

# Pensum: fra boken (H-03) + forelesninger

	unntatt	kursorisk	tema
KAP. 1			JAVA – I-110 (ikke gjennomgått)
KAP. 2			OO + <b>ABSTRAKSJON /GENERISK PROGRAMMERING</b> <b>REKURSJON</b>
KAP. 3			ALGORITME-TIDSANALYSE; O-NOTASJON
KAP. 4	4.2.3, 4.2.4, 4.5		STABEL, KØ, LISTE, <i>ADAPTER</i>
KAP. 5		5.1.4	SEKVEN: RANK, <i>POSITION, ACCESSOR, ITERATOR</i>
KAP. 6		6.4.4	TRÆR, B-TRÆR + DSF/ <b>BFS</b> (PRE/POST/IN-ORDER)
KAP. 7		7.3.5, 7.4.2	PRIORITETSKØ: HEAP; <i>COMPARATOR; LOCATOR</i>
KAP. 8	8.6	8.3.3 - 8.3.7, 8.7	ORDBOK: BINÆRSØK, HASHTABELL
KAP. 9	9.2.1 - 9.6		SØKETRÆR: BINÆRE
KAP. 10	10.3.2,10.7	10.1.2, 10.1.3, 10.2, 10.4, 10.5.2, 10.6	SORTERING:QUICKSORT (MERGESORT); RANDOMISERING
<del>kap. 11</del>	<del>Går ut</del>		
KAP. 12	12.7.2	12.4.4	GRAFER: DFS/ <b>BFS</b> , SSSP+ <b>FORD-BELLMAN</b> , MST <i>DECORABLE</i>

# ADT og OO programmering

... i JAVA, dvs. i-110 med mer struktur

## I. ADT I JAVA - INTERFACE

- I.1 grensesnitt skal dokumenteres – Javadoc
- I.2 bruk av interface
- I.3 implementasjoner av interface
- I.4 generisk programmering

## II. OO

- II.1 Arv av type og implementasjon
- II.2 Abstrakte klasser i Java

## III. BRUK OG TILPASSING

- III.1 Arv ...
- III.2 Casting (omstøping)
- III.3 Exceptions (unntak)

# Rekursjon

## **I. TRE AV REKURSIVE KALL,**

**rekursjonsdybde**

**terminering – ordning**

## **II. INDUKTIVE DATA TYPER**

**og Rekursjon over slike**

## **III. “SPLITT OG HERSK” – PROBLEMLØSNING VED REKURSJON (Kap. 10.1.1)**

## **IV. STABEL AV REKURSIVE KALL**

**iterasjon til rekursjon**

**rekursjon implementert som iterasjon**

## **V. KORREKTHET**

**terminering**

**invarianter**

## **VI. ENKEL KOMPLEKSITETSANALYSE AV REKURSIVE ALGORITMER**

# Implementasjon - Kap.3

## *Implementasjon av (A)DT*

- *Data Representasjon*
- *Data Struktur*
- *Data Invariant*
- *Korrekthet*
  - *opprettholdelse av Data Invarianten*
  - *korrekt implementasjon av abstrakte operasjoner*

## *Effektivitet:*

- *tidskompleksitet = hvordan avhenger tidsforbruket av størrelsen på input*
- *kompleksitets-klasser*
  - *linær*
  - *logaritmisk*
  - *polynomisk (kvadratisk)*
  - *eksponensiell*
- *verste-fall analyse*

# Linære Samlinger

- *Container, PositionalContainer, Position, List, Sequence, Vector.....*
- *organisering av ADTer*
  - *hierarki av interface burde avspeile alle relasjoner mellom begrepene*
- *Adapter pattern: ADT-2 kan brukes for en “generisk” implementasjon av en annen ADT-1*
  - *relativ kompleksitet, dvs kompleksitet til ADT-1 avhengig av implementasjon av ADT-2*
- *riktige abstraksjoner kan være vanskelig å finne*
  - *Position (Accessor, Locator)*
- *forskjellige implementasjoner av samme ADT*
  - *vurdering av implementasjoner opp mot hverandre*
- *“generisk” algoritme bruker kun ADTer*
  - *grensesnitt informasjon om parametre*
  - *og derfor kan brukes med vilkårlige implementasjoner*
- *sorterings algoritmer*
  - *seleksjon-, merge-, bobble-sortering*

Kap.	unntatt	kursorisk
4	4.2.3, 4.2.4, 4.5	
5		5.1.4

# Trær : Kap. 6 (kursorisk: 6.4.4; + DFS/BFS)

## 1. *Trær og Binære Trær:*

- *definisjoner og terminologi*
- *egenskaper*

## 2. *Tre-algoritmer – traversering:*

- *DFS og **BFS***
- *DFS :*
  - *pre– og postorder,*
  - *inorder for BinaryTree*

## 3. *Tree og BinaryTree ADT*

## 4. *Implementasjon av trær:*

- *LenketStruktur*
- *Sekvens – BinaryTree*
- *kompleksitet*

# PrioritetsKø : Kap. 7 (kursorisk: 7.4.2, 7.3.5)

## **Typer av Samling**

- *LiFi (Stack, Queue)*
- *PositionalContainer*
- *KeyBasedContainer*

## **Nøkkel:**

- *Totale Ordninger*
- *objekter med nøkler (Item)*

<b>Prioritetskø ADT :</b>	<b>implementasjon</b>	<b>og</b>	<b>sorteringsmønster</b>
	<i>usortert sekvens</i>		<i>seleksjonsort</i>
	<i>sortert sekvens</i>		<i>innstikksort</i>
	<i>heap</i>		<i>heapsort</i>

## **Haug (heap) datastruktur:**

- *Binært tre + datainvariant*
- *innsetting / fjerning ! opprettholdelse av datainvarianten*

## **Locator designmønster:**

- *overtar rollen av Position for KeyBasedContainer*
- *nyttig når objekter/nøkler må oppdateres mens de er i en samling*

# Grafer : Kap. 12 (kursorisk 12.4.4, untatt 12.7.2)

- *Grafer*
  - *Graf (rettet vs. ikke-rettet), terminologi*
  - *ADT og implementasjoner (kant-liste, nabo-liste, nabo-matrise)*
- **Traversering** (*generalisering fra Trær*)
  - DFS – forskjeller rettet vs. ikke-rettet tilfelle*
  - BFS**
- **Algoritmer**
  - transitiv tillukking*
    - *n \* DFS*
    - *Floyd-Warshall : nabo-matrise!*
  - topologisk sortering*
    - DAG*
- **Vektete Grafer**
  - kortest sti*
    - **Ford-Bellman algoritme :  $O(n*k)$**
    - *Dijkstra algoritme :  $O((n+k) \log n)$*
  - MST**
    - *Kruskal algoritme*
    - *implementasjon : find/union*

## **Decorable designmønster:**

- *for dynamisk annotering av noder/posisjoner med*
- *attributter som ikke utgjør en del av datastrukturen med er nødvendige for noen algoritmer*



# Ordbøker

Kap. 8: (kursorisk: 8.3.3 – 8.3.7, 8.7;  
unntatt: 8.6)

Kap. 9: (unntatt: 9.2.1 – 9.7)

- *Ordbok (symboltabell)*  
*avbildning fra nøkler til dataobjekter*
- *Ordbok og Ordnet Ordbok ADT*
  - *med nøkkel-data objekter*
  - *med [Locator](#)*
- *Implementasjon*
  - Sekvens*
    - *usortert (DL, Array)*
    - *sortert (DL, Array)*
  - Hash Tabeller*
    - *valg av hash-funksjon*
    - *håndtering av kollisjoner*
  - Binære Søketrær*
    - *binært tre + **datainvariant***
    - *innsetting/fjerning*
    - *balansering*

# Sortering

• Bubble  $\Theta(n^2)$

prioritetskø basert

• Selection  $\Theta(n^2)$

• Insertion  $O(n^2)$

• Heap  $O(n \log n)$

“splitt og hersk”

• Merge  $O(n \log n)$

• Quick  $O(n \log n)$

---

• Bucket  $O(n + N)$

• Radix

“Comparison based sorting” =  $\Omega(n \log n)$

Kap.10: (kursorisk: 10.1.2, 10.1.3, 10.2,  
10.4, 10.5.2, 10.6

unntatt: 10.3.2, 10.7)

# Eksamen

fredag, 12 desember, kl.9:00 - 15:00

tillatte hjelpemidler: *alle trykte og skrevne*

Må kunne:

- **alle algoritmer og abstrakte data typer** (ADTer):  
disse skal ofte modifiseres/tilpasses
- **alternative implementasjoner** av ADTer:  
forskjellige datastrukturer  
forskjellige virkemåter av en algoritme implementert på forskjellige datastrukturer  
kompleksitetsanalyse av algoritmer
- **rekursjon**:  
skrive/modifisere rekursive metoder  
enkel analyse av kompleksitet av slike
- **generisk programmering**:  
bruk av interface-parametre  
med forskjellige implementasjoner  
som resulterer i “forskjellige” algoritmer