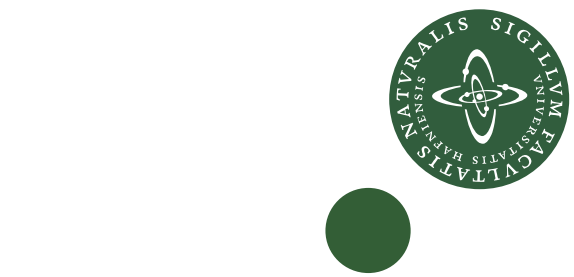
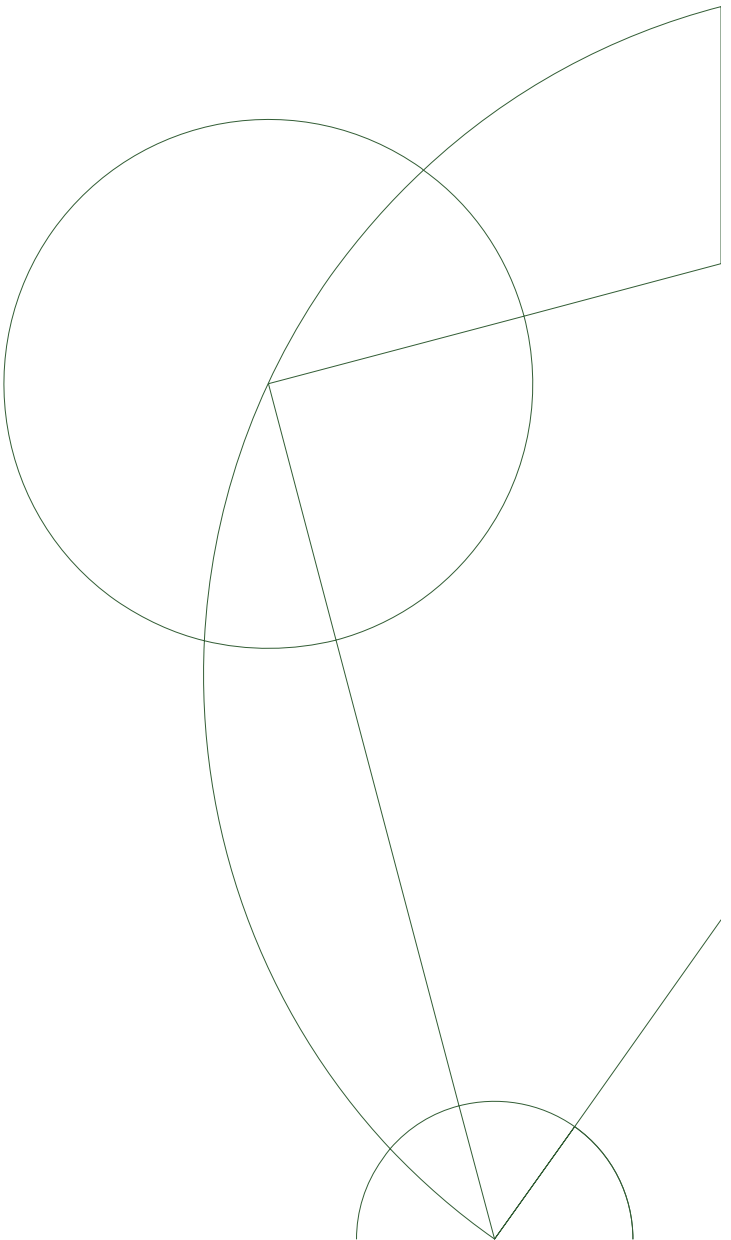


Programmering og Problemløsning

Aflevering 10g - The Animal Race



Adam Ingwersen,
Aske Fjellerup,
Peter Friborg

Datalogisk Institut
Københavns Universitet

January 9, 2017

1 Introduktion

I denne opgave stiftes der bekendskab med nedarvning i klasser og overskrivning af metoder.

2 10g.0

Der startes med at opføres et UML-diagram, der giver et overblik over følgende opgave. Diagrammet skal have en `base class`, der tager to argumenter som `constructor`: `Animal(weight, maxspeed`. Der skal være to `derived` klasser: `Herbivore(weight, maxspeed)` & `Carnivore(weight, maxspeed)`.
De skal begge have to forskellige afarter af `Hunger()`, der beskriver hvor meget mad der skal spises iforhold til deres vægt.

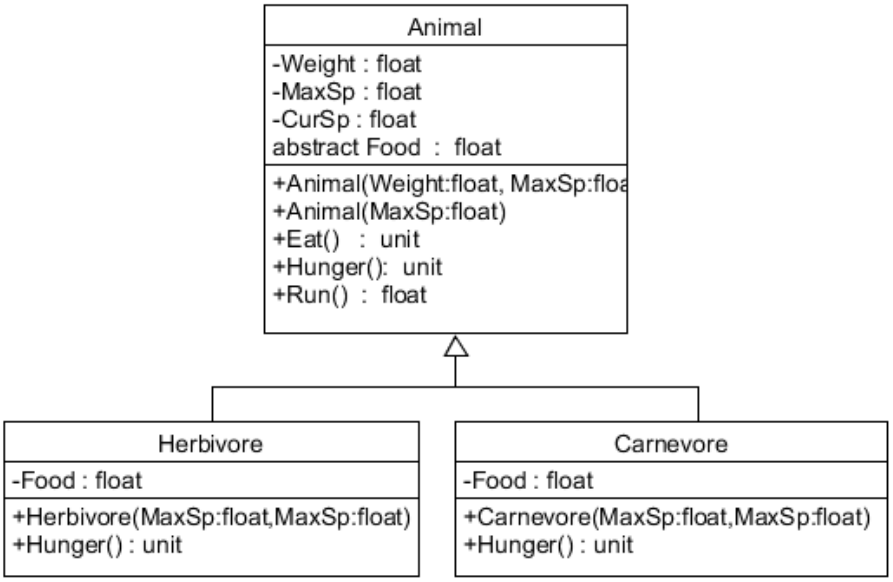


Figure 1: UML-diagram over Animal og dens børn: Herbivore og Carnivore

2.1 Animal

Der laves her en klasse udfra ovenstående `baseclass`: `Animal`.
Der startes med at defineres alle attributer i klassen animal: this.Food skal gøres overridable, da det skal kunne ændres hvor meget hvad hvert dyr skal spise efter om det er et kød eller planteædende dyr.

```
let mutable speed = 0.0
let mutable mad = 0.0
member this.weight = weight
member this.maxSp  = maxSp
member this.curSp  = speed
abstract member food: float
default this.food = mad
```

Figure 2: Attributer i Animal

Der skal også laves metoder tilsvarende til UML-diagrammet. Disse skal den skal kunne arbejde med attributterne.
Den ene metode, `Hunger()`, skal laves som en `abstract member`, da den skal overskrives i underklasserne.
Koden vil være som følger:

```
abstract member Hunger: unit-> unit
  mad  <- (weight/2.0)

member this.Eat food =
  speed <- ((food/this.food)*this.maxSp)

member this.Run =
  let pro  = RNG.Next (1,101)
  let temp = (( float(pro) ) / 100.0 )*this.food
  this.Eat temp
  printfn "%A procent" pro
  printfn "der spises %A kg og l bes %A timer" temp (
      int(10000.0/this.curSp))

new (maxSp) = Animal((float(RNG.Next(70,301))) , maxSp)
```

Figure 3: Metoder i Animal

Det ses at metoden `Hunger()` er en abstrakt metode, og den kan altså overskrives i `Carnivore` og `Herbivore`.

2.2 Carnivore & Herbivore

Disse to underklasser skal arve alle ikke private `attributes` og `methods`. Dette skrives i koden:

```
inherit Animal(Weight,MaxSp)
```

Figure 4: Nedarvning fra Animal

Nedarvningen kan dog ikke ændre på de mutable værdier der findes i `baseclass`, så derfor må der laves nogle værdier i begge tilfælde af `derived class`.

```
let mutable mad = 0.0
let temp = weight
```

Figure 5: Tilføjning af værdier til underklasserne

Der skal også overskrives en `attribute` og en `method`, og dette gøres udfra følgende:

```
override this.food = mad
override this.Hunger() =
  mad <- (temp*0.4)
```

Figure 6: Her sker vores overskrivninger.

Ovenstående eksempel af overskrivning er fra `Carnivore`, det ses at den skal spise 8% af sin vægt. Hvor `Herbivore`, skal spise 40%.

2.3 Test af program

Programmet testes ved at der laves 3 opjecter (cheetah, antelope og Wilderbeast) som kører metoden `Run`. Dette sker 3 gange hvor efter dyret med den bedste tid vinder. `Run` er en metode der ud fra et tilfældigt procent vil køre de 2 metoder der fra opløget og derefter beregn og printer hvor lang tid det vil tage dyret at rejse 10km.

Første gennemløb
`Cheetah`: 89 procent, der spises 3.56 kg og løbes 98 timer
`Antelope`: 94 procent, der spises 18.8 kg og løbes 111 timer
`Wilderbeast`: 92 procent, der spises 73.6 kg og løbes 135 timer

Andet gennemløb
`Cheetah`: 38 procent, der spises 1.52 kg og løbes 230 timer
`Antelope`: 77 procent, der spises 15.4 kg og løbes 136 timer
`Wilderbeast`: 5 procent, der spises 4.0 kg og løbes 2500 timer

Tredje gennemløb
`Cheetah`: 66 procent, der spises 2.64 kg og løbes 132 timer
`Antelope`: 88 procent, der spises 17.6 kg og løbes 119 timer
`Wilderbeast`: 68 procent, der spises 54.4 kg og løbes 183 timer

Resultatet er at dyret med det bedste gennemsnit er Antelope.