Uke I. Ingen øvelser første uke

(Nesten) alle besvarelser skal :

- dokumenteres
- plasseres i passende pakker
  - lag en katalog `pakker` og utvid stien
  - alle løsningene til ukeøvelser bør ligge i denne katalogen : de skal ofte (gjen)brukes senere

(forelesning denne uken: Abstrakt programmering + JAVA, implementasjon, datainvarianter)

Uke II. Interface/dokumentasjon **31.8-4.9**

II.1 Peker-kjeder

II.1.1

I.2 (fjern nøkler) – minn om peker-kjeder og implementer en som kan brukes i to neste oppgaver.

II.1.2

V.4 – utvid for å skrive ut & og skrive ut baklengs (forklar litt)

II.2

Lag interface `Stack`, `Queue` og plasser dem i passende `package`/katalog.

II.3 Queue (FIFO)

forklar litt om implementasjon.

II.3.1

Implementer `Queue` med peker-kjeder fra II.1. **(for Oblig.2 ?)**

II.3.2

IV.4 – dokumenter

II.4 Stack (LIFO)

III – Implementer `Stack` med peker-kjede fra II.1 og dokumenter **(for VII.1 ++)**

(forelesning denne uken: Rekursjon, invarianter)

Uke III. JAVA/Invarianter/Implementasjon/O(n) **7-11.9**

III.1 Arv

II.1 – endre og forenkle; II.ekstraoppg.

III.2 Unntak

VI.2; X.5

III.3 Invarianter

V.1 – tilpasse

III.4 O-notasjon

I.1

(forelesning denne uken: Stack, Queue, implementasjon, Sequence)

Uke IV. Rekursjon **14-18.9**

IV.1 Fibonnaci

Bruk programmet for beregning av Fibonnaci rekken og utvid det med utskriftskommandoer slik at du får en utskrift som markerer, for ethvert rekursivt kall

- argumentet
- rekursjonsdybte
- returnert resultat

Dybden kan passelig markeres med indentering av utskriften. Du kan gjerne bruke pakken easy.Stdio. For eksempel, kallet $fib(4)$ (`>javac fib` med påfølgende argument 4), burde produsere noe som følger:

```
f(4)...
    f(3)...
        f(2)...
            f(1)...
            > 1
            f(0)...
            > 1
        > 2
        f(1)...
        > 1
    > 3
    f(2)...
        f(1)...
        > 1
        f(0)...
        > 1
    > 2
> 5
```

IV.2 Avskjæring

IV.3 + generering i leksikografisk ordning **(for Oblig.1!)**

⇒ Oblig.1 går ut på onsdag-forelesning

(forelesning denne uken: Sequence, generisk sortering, QuickSort)

Uke V. NO – innlevering av Oblig.1 : 25.9 **21-25.9**

(forelesning denne uken: Trær)

Uke VI. Oblig.1 + Sequence **28.9-2.10**

VI.1 Gjennomgang av Oblig.1

VI.2

Design Interface `Position`, `Comp`, `PosSequence` (forenklet)

VI.3 Implementer

`PosSequence`.

VI.4

Generealiser rekursiv MergeSort-algoritme (gitt på forelesning) slik at den kan brukes for en vilkårlig implementasjon av `interface Comp` og `PosSequence`

VI.5

Lag to implementasjoner $\mathtt{I}_1$ og $\mathtt{I}_2$ av `Comp` for sammenlikning av heltallsobjekter slik at $\mathtt{I}_1.lt(a,b) \iff a < b$ og $\mathtt{I}_2.lt(a,b) \iff a > b$.

Lag så et testprogram som bruker algoritmen fra oppgave VI.4 (der aktuell parameter tilsv. `PosSequence` skal være implementasjon fra oppgave VI.3) til å sortere sekvenser av heltall i stigende, resp. avtakende rekkefølge.

(forelesning denne uken: slutt av Trær, PriorityQueue, Heap)

Uke VII. Trær **5-9.10**

VII.1 Rekursjonsstabel

Gjør om rekursiv MergeSort fra oppgave VI.4 til en iterasjon ved å bruke passende stabler. Bruk implementasjon av `Stack` fra II.4 – men lag passende 'adapter'-klasser som tar seg av typekonvertering. Implementer din løsning og kjør den på noen testeksempler. **(for Oblig.3)**

VII.2 Trær

Implementer `interface BinTree` med `BNode`. **(for VIII.1)**

VII.3 En enkel rekursiv tre-morsomhet (C-5.2)

VII.4 ta med C-5.8

Mer tree-programmering kommer med `Heap`, `BST`, ...

Går ut...

Implementer generisk tre-traversering som kan brukes for både DFS og BFS avhengig av om (en av) parameter(ne) instansieres med implementasjon av `Stack` eller `Queue`.

⇒ Oblig.2 går ut på mandag...

(forelesning denne uken: Dictionary, BST)

Uke VIII. PQueue **12-16.10**

VIII.1 Heap

Implementer Heap ved å bruke `BinTree interface`. Bruk implementasjon av `BinTree` fra VII.2. Heap skal være parametrisert av `Comp` som oppgis ved opprettelse av nye Heap-objekter.

VIII.2

Design `interface PrQueue` og plasser den i en passende `package`. Implementer `interface PrQueue` med `Heap` fra oppgave VIII.1. Implementasjon skal være parametrisert med `Comp` som oppgis ved opprettelse av nye objekter.

VIII.3

Implementer en generisk pritritetskø-sortering ved å bruke en parameter av type `PQueue`. Lag et test program der parameter instansieres med `Heap`-implementasjon fra forrige oppgave. Test programmet ved å bruke `Comp` implementasjonene fra oppgave VI.5. **(for Oblig.3?)**

(forelesning denne uken: Graph)

Uke IX. NO – innlevering av Oblig.2: 23.10 (flyttet til onsdag, 28.10) 19-23.10

(forelesning denne uken: Graph)

Uke X. Oblig.2 + Dictionary 26-30.10

X.1

Design `interface Dictionary` og plasser den i en passende `package`.

X.2

Utvid implementasjon av `BinTree` fra VII.2 til `BST`. Implementasjon skal parametriseres med `Comp` som oppgis ved opprettelse av nye objekter.

X.3

Bruk så denne til å implementere `interface Dictionary`. Igjen, skal denne implementasjon være parametrisert av `Comp` som bestemmes ved opprettelse av nye objekter. **(for Oblig.3?)**

(forelesning denne uken: kun onsdag/Peter: Graph (directed, weighted))

Uke XI. Grafer 2-6.11

XI.1 Gjennomganag av Oblig.2 (flyttet fra forrige uke)

XI.2

Design `interface Graph` og implementer (deler av ?) den med nabo-matrise (LL?)

XI.3

Programmer Floyd-Warshall algoritme for beregning av transitiv tillukking G^* av en graf G representert som nabo-matrise. **(for XII.1,Oblig.3)**
eller en (??) ...

XI.4 test ...

⇒ Oblig.3 går ut på fredag, 6.11

Uke XII. Grafer: vektede 9-13.11

XII.1

Utvid FW algoritmen fra XI.3 til å beregne AS-SP "all-pairs shortest-paths" for en graf G. En nabo-matrise representerer en rettet vektet graf G (der tallet x i posisjonen $(3,5)$ sier at det finnes en kant med vekt x fra node 3 til node 5). Utvid FM slik at, gitt en slik graf med ikke-negative vekter, beregner algoritme korteste stier mellom alle par av noder. **(cf. Oblig.3)**

XII.2

Implementer Kruskal algoritme MST **(for Oblig.3)**

Uke XIII. ingen
16-20.11

Uke XIV. NO – innlevering Oblig.3: 27.11
23-27.11

Uke XV. Oblig.3 gjennomgang
30.11-4.12