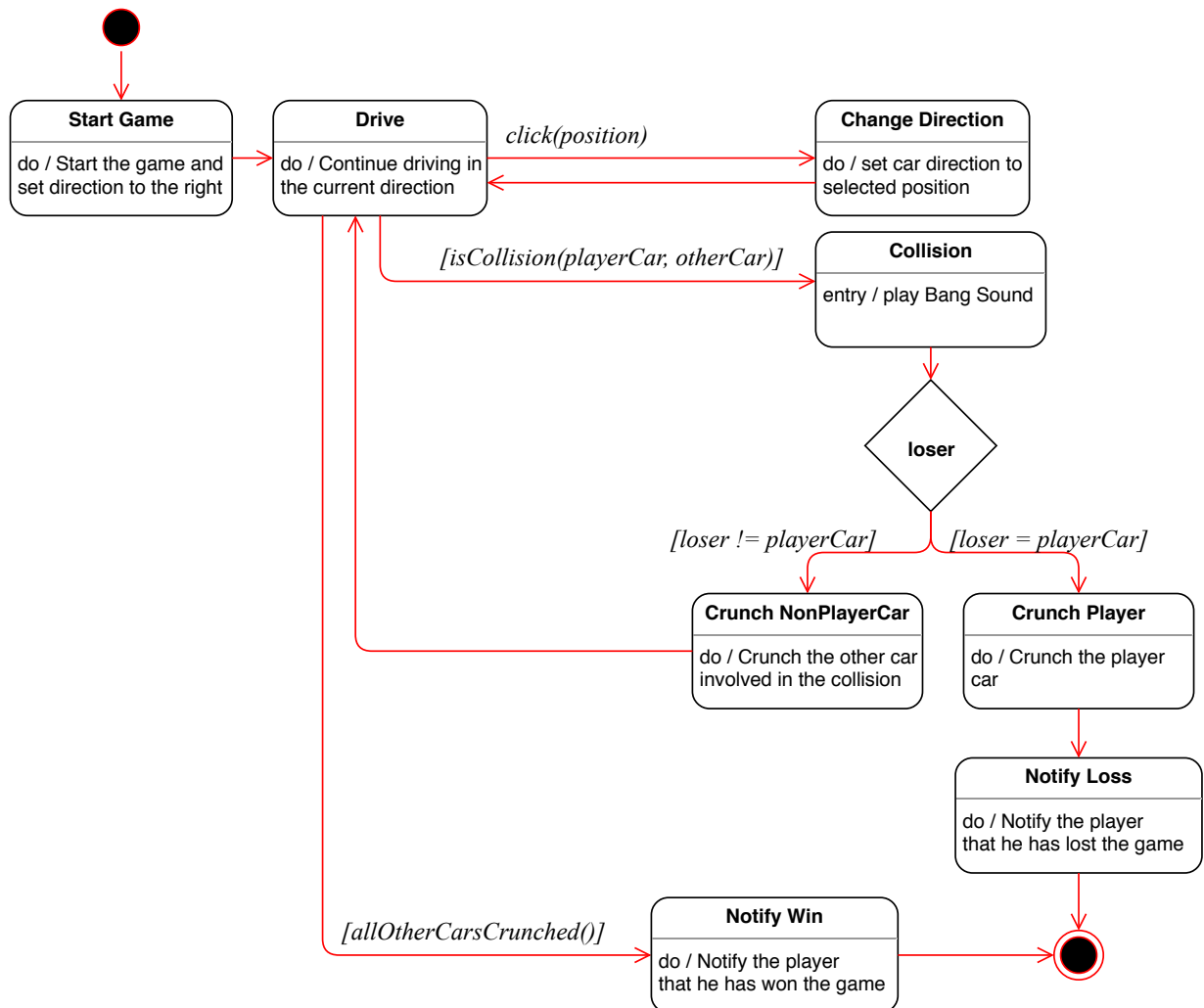# Einführung in die Softwaretechnik 2018
Sheet 04

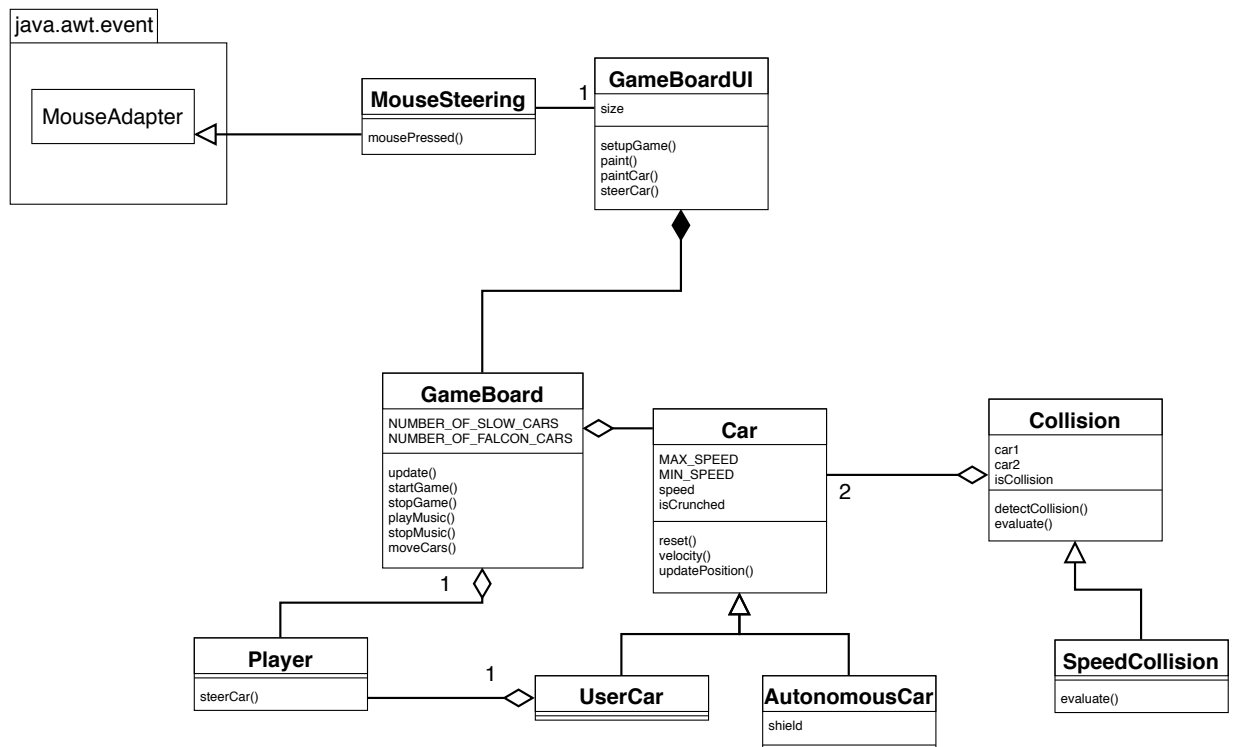## Maximilian Frühauf

13th May 2018
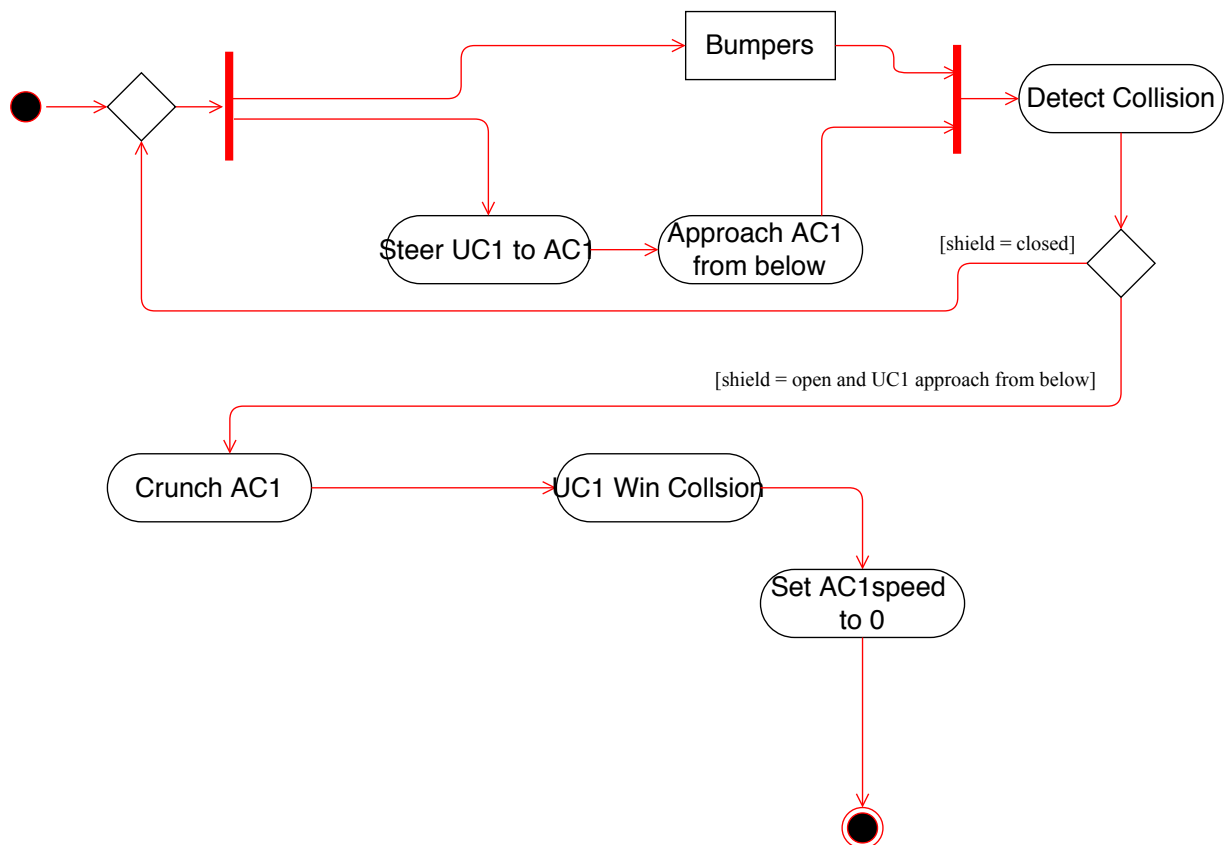
1. Dynamic behavior:
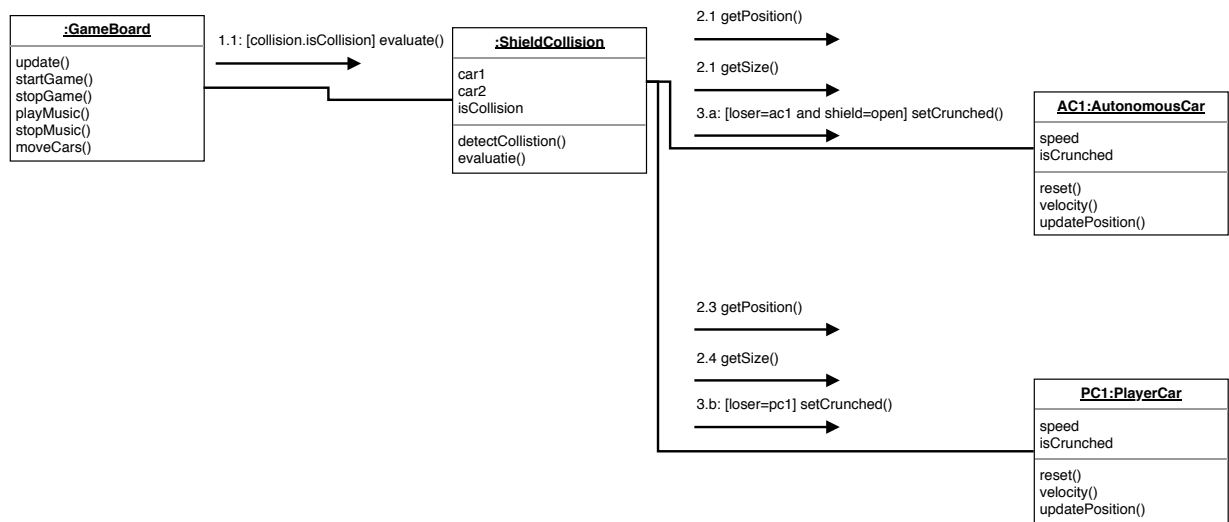

2. Taxonomy for the car shield use case:

3. Win Collision activity diagram:

4. Collision detection algorithm communication diagram:

:GameBoard

update()
startGame()
stopGame()
playMusic()
stopMusic()
moveCars()

1.1: [collision.isCollision] evaluate()

:ShieldCollision

car1
car2
isCollision

detectCollistion()
evaluatie()

2.1 getPosition()

2.1 getSize()

3.a: [loser=ac1 and shield=open] setCrunched()

AC1:AutonomousCar

speed
isCrunched

reset()
velocity()
updatePosition()

2.3 getPosition()

2.4 getSize()

3.b: [loser=pc1] setCrunched()

PC1:PlayerCar

speed
isCrunched

reset()
velocity()
updatePosition()

5. Coupling measures the dependency between different subsystems inside a project. High coupling therefore implies that changes to one subsystem will have a large impact on another subsystem. Low coupling on the other hand means that, a change in one subsystem does not affect others.

   Cohesion describes the dependency among two classes. Therefore high cohesion describes a subsystem where the classes perform similar tasks and are related to each other in many associations. Low cohesion is the inverse of the above. A subsystem with many classes that handle very different tasks and have few to no associations with each other.

   These terms can be used to describe a good system design, as a good system should have high cohesion and low coupling between the different submodules.

   An Example of this would be the *car* package in the Bumpers game. Here Cohesion is used between the different car types in the form of inheritance. This provides a close relationship between the different objects and ensures that the participating classes are concerned with only a single purpose, the details of modeling a car.

   A low coupling is then used to connects this *car* package to the other subsystems of the game.