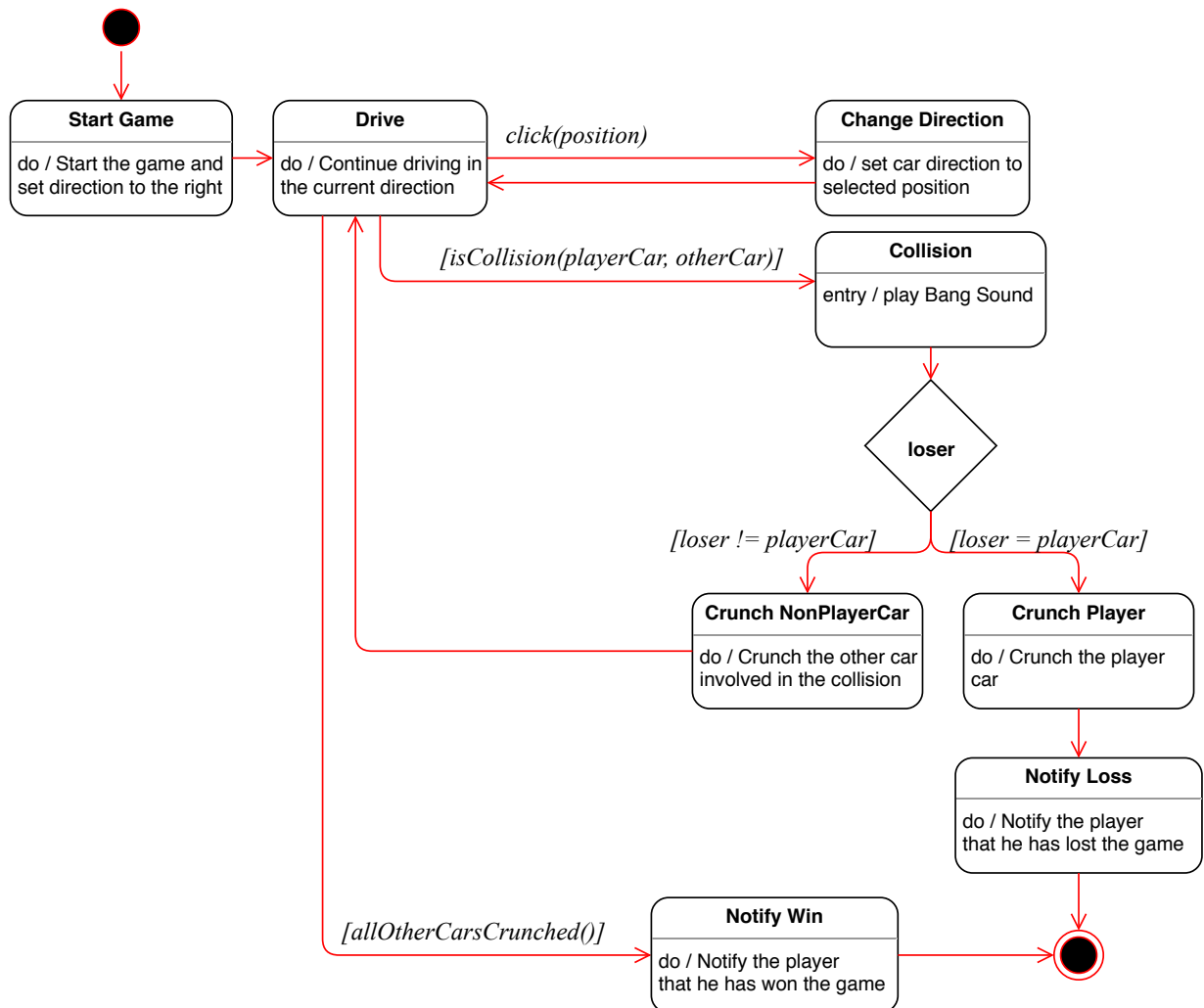# Einführung in die Softwaretechnik 2018
Sheet 04

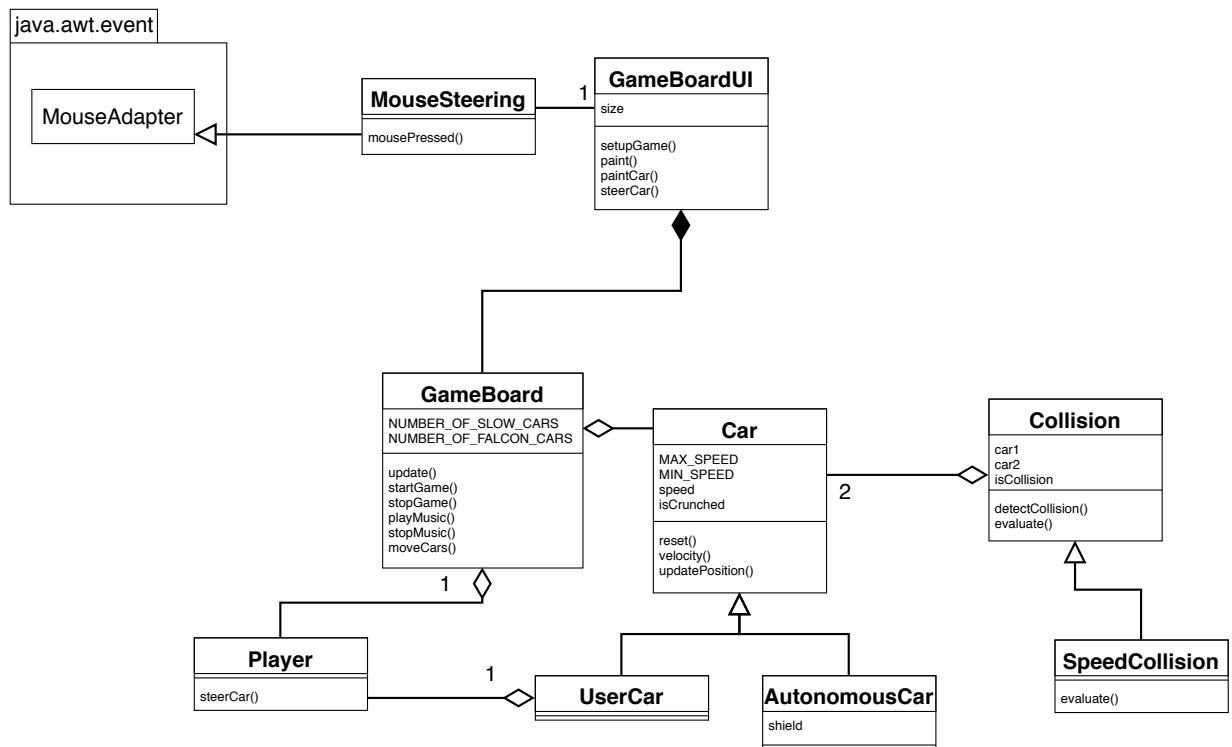## Maximilian Frühauf

12th May 2018
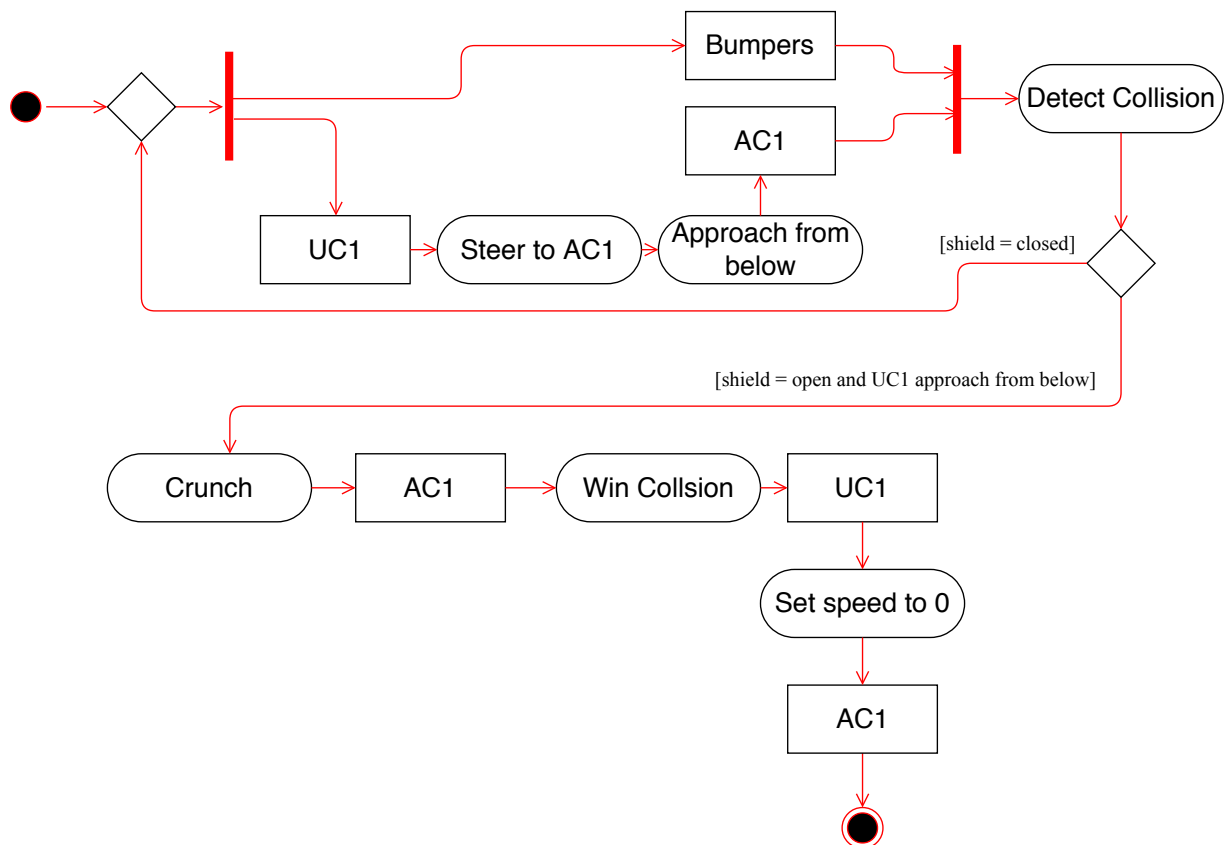
1. Dynamic behavior:

2. Taxonomy for the car shield use case:

java.awt.event

MouseAdapter

MouseSteering

mousePressed()

1

GameBoardUI

size

setupGame()
paint()
paintCar()
steerCar()

GameBoard

NUMBER_OF_SLOW_CARS
NUMBER_OF_FALCON_CARS

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

Car

MAX_SPEED
MIN_SPEED
speed
isCrunched

reset()
velocity()
updatePosition()

2

Collision

car1
car2
isCollision

detectCollision()
evaluate()

SpeedCollision

evaluate()

1

Player

steerCar()

1

UserCar

AutonomousCar

shield

3. Win Collision activity diagram:

Bumpers

AC1

Detect Collision

UC1 → Steer to AC1 → Approach from below

[shield = closed]

[shield = open and UC1 approach from below]

Crunch → AC1 → Win Collsion → UC1

Set speed to 0

AC1

4. Collision detection algorithm communication diagram:

:AudioPlayer

\+ field: Type

\+ method(): Type

:GameBoard

\+ field: Type

\+ method(): Type

1.1: [collision.isCollision] evaluate()

:ShieldCollision

\+ field: Type

\+ method(): Type

2.1: playBangSound()

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

AC1:AutonomousCar

\+ field: Type

\+ method(): Type

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

PC1:PlayerCar

\+ field: Type

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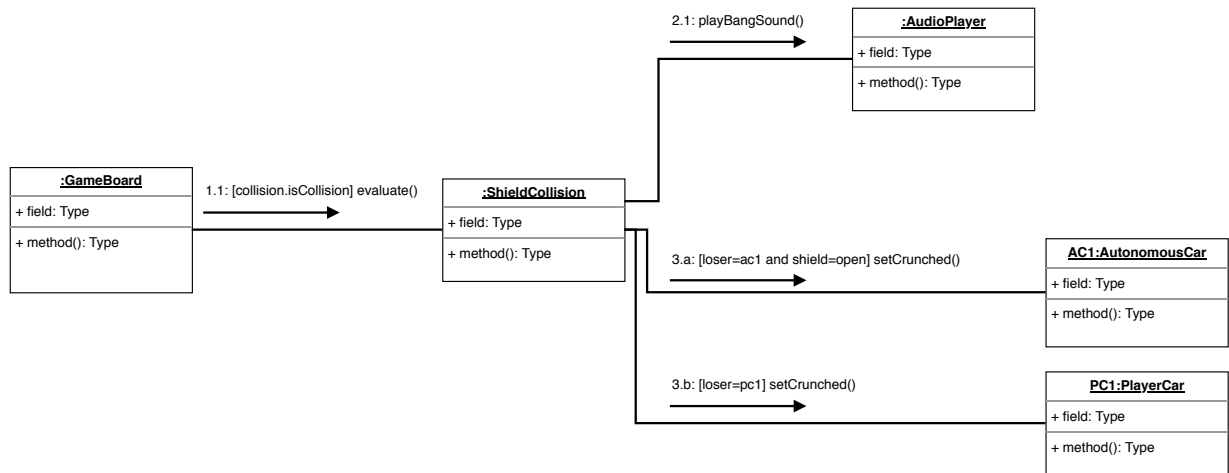
5. Coupling measures the dependency between different subsystems inside a project. High coupling therefore implies that changes to one subsystem will have a large impact on another subsystem. Low coupling on the other hand means that, a change in one subsystem does not affect others.

   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.