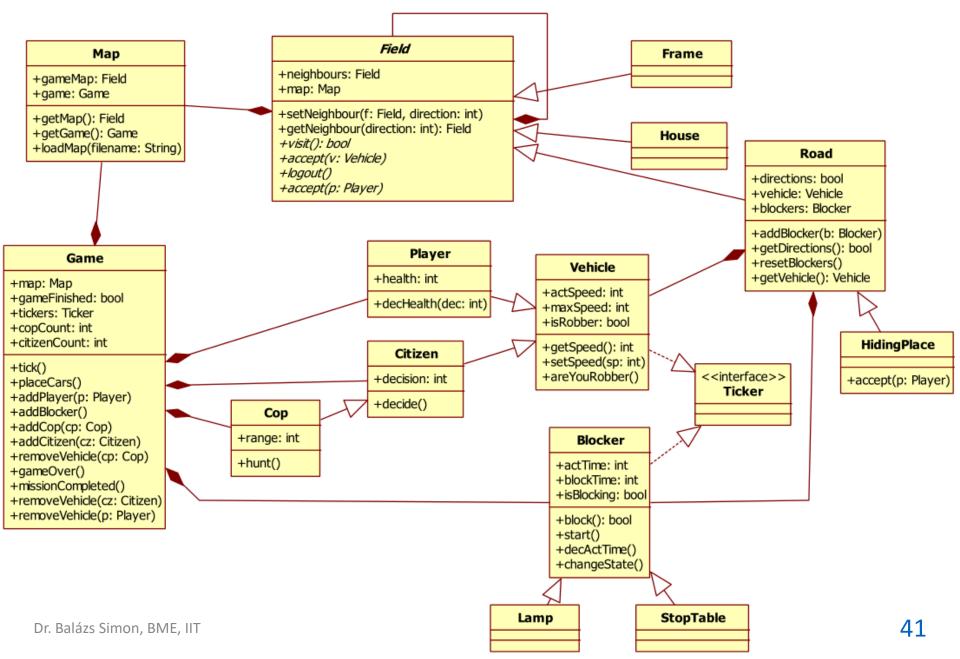


"Prefer containment over association!"

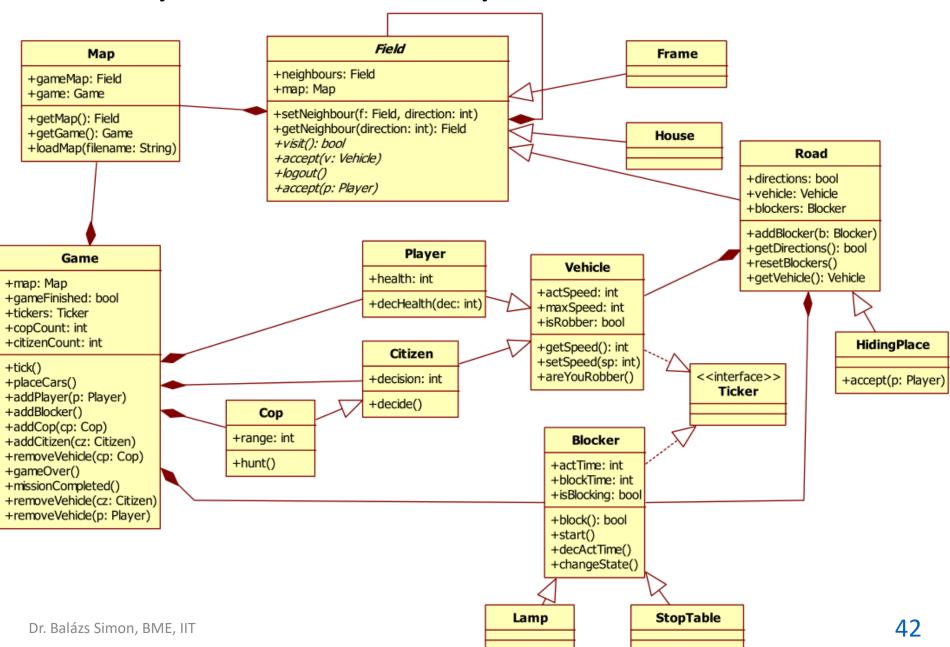


Solution:



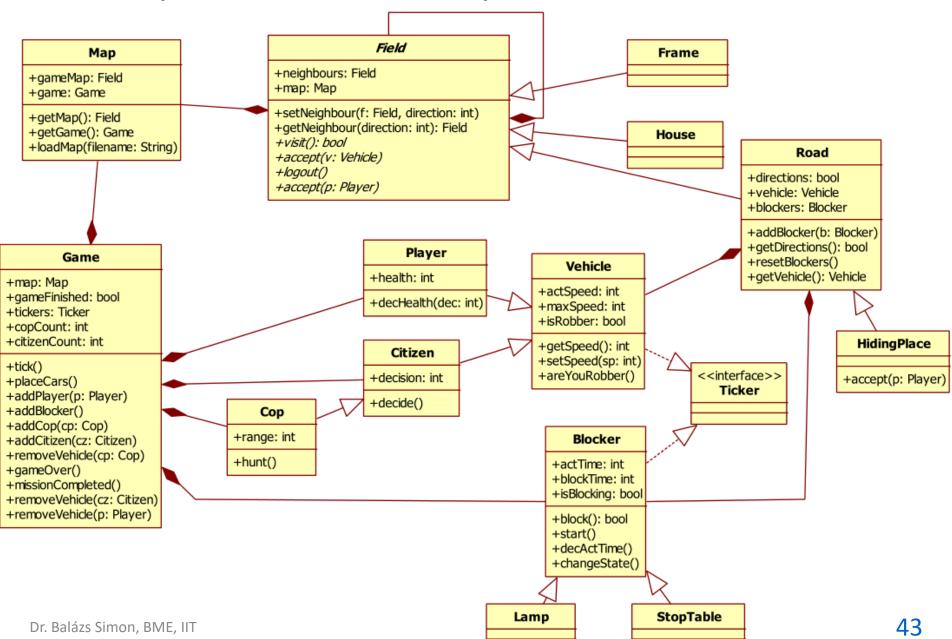


"A container object should use the contained objects!"

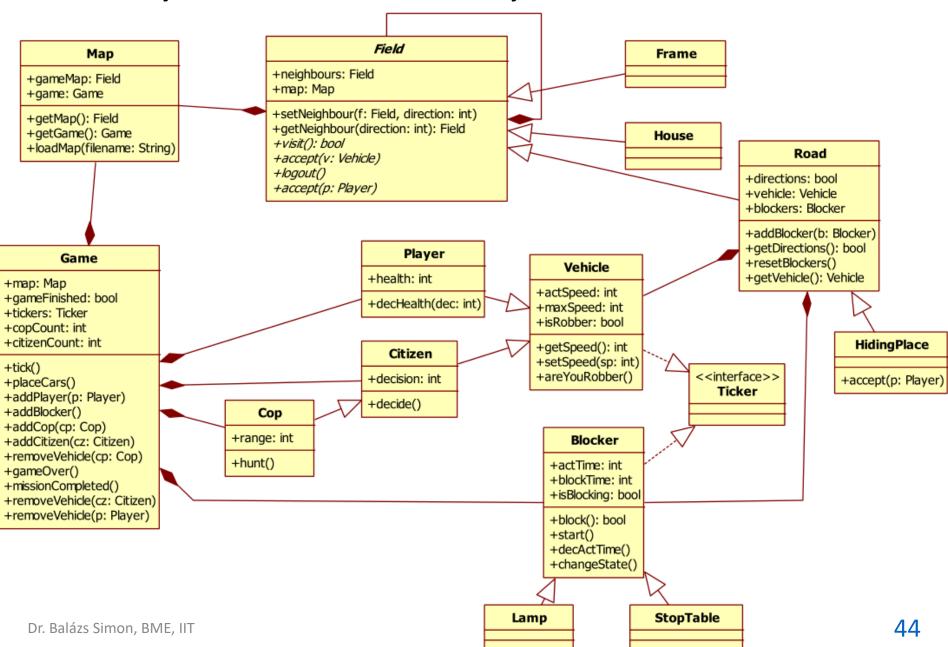


"A container object should use the contained objects!"

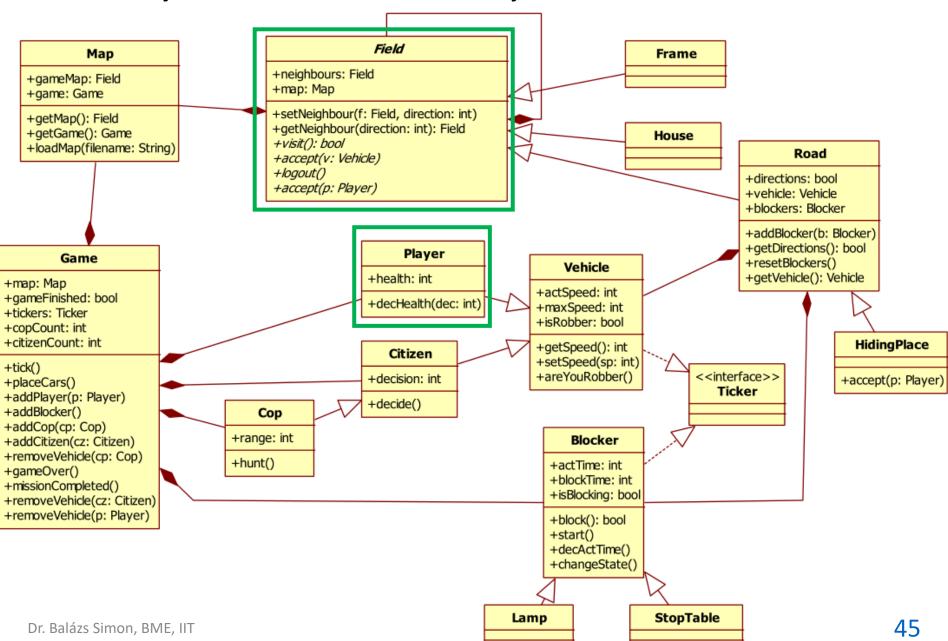
Solution: none of them



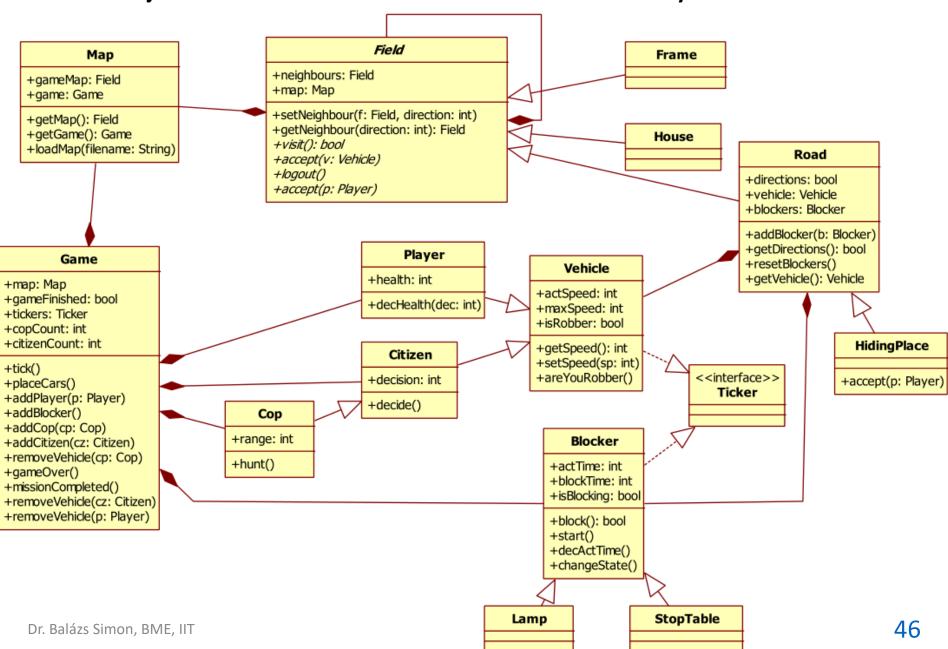
"A contained object should not use its container object!"



"A contained object should not use its container object!"

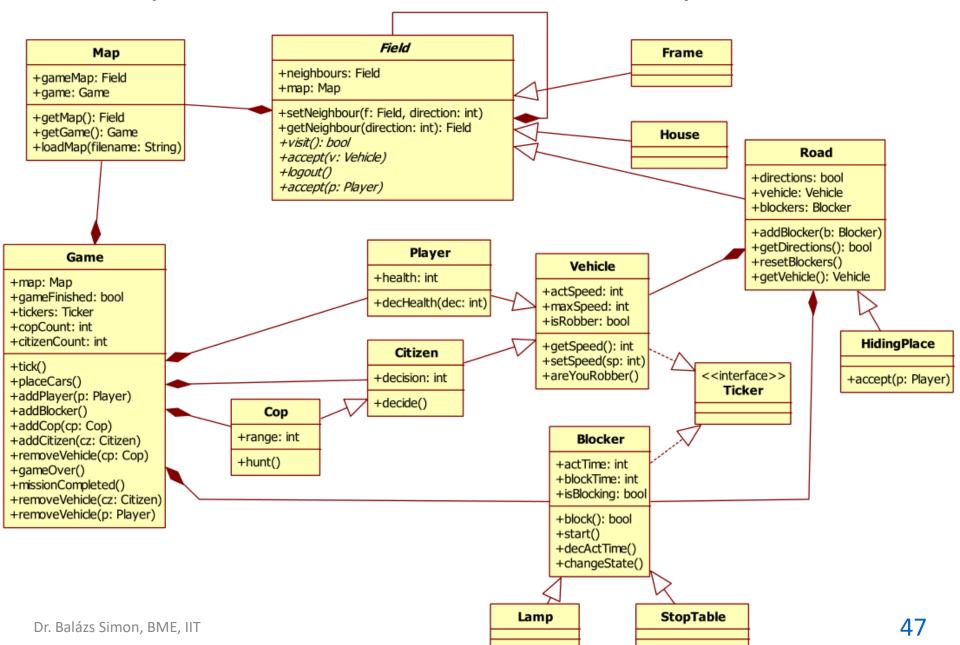


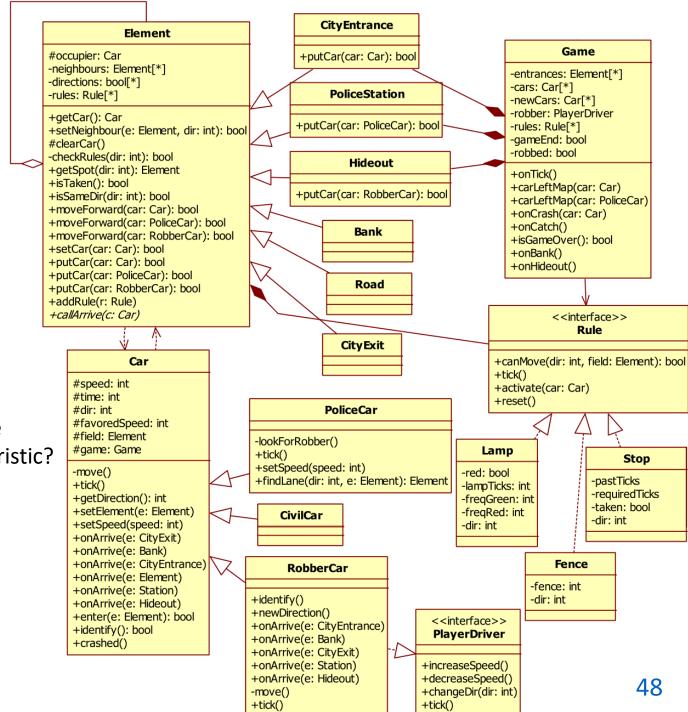
"Contained objects should not communicate with each other directly!"



"Contained objects should not communicate with each other directly!"

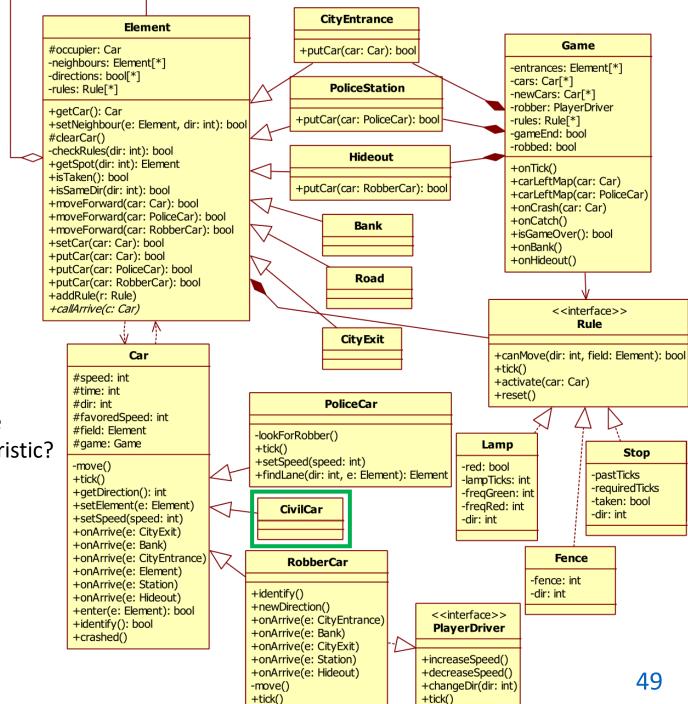
Solution: none of them





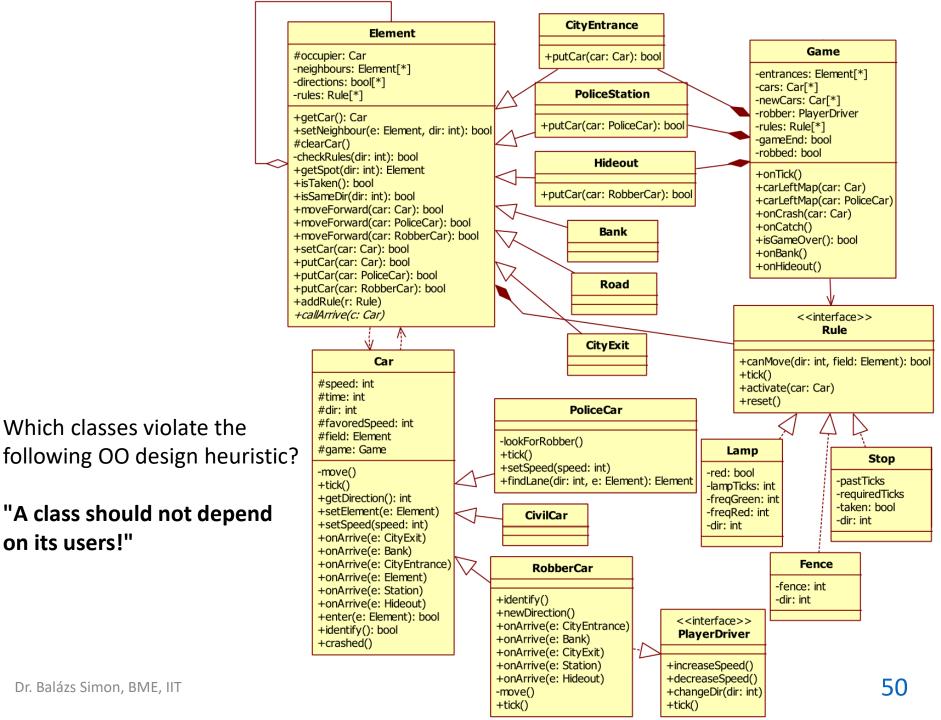
"Model for behavior not for roles!"

Solution

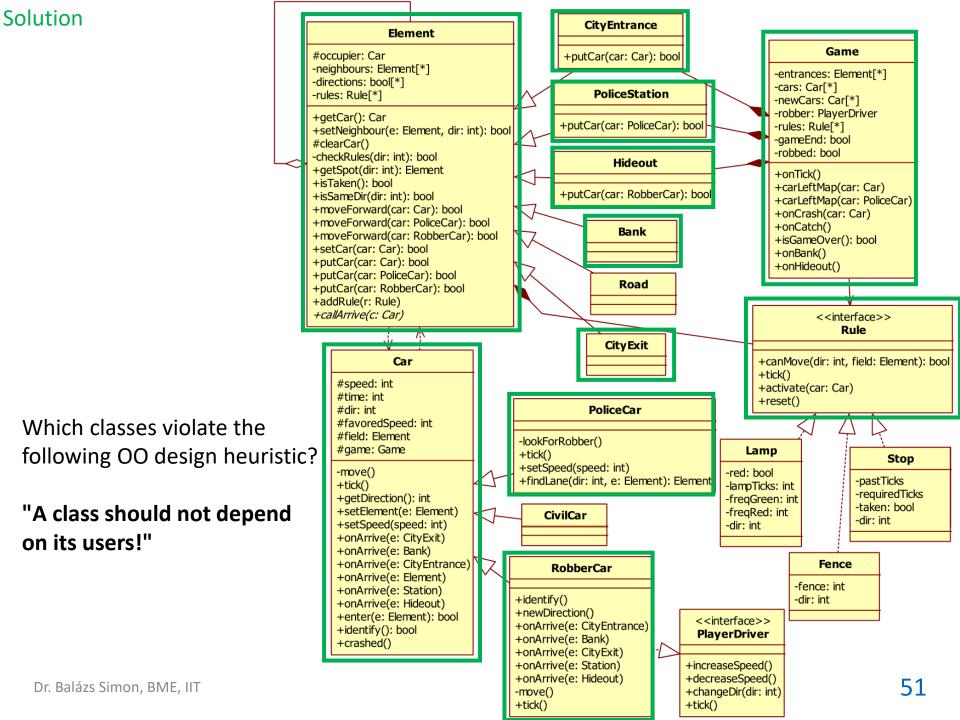


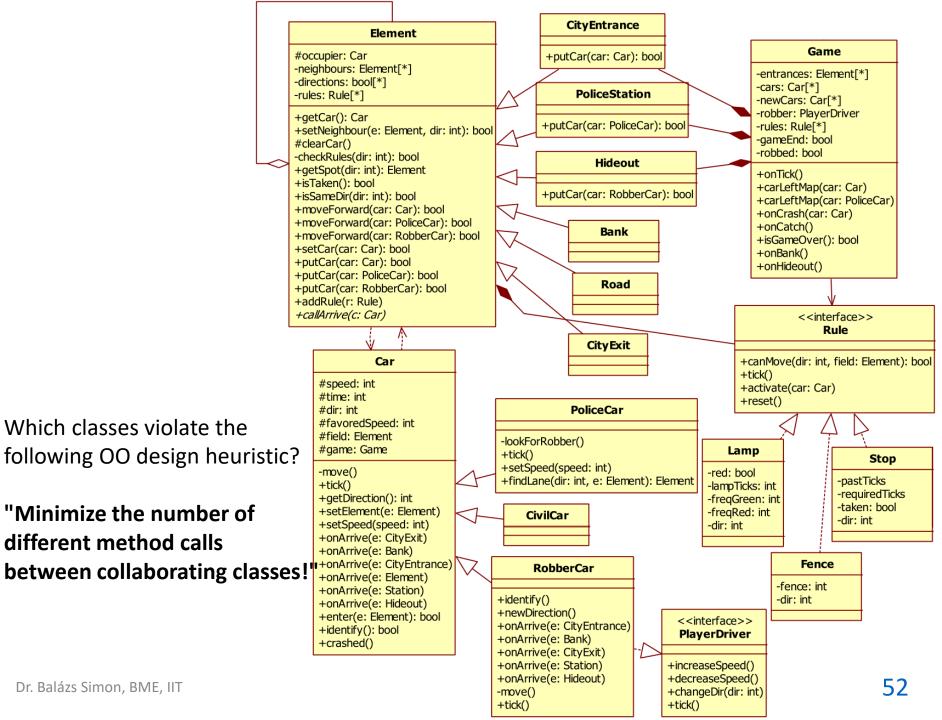
Which classes violate the following OO design heuristic?

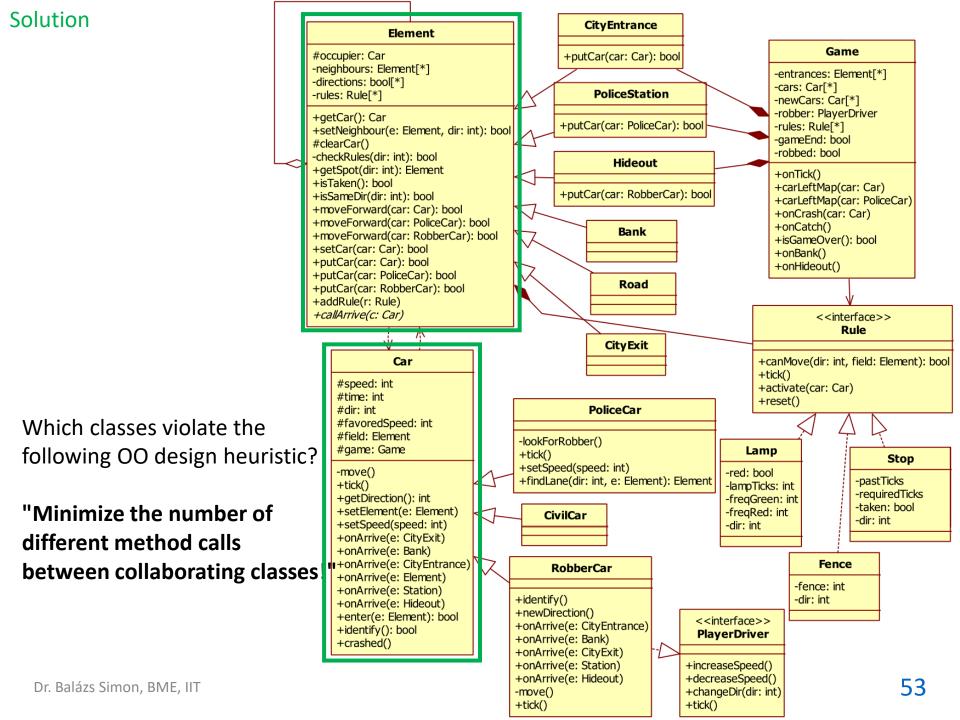
"Model for behavior not for roles!"

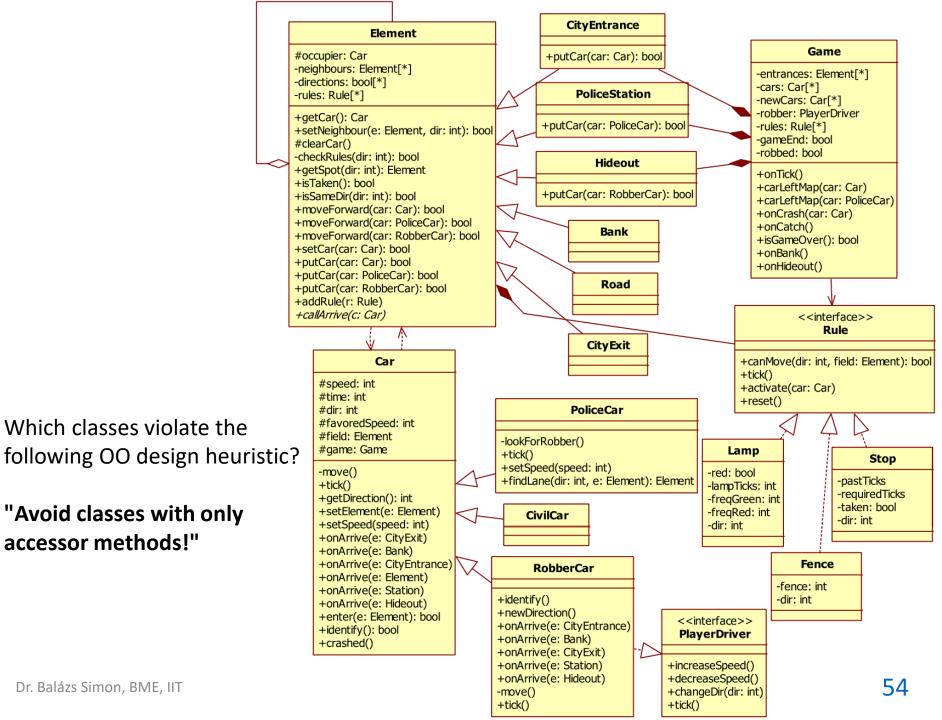


on its users!"



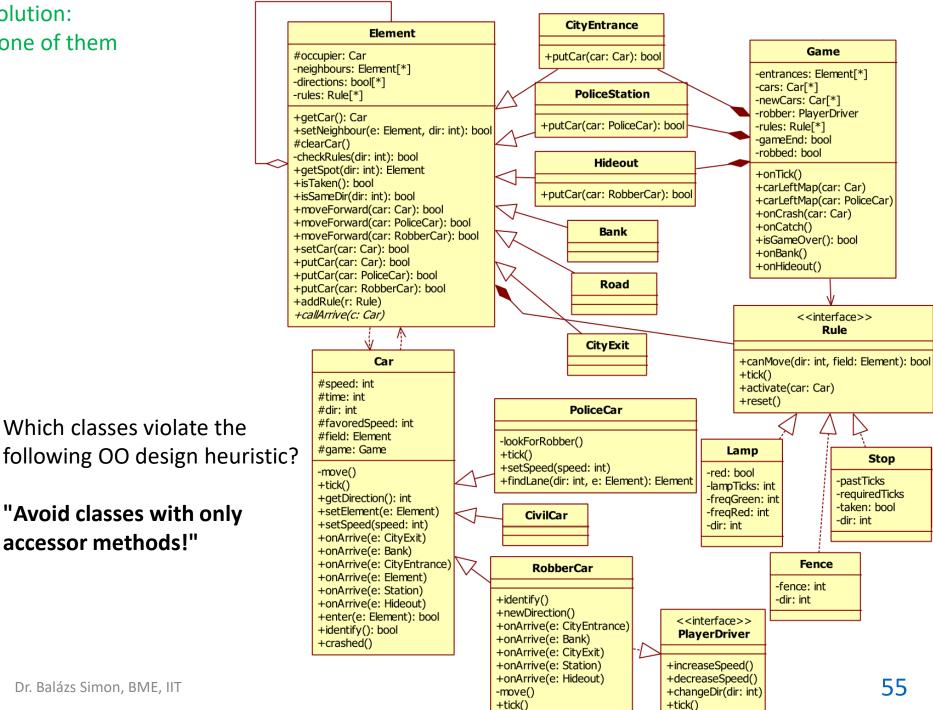






accessor methods!"

Solution: none of them

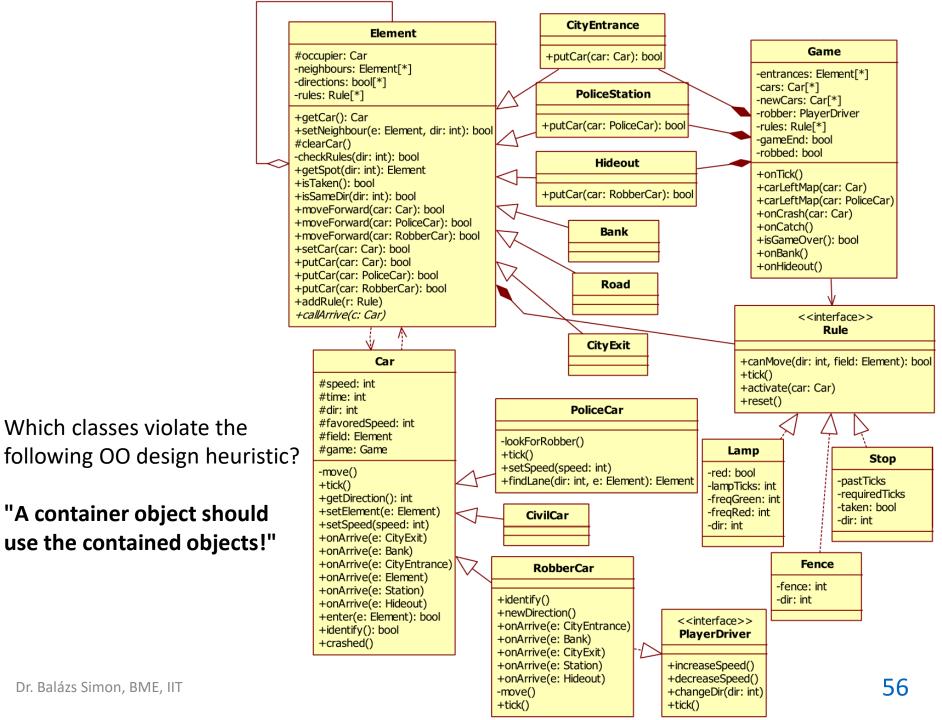


Stop

55

"Avoid classes with only

accessor methods!"



Solution CityEntrance Element Game #occupier: Car +putCar(car: Car): bool -neighbours: Element[*] -entrances: Element[*] -directions: bool[*] -cars: Car[*] **PoliceStation** -rules: Rule[*] -newCars: Car[*] -robber: PlayerDriver +getCar(): Car +putCar(car: PoliceCar): bool -rules: Rule[*] +setNeighbour(e: Element, dir: int): bool -gameEnd: bool #clearCar() -robbed: bool -checkRules(dir: int): bool Hideout +getSpot(dir: int): Element +onTick() +isTaken(): bool +carLeftMap(car: Car) +putCar(car: RobberCar): bool +isSameDir(dir: int): bool +carLeftMap(car: PoliceCar) +moveForward(car: Car): bool +onCrash(car: Car) +moveForward(car: PoliceCar): bool +onCatch() Bank +moveForward(car: RobberCar): bool +isGameOver(): bool +setCar(car: Car): bool +onBank() +putCar(car: Car): bool +onHideout() +putCar(car: PoliceCar): bool Road +putCar(car: RobberCar): bool +addRule(r: Rule) +callArrive(c: Car) <<interface>> Rule City Exit Car +canMove(dir: int, field: Element): bool +tick() #speed: int +activate(car: Car) #time: int +reset() **PoliceCar** #dir: int Which classes violate the #favoredSpeed: int #field: Element -lookForRobber() #game: Game Lamp following OO design heuristic? +tick() Stop +setSpeed(speed: int) -move() -red: bool +findLane(dir: int, e: Element): Element -pastTicks +tick() -lampTicks: int -requiredTicks +getDirection(): int -fregGreen: int -taken: bool "A container object should +setElement(e: Element) -fregRed: int CivilCar -dir: int +setSpeed(speed: int) -dir: int +onArrive(e: CityExit) use the contained objects!" +onArrive(e: Bank) +onArrive(e: CityEntrance) **Fence** RobberCar +onArrive(e: Element) -fence: int +onArrive(e: Station) +identify() -dir: int +onArrive(e: Hideout) +newDirection() +enter(e: Element): bool <<interface>> +onArrive(e: CityEntrance) +identify(): bool **PlayerDriver** +onArrive(e: Bank) +crashed()

+onArrive(e: CityExit) +onArrive(e: Station)

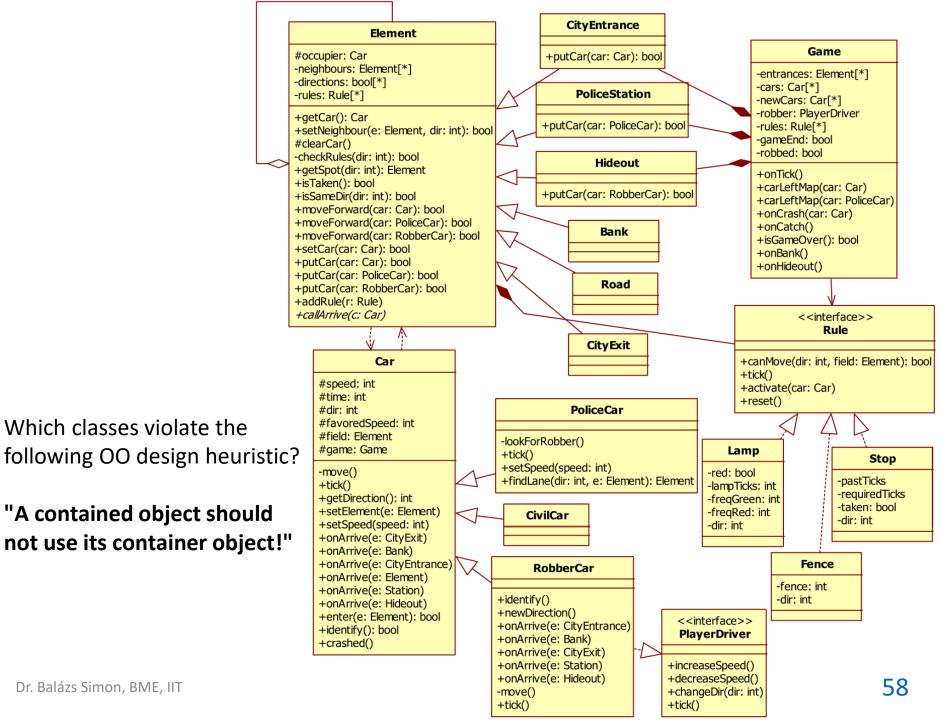
+onArrive(e: Hideout)

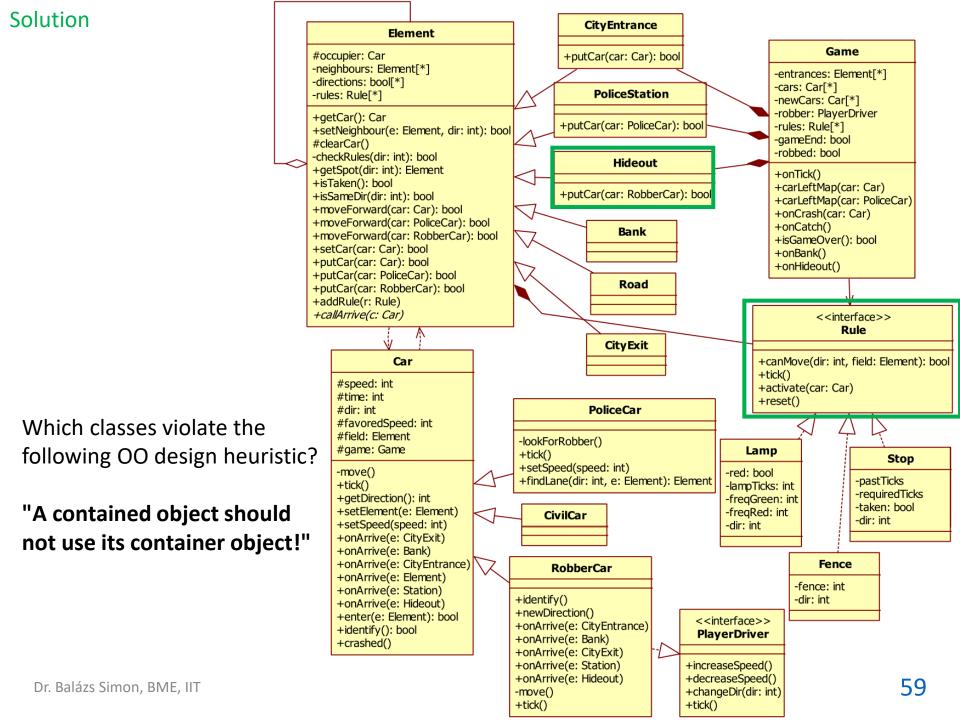
-move() +tick() +increaseSpeed()

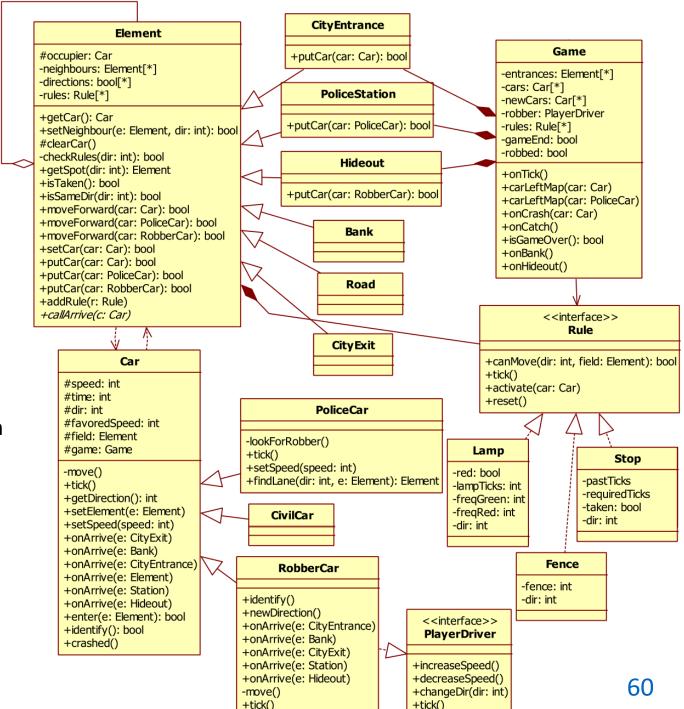
+decreaseSpeed()

+changeDir(dir: int)

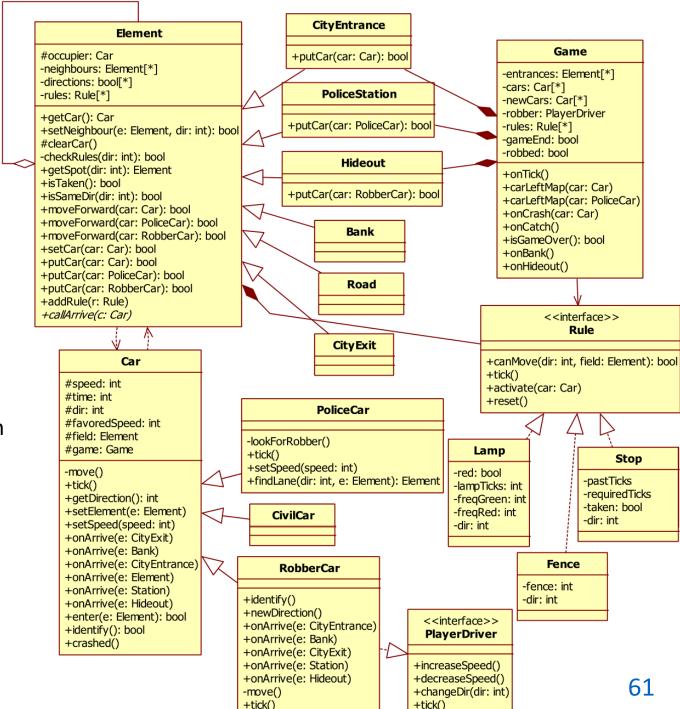
+tick()





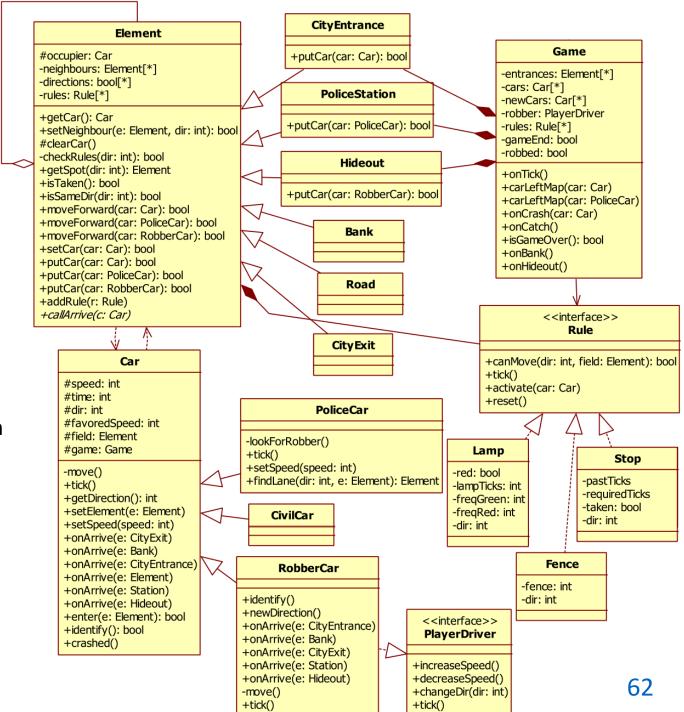


"Road is a descendant of Element."

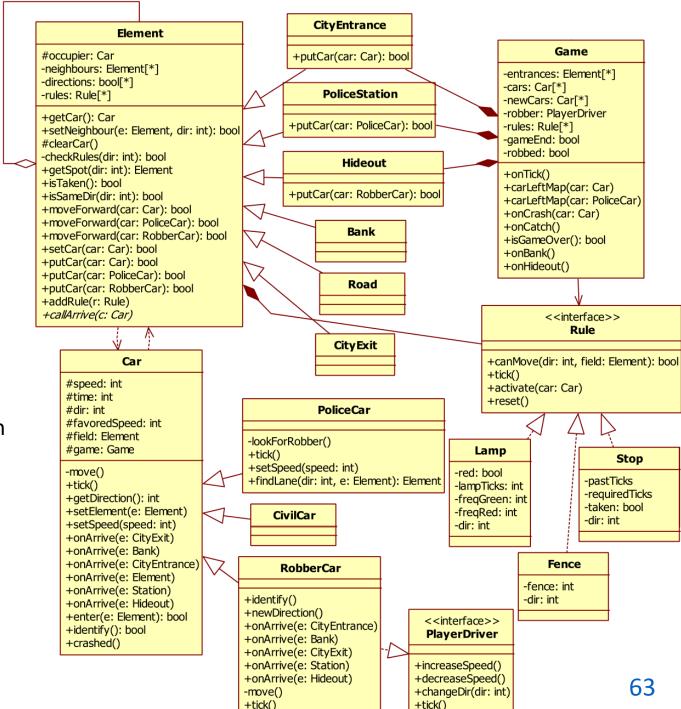


"Road is a descendant of Element."

Solution: Correct

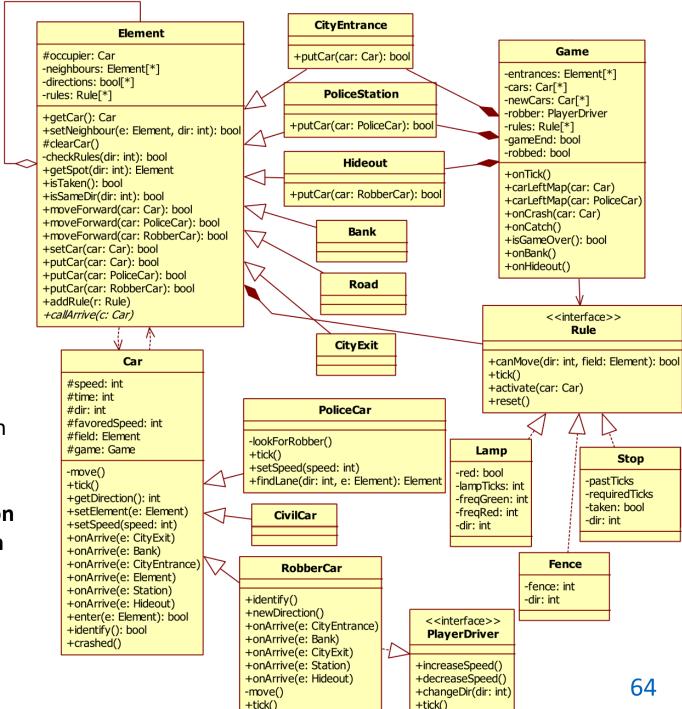


"Game does not store Elements."

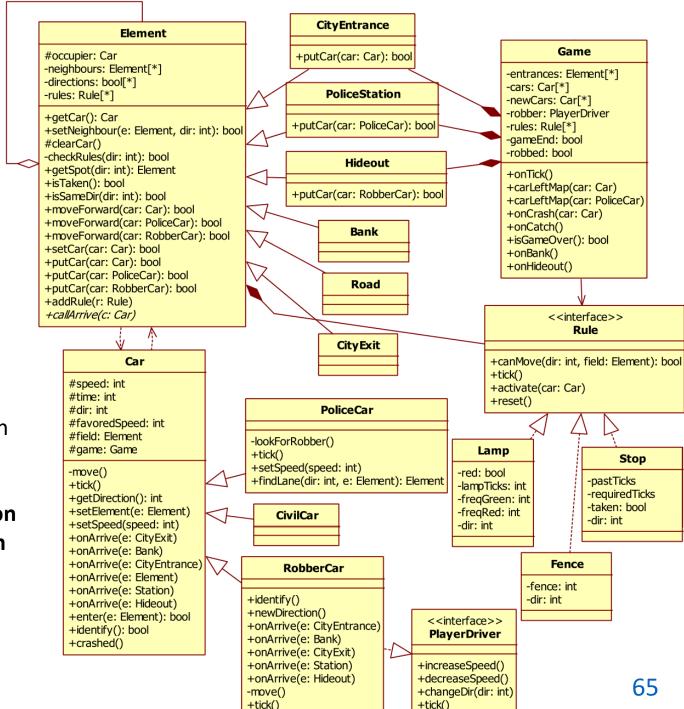


"Game does not store Elements."

Solution: Incorrect

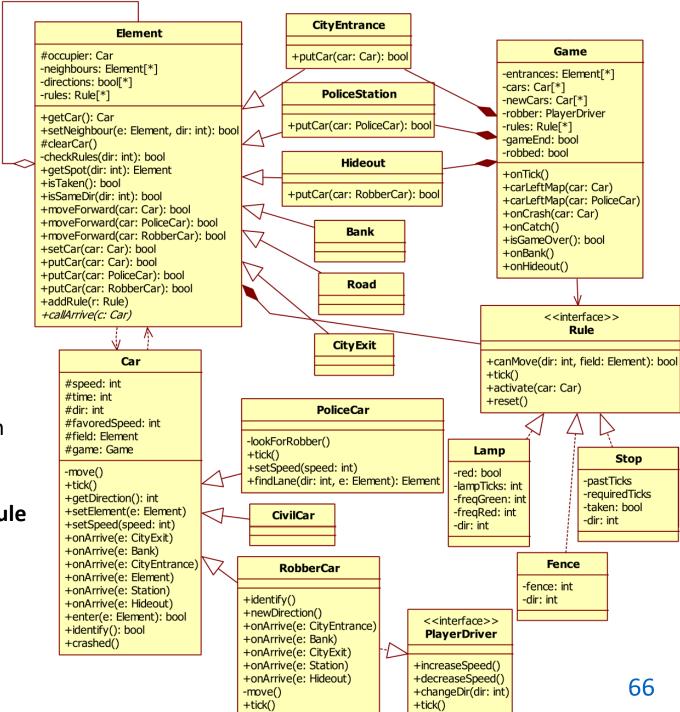


"Element only depends on Car, but does not have an association for a Car. "

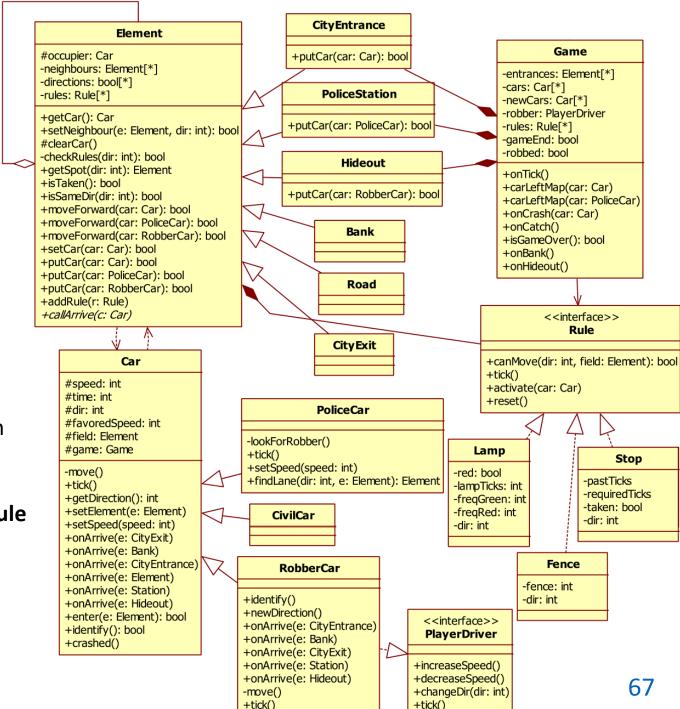


"Element only depends on Car, but does not have an association for a Car."

Solution: Incorrect



"Lamp implements the Rule interface."



"Lamp implements the Rule interface."

Solution: Correct