# Danmarks Tekniske Universitet


Lukas Leindals, s183920
Gustav Gamst Larsen, s180820

---

# Afleveringsopgave 4

---

## 02101 Indledende Programmering

| Opgave | Primært kode ansvar | Primært rapport ansvar |
|---|---|---|
| 1 | s180820 | s183920 |
| 2 | s180820 | s180820 |
| 3 | s183920 | s183920 |

Tabel 1: Alle opgaver er hovedsageligt lavet i samarbejde og tabellen angiver derfor hvem der har haft ansvaret for udformningen af opgaven.

15. November 2020

# Opgave 1

I opgave 1 skal man få et input via terminalen til at printe antallet af tegn i inputtet. Koden tager inputtet som flere elementer adsklit med mellemrum i terminalen og sammensætter alle argumenterne der indtages til en samlet string. Når de er samlet til en string, fjernes mellemrum da det ikke tæller som tegn og til sidst får vi systemet til at print put længden af den samlede string.

Et forsøg:

```
Gustavs-Air:src Ginger$ java Letters.java abs hg99!
8
```

Der er dog tegn som terminalen ikke tager imod. Hvis vi benytter parenteser leder terminalen efter en funktion og derfor giver en fejl besked.

```
Gustavs-Air:src Ginger$ java Letters.jave abs( 10 hi
bash: syntax error near unexpected token `('
```

# Opgave 2

Formålet er at skrive et program som kan håndtere artikler og referencer. Vi starter med at skabe en klasse `Forlag` som kan håndtere et forlag. Den har et sæt private variable for navnet på forlaget og hvor det oprinder fra. En konstruktør er også lavet for at kunne oprette forlag i programmet, samt en `toString` metode til at printe forlaget. Dernæst har vi lavet klassen `Tidsskrift` som kan håndtere titel, forlag og ISSN nummer. En konstruktør er lavet til at kunne definere tidskrifter og til at senere sætte ISSN nummer og forlag for tidsskriften. `toString` metoder er lavet for at printe titlen på tidsskriften og ISSN nummeret hvis det eksistere. Til sidst lavede vi klassen `Artikel` som kan håndtere forfattere, titel, en tidskrift og referenceliste. En konstruktor er lavet for at kunne oprette en ny artikel, samt en setreferenceliste til at tilføje en referenceliste. `toString` metoder er igen implementeret for at printe artiklen på formen som ønsket i opgaven. For at teste det hele laves `ArtikelTest` med en main metode for at teste om vi kan få programmet til at skrive det hele ud på formen som ønsket i opgaven.

Et output fra Artikeltest med variabler:

```
Forlag f = new Forlag("University Press", "Denmark");
       Tidsskrift t1 = new Tidsskrift("Journal of logic", "", f);
       Tidsskrift t2 = new Tidsskrift("Brain", "", f);
       t1.setIssn("98887");
       Artikel a = new Artikel(new String[]{"A. Abe", "A. Turing"}, "A", t1);
       Artikel b = new Artikel(new String[]{"B. Bim"}, "B", t1);
       a.setReferenceliste("B");
       b.setReferenceliste("Boogalo");
```

```
Forlaget University Press, Denmark.
```

```
Tidskrifterne Journal of logic og Brain. Disse to tidskrifter kommer begge fra
    University Press. ISSN for Journal of logic: 98887. Der er ingen Issn for
    Brain.
 Flgende  to artikler:
- A. Abe & A. Turing: "A". Journal of logic
- B. Bim: "B". Journal of logic
Den  frste  af disse artikler har en refrence til den anden.
```

---

# Opgave 3

Opgave 3 handler om at implementere Conway's Game of Life. Til dette laver vi en klasse `GameOfLife`, som testes og anvendes i klassen `GameOfLifeMain`. Al funktionalitet er implementeret i `GameOfLife` klassen for at holde det hele samlet. Klassen indeholder de private felter `state` og `n`, hvor state er et $n$ gange $n$ grid af 0- og 1-taller, hvor1 angiver en levende celle. Til klassen findes tre varianter af dens konstruktør. Den ene tager imod et heltal, som bestemmer størrelsen af grid'et og initialisere værdierne af dets state tilfældigt. Den anden tager imod en bruger defineret state og sætter dette som state værdien. Den sidste tager imod et filnavn som en string og indlæser staten fra denne fil. Værdien `n` sættes efter længden af den første række. Det tjekkes desuden at alle værdier er enten 0 eller 1 for et gyldigt state, hvis der anvendes et brugerdefineret eller filbaseret state.

Til klassen tilføjes metoderne `toString()`, `setState(int x, int y, int newState)`, `liveNeighbors(int x, int y)`, `nextState()`, `drawState()`, `drawSimulation(int sleepTime)` samt den private metode `readState(String fileName)`.

Metoden `toString` sørger for, at hvis et `GameOfLife` objekt printes, vil dets state udskrives som en matrix. `setState` ændrer værdien af en celle. Her tjekkes det at værdien er enten 0 eller 1 og at den ønskede position findes i grid'et. `liveNeighbors` returnere antallet af levende naboer, ved at summe alle værdier, som ikke er cellen selv. Dette kan gøres da en død celle har værdien 0. Desuden tages der højde for, at hvis man tjekker en celle på kanten af grid'et, vil kun de omkring liggende celler, som rent faktisk er en del af grid'et tælles med. `nextState` gennemløber state matricen og gemmer en nye matrix med antallet af levende naboer for hver celle. Dette gøres inden værdierne ændres, da en ændring af den første værdi, vil påvirke antallet af naboer for den næste, hvilket ikke er det ønskede resultat. Matricen og `setState` metoden bruges derefter til at ændre alle celleværdierne efter spillets regler. `drawState` tegner den state `GameOfLife` objektet har på det givne tidspunkt, ved at tegne en prik for alle de levende celler. `drawSimulation` bruger metoderne `nextState` og `drawState` til at køre i en løkke, hvor det nuværende stadie tegnes og derefter sættes til det næste stadie. Dette gøres med `sleepTime` ms mellemrum. Inden det næste stadie sættes, gemmes dog en kopi af det nuværende stadie. Denne kopi bruges til at tjekke om det næste stadie er lig det forrige og der derved ikke vil komme til at ske nogle ændringer i fremtiden. Er disse ens breakes løkken. Det kan ske at givne stadier kører i et gentagende mønster, her er det op til brugeren selv at standse animationen, som det ses med flere af de stadier, som er givet som filer. Til er den private metode `readState` lavet til at indlæse en fil og returnere stadiet som findes i filen. Her indlæses først den første linje for at bestemme $n$ til at sætte

størrelsen af matricen og herefter indlæses de resterende $n$-1 linjer.

Koden er testet i klassen `GameOfLifeMain` med alle tre konstruktører og det er lavet, som en klasse med bruger input, hvor brugeren selv kan vælge sit test eksempel. Valget angives med et heltal mellem 1 og 6, vælges et heltal uden for dette bruges et tilfældigt initialiseret $n$ gange $n$ grid, hvor $n$ er mellem 3 og 10.