

Obligatorisk Opgave C
Deadline: Oct 27, 2016



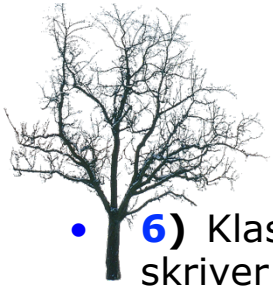
Opgave C



- Denne opgave handler om simulering af træer, i fortsættelse af tidligere opgaver. Men nu skal der ud over klassen `Tree` laves to subklasser `Ash` og `Beech`, svarende til asketræer og bøgetræer. Ideen er at asketræer og bøgetræer har forskellige vækstforløb, som bedst beskrives med forskellige `growOneYear()` metoder.
- **1)** Lav en klasse `Tree` med to public felter:
 `age` af type `int`, som er træets alder; og
 `height` af type `double`, som er træets højde.
- **2)** Lav en metode `void growOneYear()`, som forøger `age` med 1 og ellers ikke gør noget.
- **3)** Lav en metode `void show()`, som udskriver træets alder og højde, fx i dette format:

alder = 17 år og højde = 3.66 meter.
- **4)** Lav en klasse `Ash`, som en subklasse af `Tree`. Klassen skal ikke erklære nogen felter selv, den arver jo `age` og `height` fra superklassen `Tree`. Klassen skal have en konstruktør `Ash()` uden parametre som sætter `age` til 1 og `height` til 1.0.
- **5)** Klasse `Ash` skal have en ny version af metoden `public void growOneYear()`. Dels skal denne metode udføre `super.growOneYear()`, som kalder superklassens `growOneYear`-metode der tæller `age` op med 1. Dels skal den lægge 20 % til højden ved at gange `height` med 1.20, forudsat at *højden* er mindre end 15.

Opgave C



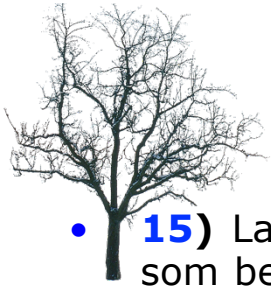
- **6)** Klasse **Ash** skal desuden have en ny version af metoden `public void show()`, som skriver "**Ask:**" og dernæst udfører `super.show()` for at kalde superklassens `show()`.
- **7)** Lav en klasse **Beech** som en subklasse af **Tree**. Klassen skal have en konstruktør `Beech()` uden parametre, som sætter age til 1 og height til 0.5.
- Klasse **Beech** skal også have en ny version af metoden `public void growOneYear()`. Dels skal denne metode kalde `super.growOneYear()`, dels skal den lægge 6 % til højden, forudsat at *alderen* er mindre end 65. (Så i denne model er bøgetræers vækst begrænset af deres alder, mens asketræers vækst er begrænset af deres højde. I virkeligheden er det lidt mere indviklet.)
- Klasse **Beech** skal desuden have en ny version af metoden `public void show()`, som skriver "**Bøg:**" og dernæst kalder superklassens `show()`.
- **8)** Lav en klasse **Forest** med en arrayliste af **Tree**-objekter. Lav en metode `initialize()` der kommer to asketræer og to bøgetræer af forskellig alder og højde i arraylisten.
- **9)** Lav en metode `public void growOneYear()` i klasse **Forest**, der gennemløber arraylisten og kalder `growOneYear` på hvert træ. Lav en metode `public void show()` i klasse **Forest** der gennemløber arraylisten og kalder `show()` på hvert træ.
- **10)** Lav en metode `public void growManyYears(int n)`, der simulerer at skoven vokser i n år, dvs. n gange kalder `growOneYear()` og `show()`.

Opgave C



- For at få en mere realistisk simulering af et træs vækst er man nødt til at vide om træet står i nærheden af andre (høje) træer der skygger; så vil træet jo vokse langsommere.
- For at lave sådan en simulering kan man repræsentere skoven som et to-dimensionalt array `Tree[][] trees` af pladser, hvor hver plads `trees[i][j]` kan indeholde et `Tree` eller ingenting (null). Træet på plads `(i, j)` har så op til fire nabotræer, på pladserne `(i-1, j)` og `(i+1, j)` og `(i, j-1)` og `(i, j+1)`.
- **11)** Opret et nyt BlueJ-projekt til denne nye del af opgaven. Du kan genbruge næsten alt fra dine `Tree`, `Ash` og `Birch`- klasser, men klassen `Forest` udskiftes med `NewForest`.
- **12)** Lav en klasse `NewForest` som har et felt `trees` af type `Tree[][]` til at indeholde alle skovens træer, og et felt `n` der angiver hvor stor skoven er på hver led. Klassen skal have en konstruktør `NewForest()` som sætter `n` til 4 og opretter arrayet `trees` med størrelse `4×4`. Konstruktøren skal også oprette mindst 8 aske- og birke-træer og sætte dem på forskellige pladser i "skoven".
- **13)** Lav en metode `Tree getTree(int i, int j)` i `NewForest`, der returnerer træet på plads `(i, j)` hvis `i` og `j` er inden for skoven, og ellers returnerer null.
- **14)** Lav en metode `double getHeight()` i `Tree`, der returnerer træets højde.

Opgave C



- **15)** Lav en metode `double shadow(int i, int j, NewForest forest)` i klasse `Tree`, som beregner hvor meget skygge træet på plads (i,j) i skoven forest er udsat for, som et tal mellem 0.0 og 1.0 (hvor 0.0 betyder ingen skygge og 1.0 betyder komplet skygge).
 - [Hint: Du kan beregne skyggen som $k/10.0$, hvor k er antallet af nabotræer der er højere end det givne træ. (Vi dividerer med 10.0 i stedet for 4.0 for selv hvis alle fire nabotræer er højere, er der jo ikke komplet skygge). Metoden kan evt benytte kald af formen `forest.getTree(..., ...)` for at finde træets naboer.]
 - Hvis træet står på plads (i,j) så har det op til fire nabotræer der står nord for træet (over), eller syd (under), eller vest (venstre), eller øst (højre). De fire muligheder svarer til pladserne (i - 1, j) og (i+1, j) og (i, j-1) og (i, j+1). Men læg mærke til at dels kan det være at der ikke er noget træ (men `null`) ved siden af, dels står nogle af træerne i randen af skoven og har derfor ikke fire naboer. Hvis for eksempel $i = 0$ så står træet i øverste række og så er der ikke nogen nabo over træet. Metoden skal altså tjekke for null.
- **16)** Lav en metode `void growOneYear(int i, int j, NewForest forest)` i klasse `Tree`, som modificerer et træs højdetilvækst ud fra hvor meget skygge det får. Parametrene i og j angiver træets position, og forest er skoven det står i. Metoden kan for eksempel aflæse træets aktuelle højde, kalde træets eksisterende `growOneYear`-metode, udregne højdetilvæksten, udregne skyggefaktoren, og så reducere højdetilvæksten ved at trække skyggefaktor gange højdetilvækst fra højden.
- **17)** Lav en metode `void growOneYear()` i `NewForest`, der for alle træer `tree = trees[i][j]` i skoven: kalder `tree.growOneYear(i,j,this)`, men selvfølgelig kun når der er et træ på plads (i, j), ikke når der står null.