

Herhaling

- Nood aan methodes
 - Waarom?
 - Code overzichtelijker maken
 - Wat?
 - Opeenvolging van instructies die logischerwijze samenhangen
 - Opsplitsen in logische eenheden
 - ->stapsgewijs verfijnen
- Wereldniveau-methodes
 - Methodes waarin objecten met elkaar interageren

Herhaling

- Methodes met parameters
 - Wanneer methodes gelijkaardig zijn op 1 object, nummer, string,...
 - Minder typewerk
- Klasse
 - Voorbeelden: Mensen, Dieren, Gebouwen,...
- Vehicle-property
 - Objecten een elkaar koppelen
 - Bijv. instrument aan persoon koppelen
 - ...

Klasseniveaumethodes

- Klasse mens, dier, ...
 - Kunnen bepaalde opdrachten uitvoeren
 - Move, turn, resize, roll
- Nieuwe, typische methodes toekennen aan een bepaalde klasse.
 - Bijv. ijsschaatster een schaatsroutine laten uitvoeren.
 - Voorbeelden...
- Nieuwe acties definiëren voor object alleen
 - = Klasseniveaumethode

Overerving

- Methodes toekennen aan object
- In andere wereld dit object toevoegen
- Object kent vroeger gedefinieerde methodes
- => Latere instanties van het object zijn in staat om methodes uit te voeren
- =Overerving
- Voorbeeld : schaatster leert schaatsen. Nieuwe wereld, nieuwe schaatster=>schaatster kan ook schaatsen

Uitgewerkt voorbeeld

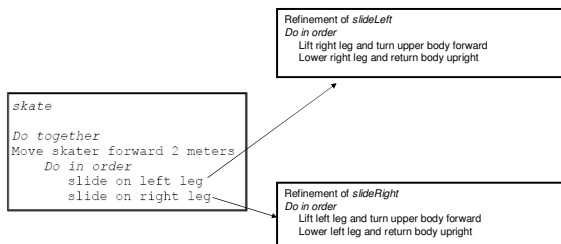
- Schaatsster op ijs, weet niet wat schaatsen is
- Schaatsen= opeenvolging van bewegingen
- Schaatsen
 - Vooruit gaan
 - Met linkerbeen glijden
 - Met rechterbeen glijden
 - Bovenlichaam in evenwicht houden

Hoe schaatsmethode maken?



- Nieuwe methode moet geassocieerd worden met de ijsschaatser
- Schrijf de nieuwe methode
- Klik in object tree 'iceskater' aan.
- In tabblad methode : create new method

Methode: schaatsen



Uitgewerkt voorbeeld

- Schaatsen is een actie specifiek voor de ijsschaatster.
- Er zijn geen andere objecten voor nodig
- Dit geldt ook voor de methodes slideLeft en slideRight
- Dus klasseniveau-methodes

slideLeft

iceSkater.slideLeft No parameters create new parameter

No variables create new variable

Do in order

- Do together**
 - iceSkater.rightLeg **turn forward** 0.1 revolutions **duration = 0.5 seconds** **more...**
 - iceSkater.upperBody **move forward** 0.01 meters **duration = 0.5 seconds** **more...**
- Wait 0.5 seconds**
- Do together**
 - iceSkater.rightLeg **turn backward** 0.1 revolutions **duration = 0.5 seconds** **more...**
 - iceSkater.upperBody **move backward** 0.01 meters **duration = 0.5 seconds** **more...**

slideRight

iceSkater.slideRight No parameters create new parameter

No variables create new variable

Do in order

- Do together**
 - iceSkater.leftLeg **turn forward** 0.1 revolutions **duration = 0.5 seconds** **more...**
 - iceSkater.upperBody **move forward** 0.01 meters **duration = 0.5 seconds** **more...**
- Wait 0.5 seconds**
- Do together**
 - iceSkater.leftLeg **turn backward** 0.1 revolutions **duration = 0.5 seconds** **more...**
 - iceSkater.upperBody **move backward** 0.01 meters **duration = 0.5 seconds** **more...**

Schaatsen

- Samenvoegen slideLeft en slideRight
- Vooruit gaan

iceSkater.schaats No parameters create new parameter

No variables create new variable

Do together

- iceSkater **move forward** 2 meters **duration = seconds** **more...**

Do in order

- iceSkater.slideLeft
- iceSkater.slideRight

Tijdsduur?

- Som van de tijdsduur van slideLeft en slideRight
- Let op : seconden in do together slecht eenmaal tellen
- Duration = 3 seconden

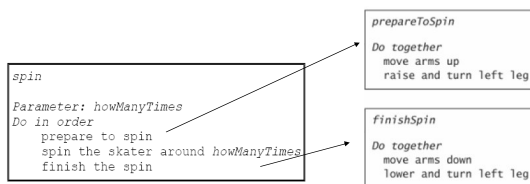
Testen methode

- Methode aanroepen
- Uittesten

Methode met parameter

- ijschaatster een spin laten uitvoeren
- Opsplitsen
 - Voorbereiden
 - Draaien
 - Einde
- Wat is de parameter?
 - Aantal spins
 - = number

Uitgewerkt voorbeeld



The screenshot shows a programming environment with two methods defined for an `iceSkater` object.

iceSkater.voorbereidenSpin No parameters

No variables

Do together

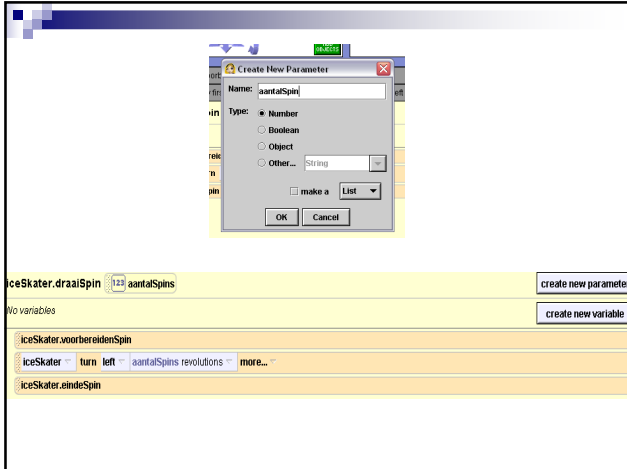
- iceSkater.upperBody.chest.leftShoulder.arm = turn backward = 0.5 revolutions = more...
- iceSkater.upperBody.chest.rightShoulder.arm = turn backward = 0.5 revolutions = more...
- iceSkater.leftLeg = turn left = 0.2 revolutions = more...
- iceSkater.leftLeg = turn backward = 0.25 revolutions = more...

iceSkater.eindeSpin No parameters

No variables

Do together

- iceSkater.upperBody.chest.leftShoulder.arm = turn forward = 0.5 revolutions = more...
- iceSkater.upperBody.chest.rightShoulder.arm = turn forward = 0.5 revolutions = more...
- iceSkater.leftLeg = turn right = 0.2 revolutions = more...
- iceSkater.leftLeg = turn forward = 0.25 revolutions = more...



Uittesten

- De ijsschaatster
 - ☐ Schaatst vooruit
 - ☐ Draait 3 spins
 - ☐ Schaatst verder vooruit
 - ☐ Draait 1 spin

Opslaan en importeren

- Object opslaan als nieuw 3D model
- Twee stappen
 - ☐ Naam veranderen!
 - Rechtsklikken : rename : CleverIceSkater
 - ☐ Opslaan
 - Rechtsklikken : save object
 - .a2c => Alice 2.0 Class
- Importeren
 - ☐ File -> Import

Voordelen overerving

- Code eenmaal schrijven, maar mogelijkheid tot hergebruik
- Projecten maken in teams.
 - ☐ Ieder maakt elk apart een object
 - ☐ Later alles samenvoegen

Tips en valkuilen

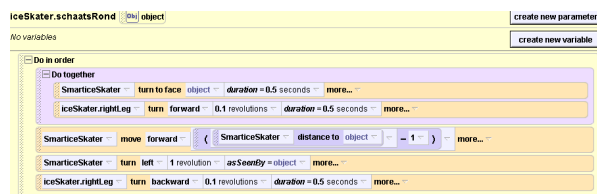
- Tip
 - Veel verschillende klasseniveau-methodes schrijven is zeer nuttig
- Valkuilen
 - Geen wereldniveau-methodes oproepen in een klasseniveau-methode

Valkuilen



- Geen andere objecten specifiek oproepen
 - Mogelijk dat ze niet bestaan in een andere wereld
 - Oplossing: Klasseniveau-methode met object-parameter

Klasseniveaumethode met parameter



Uittesten uitgewerkt voorbeeld

Individuele Oefening

- Maak een nog betere ijsschaatster.
 - achteruitSchaatsen
 - Analooq aan vooruit, maar moet nu achteruit glijden
 - springen
 - Vooruit bewegen
 - 1 benen omhoog houden
 - Omhoog springen
 - Elegant landen en been terug op grond plaatsen
- Bewaar het nieuwe object als heelSlimmeSchaatster

Individuele oefening

- Voeg in wereld toe:
 - Lake
 - HeelSlimmeSchaatster
 - Duck
 - Penguin
- Test je gemaakte methodes
- Laat je figuur rond de eend en de pinguin schaatsen