Communication and Multimedia Design

Hogeschool van Amsterdam

tt 24/25

OO puntnotatie en OO libraries

L. Benvenuti

Creating Tomorrow

Object Oriented Programming

Ontstaan in de softwarecrisis

1. Simulaties mogelijk te maken
2. Hergebruik van code mogelijk te maken
   (en dus softwareontwikkeling goedkoper te maken)

Vergelijk software met een fiets – bestaat uit onderdelen. OO Software ook.

Rollenspel!

Simulatie van een bordspel

De code bestaat uit 3 soorten objecten: **Dobbelstenen, Pionnen en CodeMonkeyIsland** spellen

```
let lorenzo = new Dobbelsteen();

let binc = new Pion('blauw'),
let kim = new Pion('rood'),
let ali = new Pion('geel'),
let mara = new Pion('groen'),

let myPionnen = [binc , kim , ali ,mara ];

let sep = new CodeMonkey(myPionnen, lorenzo );
```

CodeMonkey

```
public  pionnen: [camil, sep, jesse, kim ]
public  dobbelsteen: binc
private pionAanZet: kim
private stappen: 3
private veld: fruit
```

```
public CodeMonkey( Pion [] myPionnen,
                              Dobbelsteen myDobbelsteen) {
                this.pionnen = myPionnen;
                this.dobbelsteen = myDobbelsteen;
                this.pionAanZet = this.pionnen[0];
}

public speel(){ //bij simulatie in de les: herhaal dit
                this.pionAanZet = volgendePion met de klok mee;
                this.dobbelsteen.werp();

                this.stappen = this.dobbelsteen.getWorp();
                this.pionAanZet.zet(stappen);
                this.veld = pionAanZet.getVeld();

                if(this.veld == 'quicksand') this.pionAanZet.zet(0);
                if(this.veld == 'tree')        this.pionAanZet.zet(3);
                if(this.veld == 'vine')        this.pionAanZet.zet(6);
                if(this.veld == 'rock')        this.pionAanZet.zet(-2);
                if(this.veld == 'bottle')    pionAanZet.zet(-1);
                if(this.veld == 'fruit')       fruit();
}

private fruit(){
                let resultaat = 0;

                this.pionnen.forEach( pion => {
                                if (pion.getVeld() == 'tree') {
                                      this.resultaat = resultaat + 1;
                                }
                };

                this.pionAanZet.zet(resultaat);
}
```

Object Oriented Programming: roundup

- Alle Pionnen, Dobbelstenen hebben dezelfde code:
  1x code schrijven, maximaal hergebruik

- Ieder object heeft zijn eigen verantwoordelijkheid:
  - voor het beheer van eigen data
  - Voor het beantwoorden van “verzoeken” van andere objecten

  Dobbelsteen-objecten zijn verantwoordelijk voor het genereren van toevalsgetallen tussen 1 en 6

  Pion-objecten zijn verantwoordelijk voor het beheer van de positie op het bord

  CodeMonkeyIsand- objecten zijn verantwoordelijk voor de spelregels

Pion

```
public Pion(String myKleur) {

  }


public getVeld() {

  }


public zet(Number stappen) {

}
```

Dobbelsteen

```
public Dobbelsteen() {
    }


public werp(){

}


public getWorp(){

    }
```

Pion

```
 / Constructor
// myKleur: de kleur van de Pion
public Pion(String myKleur) {

   }

// methode getVeld() geeft informatie van het veld waarop
// de Pion staat
// return: een String met het symbool op het veld
public getVeld() {

   }


// methode zet() verplaatst de Pion op het bord
// stappen: het aantal stappen dat de Pion moet
// zetten
public zet(Number stappen) {

   }
```

Dobbelsteen

```
// Constructor
public Dobbelsteen() {
     }


// methode werp() werpt de Dobbelsteen
public werp(){

}


// methode getWorp() haalt de laatste worp van de
// Dobbelsteen op.
// return: de laatste worp van de Dobbelsteen
public getWorp(){

     }
```

CodeMonkey

```
public  pionnen: []
public  dobbelsteen:
private pionAanZet:
private stappen:
private veld:
```

```
public CodeMonkey( Pion [] myPionnen,
                                Dobbelsteen myDobbelsteen) {
                  this.pionnen = myPionnen;
                  this.dobbelsteen = myDobbelsteen;
                  this.pionAanZet = this.pionnen[0];
}

public speel(){ //bij simulatie in de les: herhaal dit
                  this.pionAanZet = volgendePion met de klok mee;
                  this.dobbelsteen.werp();

                  let stappen = this.dobbelsteen.getWorp();
                  this.pionAanZet.zet(stappen);
                  this.veld = pionAanZet.getVeld();

                  if(this.veld == 'quicksand') this.pionAanZet.zet(0);
                  if(this.veld == 'tree')      this.pionAanZet.zet(3);
                  if(this.veld == 'vine')      this.pionAanZet.zet(6);
                  if(this.veld == 'rock')      this.pionAanZet.zet(-2);
                  if(this.veld == 'bottle')     pionAanZet.zet(-1);
                  if(this.veld == 'fruit')     fruit();
}

private fruit(){
                  let resultaat = 0;

                  this.pionnen.forEach( pion => {
                                    if (pion.getVeld() == 'tree') {
                                          this.resultaat = resultaat + 1;
                                    }
                  };

                  this.pionAanZet.zet(resultaat);
}
```

CodeMonkey

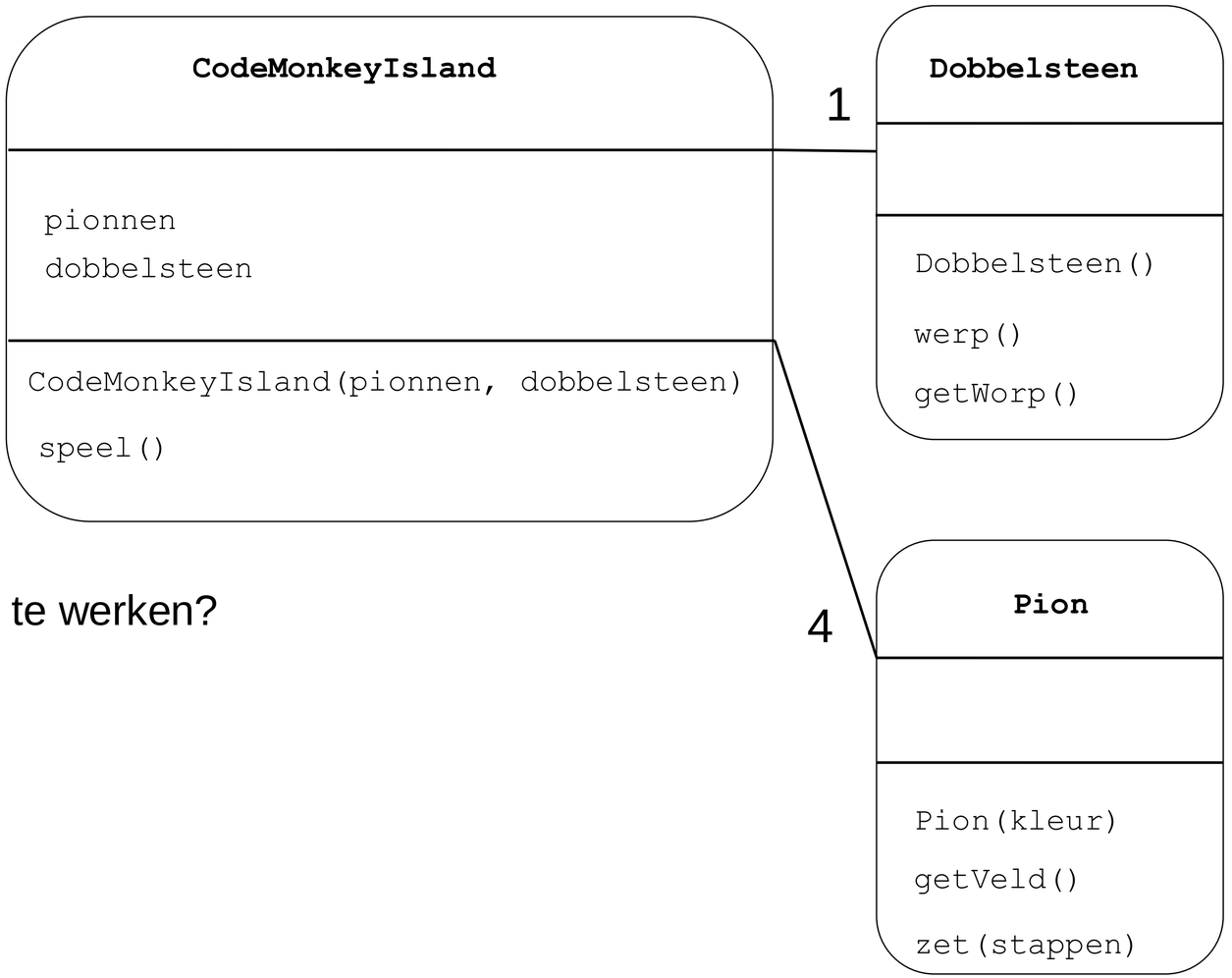
```
// pionnen: array van Pion-objecten
public pionnen:

// dobbelsteen: Dobbelsteen-object
public dobbelsteen
```

```
// Constructor
// myPionnen: array van Pion-objecten
// myDobbelsteen: Dobbelsteen-object
public CodeMonkey (Pion[] myPionnen,
                                    Dobbelsteen myDobbelsteen) {
}
```

```
// methode speel() bevat de spelregels van Code Monkey
// Island en zorgt er voor, dat het spel wordt gespeeld
public speel(){ //bij simulatie in de les: herhaal dit

}
```



Hergebruik van code

Wat moet je weten om met objecten van het type Dobbelsteen te werken?

Dobbelsteen

Representeert een zuivere, 6-zijdige dobbelsteen

Attributen

Methoden:

Dobbelsteen() – constructor, hiermee maak je een nieuwe Dobbelsten aan

werp() – simuleert het eenmaal werpen van een dobbelsteen

getWorp() – geeft de laatste worp terug
return: Number uit {1,2,3,4,5,6}

Reference Guide (als MDN, d3)
Doelgroep: andere programmeurs

Willen weten:
- welke objecten mag ik gebruiken?
- wat zijn hun verantwoordelijkheden?
- welke methoden mag ik aanroepen?
- welke parameters hebben die methoden?
- Wat geven die methoden terug?

CodeMonkeyIsland
Bewaakt de spelregels van het bordspel

Attributen
pionnen - array objecten van het type Pion
dobbelsteen - Dobbelsteen

Methoden
CodeMonkeyIsland() – constrc tor, heeft
parameters: pionnen - array objecten van het type Pion
dobbelsteen - Dobbelsteen

speel() – methode die de ‘beurt’ van volgende speler uitvoert

Pion
Beeldt de toestand van de pion af op het bord

Attributen

Methoden:
Pion() – constructor, heeft
parameter: kleur - String uit {‘rood’, ’blauw’, ‘geel’, ‘groen}

getVeld() – geeft informatie terug over de toestand van de Pion
return: String

zet() – verandert de toestand van de Pion, heeft
parameter: stappen - Number.

Dobbelsteen
Representeert een zuivere 6-zijdige dobbelsteen

Attributen

Methoden:
Dobbelsteenr() - constructor

werp() – simuleert het eenmaal werpen van een dobbelsteen

getWorp() — geeft de laatste worp terug
return: Number

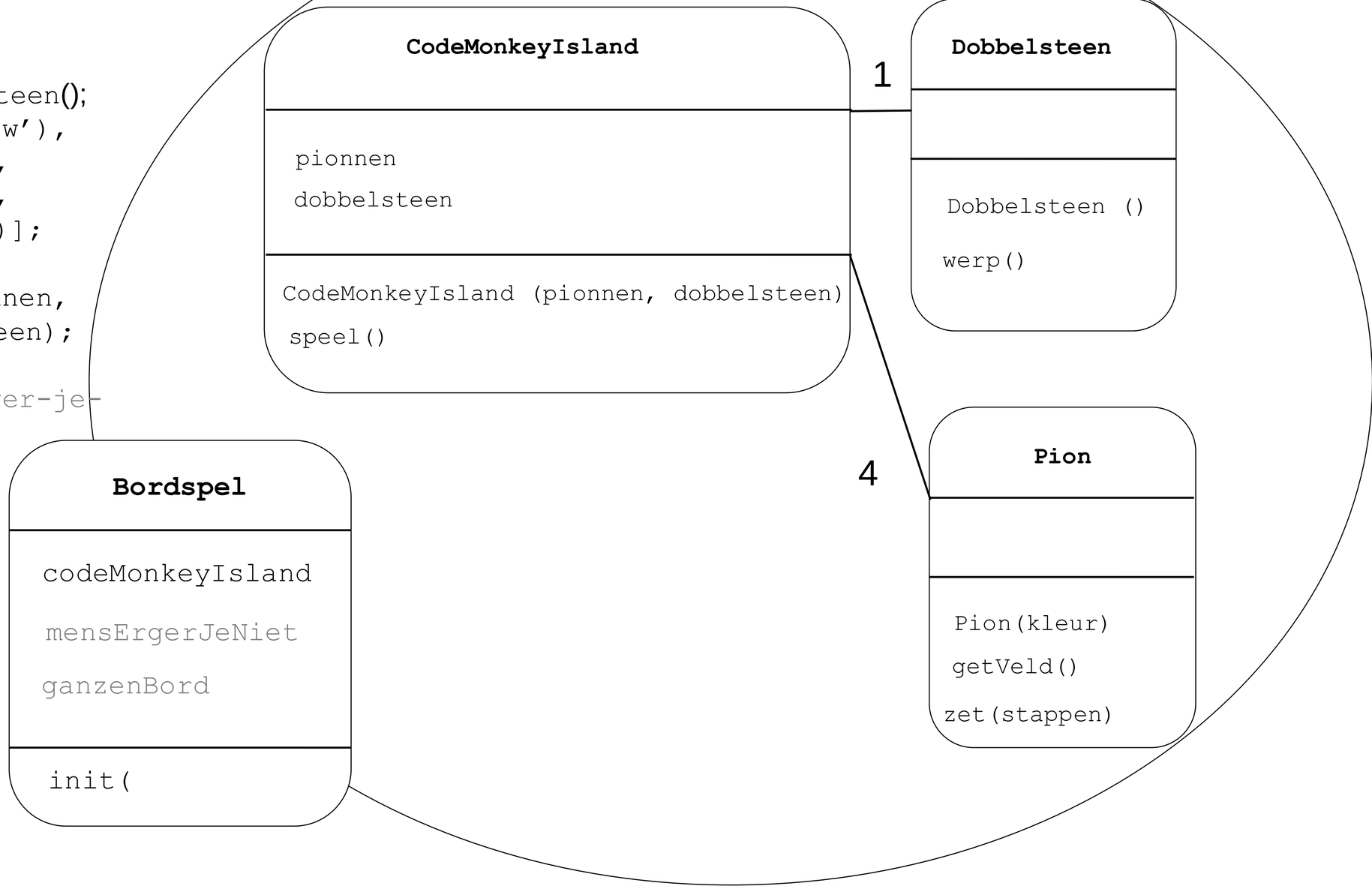
Library bordspel

```
Bordspel() {
 let myDobbelsteen = new Dobbelsteen();
 let myPionnen = [new Pion('blauw'),
                  new Pion('rood'),
                  new Pion('geel'),
                  new Pion('groen')];
 this.codeMonkeyIsland =
     new CodeMonkeyIsland(myPionnen,
                          myDobbelsteen);

enz. voor ganzenbord en mens-erger-je-
niet


getCodeMonkey() {
 return this.codeMonkeyIsland;

enz. ook voor ganzenbord en
mens-erger-je-niet
}
```



Documentatie

bordspel
Library waarmee verschillende spellen kunnen worden gesimuleerd: (bordspellen, kaartspellen, gokspellen enz.)

Bevat de API van bordspellen

codeMonkeyIsland,
mensErgerJeNiet,
ganzebord

Kaartspel (Bevat de API van kaartspelen)
poker,
pesten,
klaverjassen

CodeMonkeyIsland
Bewaakt de spelregels van het bordspel

Attributen
pionnen - array objecten van het type Pion
dobbelsteen - Dobbelsteen

Methoden
CodeMonkeyIsland() – constructor, heeft
parameters: pionnen, array objecten van het type Pion
dobbelsteen - Dobbelsteen

speel() – methode die de ‘beurt’ van volgende speler uitvoert

Dobbelsteen
Representeert een zuivere, 6 zijdige dobbelsteen

Attributen

Methoden:
Dobbelsteen – constructor

werp() – simuleert het eenmaal werpen van een dobbelsteen

getWorp() – geeft de laatste worp terug
return: Number

Morgen, dinsdag 22 oktober

13:40 15:20 TTH02B21

Researching datasets and sketching ideas

Brianne de Deugd en Yeliz Unlü, 4de jaars CMD komen hun werk laten zien

Vragen?