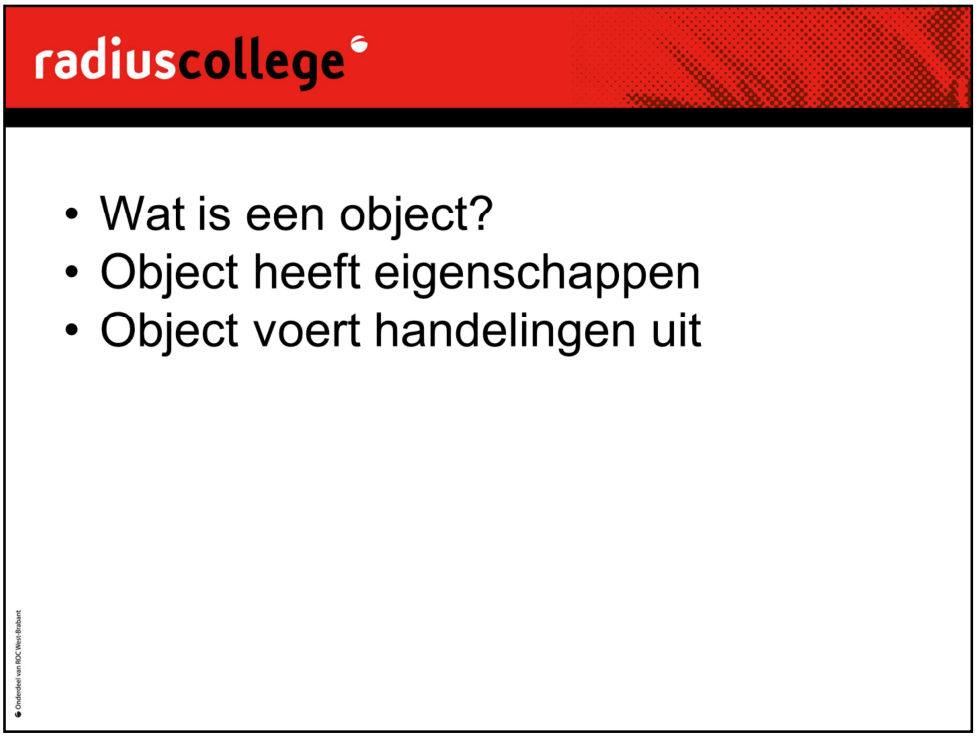
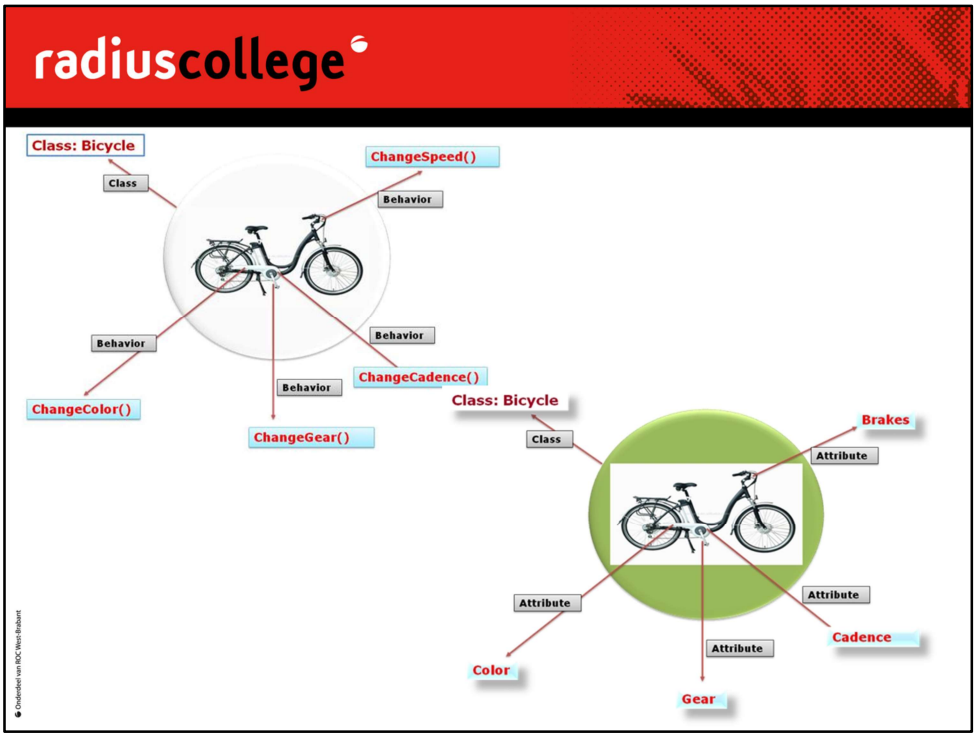
radiuscollege

PR FIFA - HC 3

Back to OOP

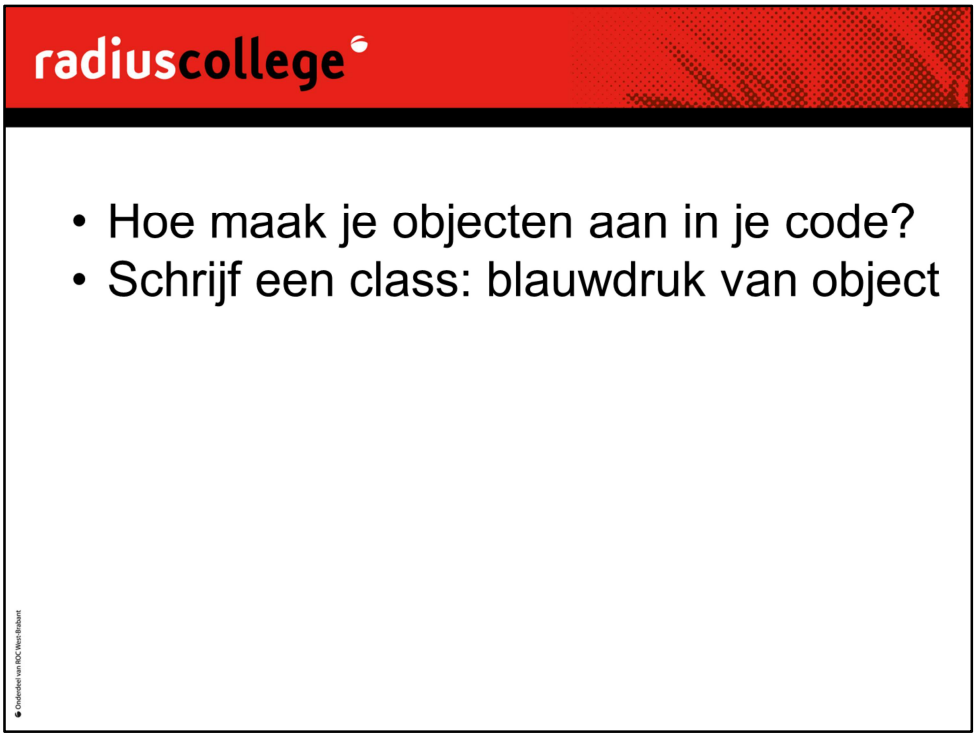


We gaan weer even terug naar OOP (object oriented programming). Een object kan van alles zijn, bijvoorbeeld een voetbalteam, een speler of een schoen. Objecten hebben altijd bepaalde eigenschappen. Zo heeft een team de eigenschap dat het bestaat uit spelers. Een speler heeft zelf ook eigenschappen (denk aan aantal goals, assists, etc.). Objecten kunnen ook handelingen uitvoeren (functies). Een speler kan bijvoorbeeld een doelpunt maken.



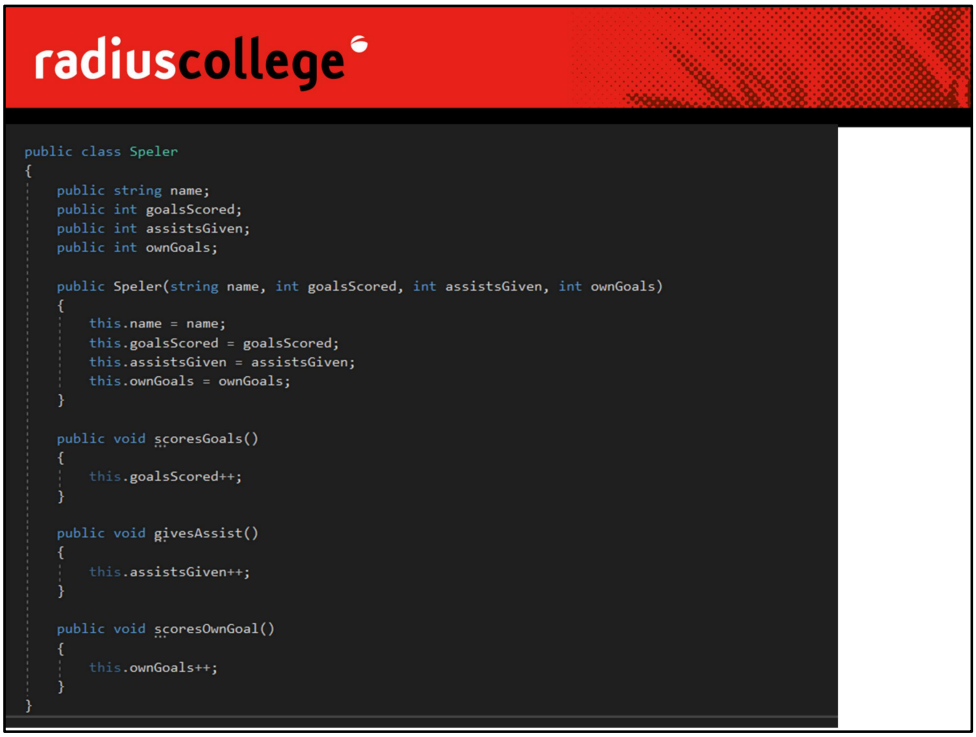
Als voorbeeld een fiets object. Linksboven fiets met functies, rechtsonder de eigenschappen.

Kan de klas nog meer voorbeelden bedenken?



Om objecten in je code te maken, schrijf je in je code een Class. Dit is als het ware een stempel: de algemene vorm verandert niet, maar je kan binnen die stempel wel eigenschappen aanpassen. Denk hierbij bijvoorbeeld weer aan de Speler. Aan het begin van het seizoen heeft niemand doelpunten of assists op zijn naam staan. Tijdens het seizoen kan dit per speler worden aangepast, maar de class zelf verandert niet: alleen het object verandert.

radiuscollege

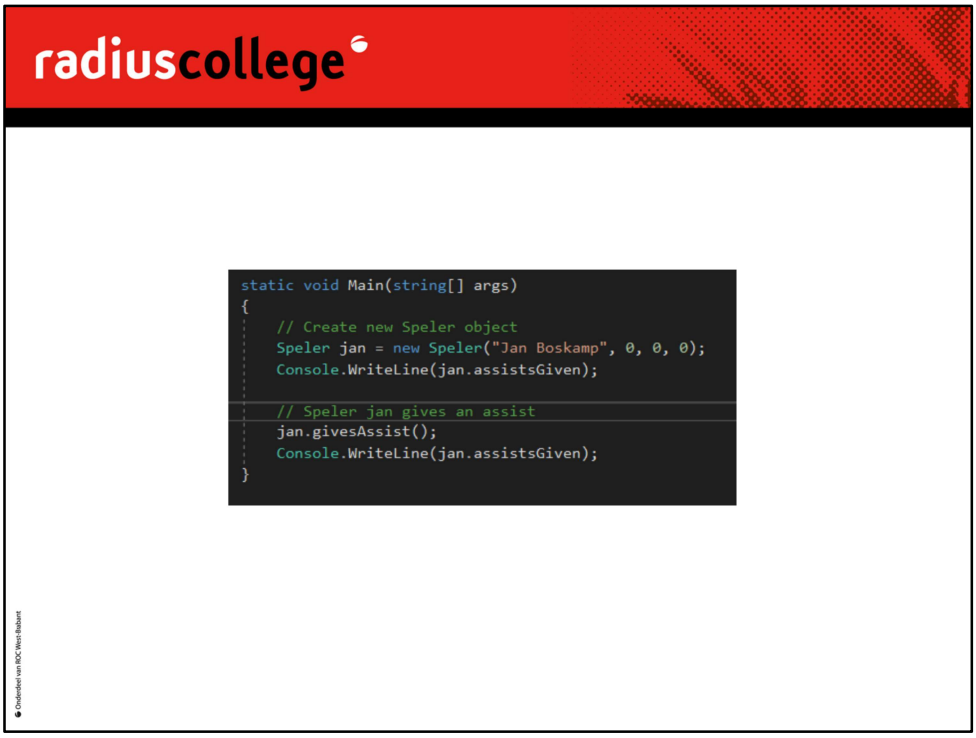


```
public class Speler
{
    public string name;
    public int goalsScored;
    public int assistsGiven;
    public int ownGoals;

    public Speler(string name, int goalsScored, int assistsGiven, int ownGoals)
    {
        this.name = name;
        this.goalsScored = goalsScored;
        this.assistsGiven = assistsGiven;
        this.ownGoals = ownGoals;
    }

    public void scoresGoals()
    {
        this.goalsScored++;
    }

    public void givesAssist()
    {
        this.assistsGiven++;
    }

    public void scoresOwnGoal()
    {
        this.ownGoals++;
    }
}
```

Vraag hierbij wat hier gebeurt: wat is de constructor, wat is het keyword void, etc.

Voorbeeldje van een class. Bovenaan staan de eigenschappen die de speler heeft. Daaronder, dus “public Speler(string name, …)”, staat de constructor. Als je een nieuw Speler object aan wilt maken, roep je deze altijd aan.

Onder de constructor staan de verschillende handelingen die een Speler object uit kan voeren.

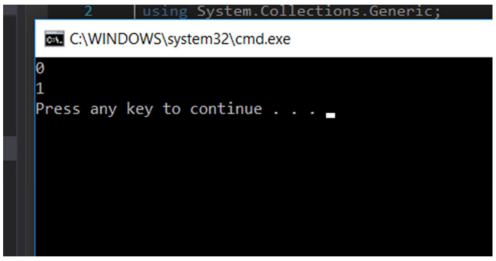


Vraag hier wat de student voor uitkomt verwacht.

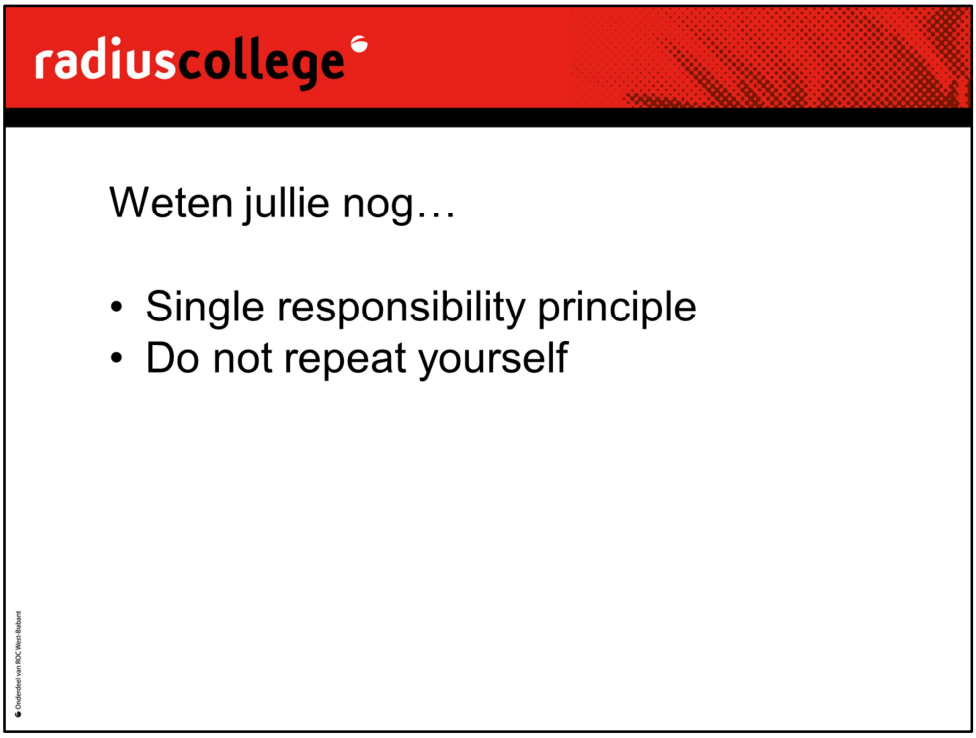
Het aanmaken van nieuwe Speler objecten gaat via de constructor zoals je hier op het voorbeeld kunt zien.

Het nieuwe object kan aangeroepen met de naam “jan”. Dit object heeft dus bij eigenschap naam, “Jan Boskamp” staan. De Speler heeft nog geen goals of assists op zijn naam staan en gelukkig ook nog geen eigen doelpunten. Hierboven geeft Speler jan een assist: als je deze code uitvoert, print je console eerst een 0, dan een 1 uit.

radiuscollege



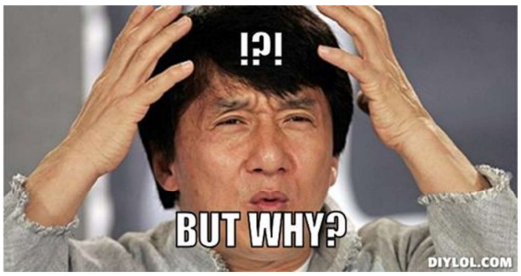
Eigenschappen, functies en classes kunnen private, public of protected zijn. Als een functie of eigenschap public is, kun je deze van buiten de class aanroepen. Als deze private is kan dat niet: alleen de class zelf kan de functie of eigenschap aanroepen.



Één class heeft maar één verantwoordelijkheid. Geef voorbeelden hiervan.

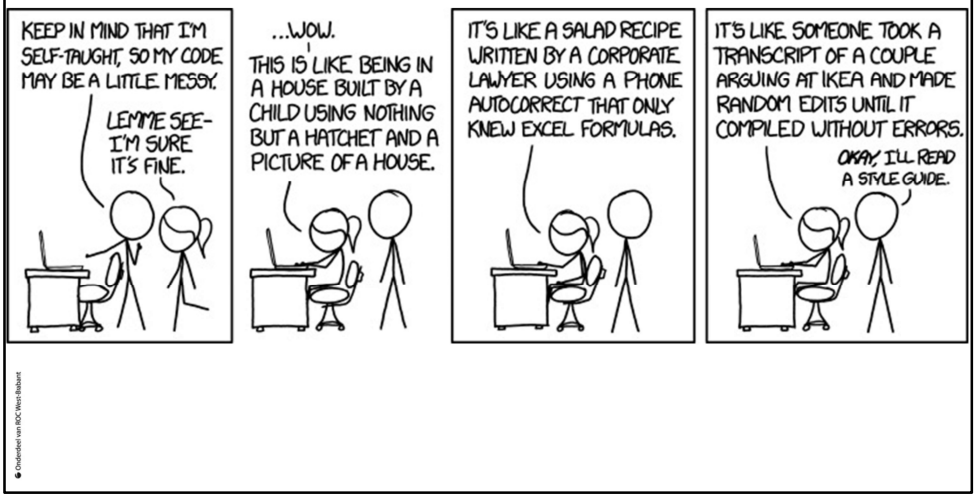
radiuscollege

- Waarom zouden we OOP gebruiken?
- Zijn er ook nadelen?



OOP voordelen: de code is goed herbruikbaar want je kan makkelijk een class uit het ene project pakken en gebruiken voor een ander project. Code kan ook makkelijk worden uitgebreid en OOP code is beter voor de leesbaarheid. OOP voldoet aan de single responsibility principle en het is heel makkelijk om jezelf niet te herhalen.

radiuscollege



Dit plaatje laat zien dat OOP code beter leesbaar is dan procedurele code.

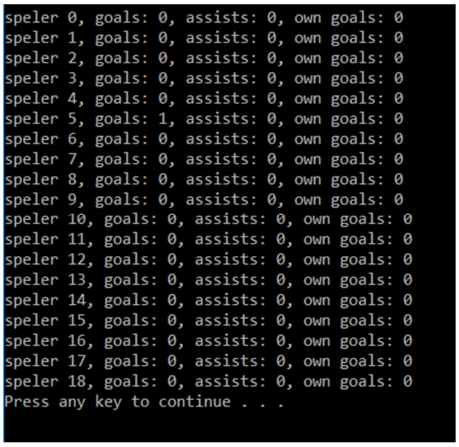
radiuscollege

```
List<Speler> team = new List<Speler>();
for(int i = 0; i < 19; i++)
{
    string name = "speler " + i.ToString();
    int goals = 0;
    int assists = 0;
    int ownGoals = 0;
    Speler temp = new Speler(name, goals, assists, ownGoals);
    team.Add(temp);
}

team[5].ScoresGoal();
```

```
foreach(Speler speler in team)
{
    Console.WriteLine(speler.ToString());
}
```


Nog een voorbeeldje. Vraag wat hier gebeurt.

radiuscollege



Output van vorige sheet