

Uke 8 -
Oppramstyper, HashMap og Innstikksortering, litt
javadoc

18. oktober 2011,
Arne Maus
Inst. for informatikk, UiO



Enummerering – å lage egne oppramstyper

Brukes til å lage 'typer' som har et lite antall verdier, ofte tekst.

```
enum Farge {rød,grønn,blå,gul};

class EnumEks {
    public static void main(String[] args) {

        Farge f = Farge.rød;
        System.out.println("Farge f er:" + f);

        for (Farge ff: Farge.values())
            System.out.println("ff:" + ff
                + ", nummer:" + ff.ordinal());
    }
} // end class EnumEks
```

```
Farge f er:rød
ff:rød, nummer:0
ff:grønn, nummer:1
ff:blå, nummer:2
ff:gul, nummer:3
```

Slik 'enum' kan ha metoder , og hver verdi har et tall assosiert ved seg.

En enum virker omtrent som en klasse-deklarasjon.

```
public enum Karakter {
    A(6),
    B(5),
    C(4),
    D(3),
    E(2),
    F(0);

    final int verdi;

    boolean erBedre(Karakter k){
        return this.verdi > k.verdi;
    }

    Karakter (int v){
        verdi = v;
    }
} // end enum Karakter
```

```
class EnumEks2 {
    public static void main(String[] args) {

        Karakter min = Karakter.A, din = Karakter.E;
        System.out.println("Karakteren min er:" + min);

        if (min.erBedre(din))
            System.out.println("Min karakter:" + min
                + " er bedre enn din " + din);
        }
    } // end class EnumEks2
```

```
Karakteren min er:A
Min karakter:A er bedre enn din E
```



Holde orden på objekter - HashMap

- Ofte har vi flere, mange objekter av en bestemt klasse - eks. :
 - elever på en skole
 - biler som har passert bomringen i Oslo
 - telefonsamtaler fra en bestemt person,.
- Vi har hittil lært arrayer (Elev [] elevene = new Elev[400], Bil [] bomringBiler = new Bil[10000];.....) og må da passe på at vi har
 - nok plass
 - for å finne et bestemt objekt må vi ofte lete gjennom 'hele' arrayen
- Vi skal nå lære en bedre måte å lagre objekter hvor det er viktig å raskt og enkelt finne igjen ett av objektene (som da har ett bestemt kjennetegn som: Navnet til eleven, registreringsnummeret til en bil,...)
- Et slikt kjennetegn som skiller ett objekt fra alle andre objekter, kaller vi en nøkkel (key)
- **HashMap** er svaret



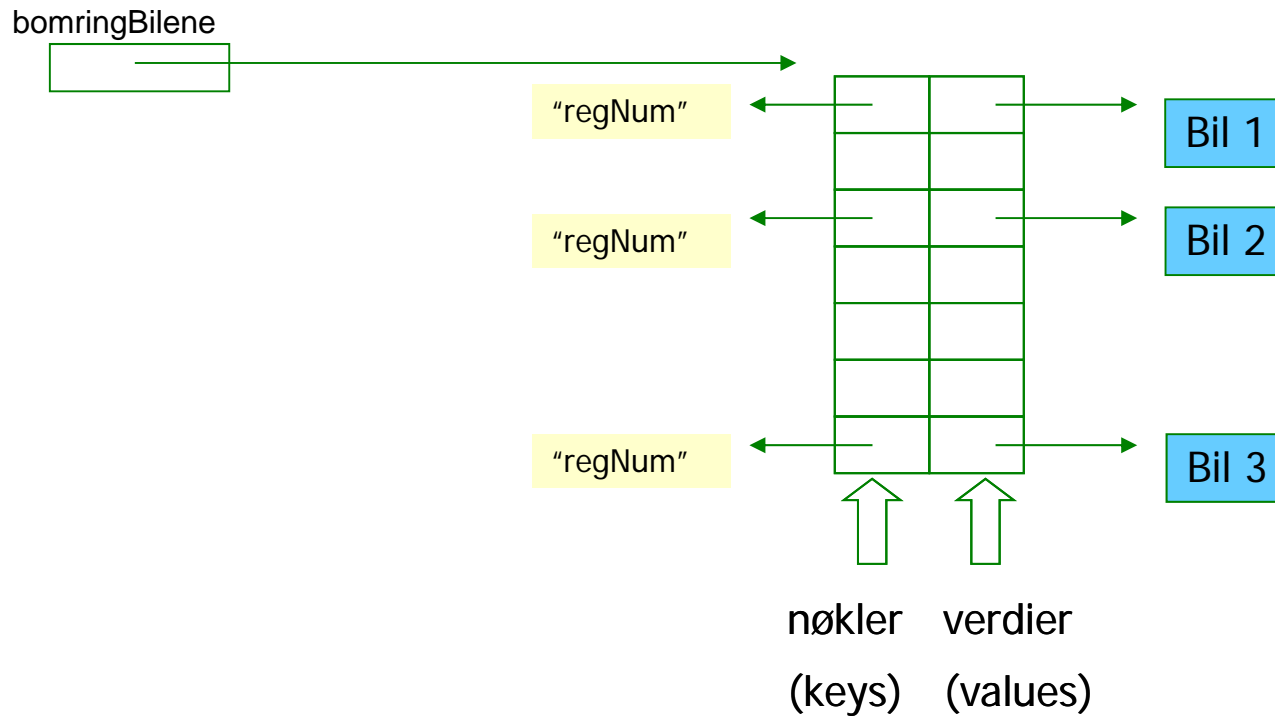
HashMap = lagre objekter med en søkenøkkel

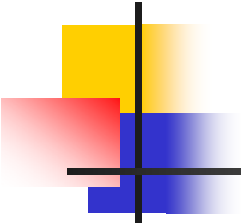
- Brukes til å holde orden på en samling objekter
- Alternativ til arrayer
- Med arrayer kan man:
 - I en array legger vi inn objekter i en bestemt posisjon, og vi må gå tilbake til denne posisjonen/indeksen når vi senere skal se på objektet. Indeksen er et heltall mellom 0 og length-1.
- To viktige forskjeller mellom arrayer og HashMap:
 - I en HashMap oppgir vi en bestemt *nøkkel* (vanligvis en tekststreng) når vi legger vi inn et nytt objekt (kalt *verdien*), og vi oppgir denne nøkkelen når vi senere skal se på objektet. Dvs. indeksen er en tekststreng.
 - En HashMap har ingen gitt lengde når vi lager den. Den 'vokser' når legger inn nye elementer (inntil hele lagerplassen er oppbrukt)
- En Hashmap er mer fleksibel (men langsommere) måte å lagre flere/mange elementer i et program

Hvordan vi skal tenke oss en HashMap

en HashMap er som en slags dobbelt-array (f.eks bomringBilene)

```
HashMap bomringBilene = new HashMap();
```





Ulike versjoner i Java 1.4 (gammel) og Java 1.5/1.6/1.7 av HashMap

- Vi anbefaler klart at 1.5-måten nyttes, da den hjelper deg mot visse feil (som ellers er lett å gjøre).
- 1.4 måten gjennomgås bare en gang (bare) fordi mange gamle programmer inneholder slik kode.

Eksempel på bruk av HashMap (1.5-1.7)

Importer pakken java.util

```
import java.util.*;
```

Opprett en HashMap og forteller hvilke klasser nøkkelen og verdier har.

```
class BrukAvHashMap {  
    public static void main (String[] args) {  
        HashMap<String,Person> h = new HashMap <String,Person>();  
  
        String fnr1 = "30128812344";  
        Person per1 = new Person(fnr1, "Harald Olsen");  
        h.put(fnr1, per1);  
  
        String fnr2 = "14109522547";  
        Person per2 = new Person(fnr2, "Lena Torsen");  
        h.put(fnr2, per2);  
  
        Person p = h.get("30128812344");  
    }  
}
```

Legg inn Person-objekt i HashMap'en

Legg inn Person-objekt i HashMap'en

Hent Person-objekt fra HashMap'en

```
class Person {  
    String fnr;  
    String navn;  
  
    Person(String fnr, String navn) {  
        this.fnr = fnr;  
        this.navn = navn;  
    }  
    String fåNavn() { return navn;}  
}
```


Opprette en HashMap, Java1.4 (gammel) og Java 1.5-1.6

I starten av programmet:

```
import java.util.*;
```

Dette importerer pakken java.util hvor bl.a. klassen HashMap ligger.

- I klassen eller metoden som skal bruke HashMap'en **Java 1.4:**

```
HashMap h = new HashMap();
```

- I klassen eller metoden som skal bruke HashMap'en **Java 1.5** og nyere (best):

```
HashMap <String,Person> h = new HashMap <String,Person>();
```

I Java 1.5 forteller vi hvilke klasser nøkkel- og verdi-objektene kommer fra. Vi sier at vi da låser objektene til både nøkkelen og verdi-objektene til å være av disse typene.

NB: Hvis tabellen skal brukes av flere metoder i en klasse, deklarerer variabelen ovenfor i starten av klassen (som en objektvariabel).

Hvis tabellen kun skal brukes av en enkelt metode, er det naturlig å deklarere HashMap variabelen ovenfor inni den aktuelle metoden.



Legge inn objekt i HashMap (samme i 1.4 og 1.5-1.7)

- Et hvilket som helst objekt i Java kan legges inn i en HashMap, men i 1.5 må det være av den klassen vi har 'lovet' systemet
- Når vi legger et objekt inn i HashMap'en, må vi samtidig oppgi en nøkkel, dvs en tekststreng som entydig identifiserer objektet.
- Vi trenger denne nøkkelen dersom vi senere skal finne eller fjerne objektet i HashMap'en.
- Eksempel:

```
String fnr = "30126512345";  
Person p = new Person(fnr, "Kari Olsen");  
h.put(fnr, p);
```

Her lager vi først et Person-objekt (med passende argumenter) og legger det deretter inn i tabellen med fødselsnummeret som nøkkel.

- Dersom vi legger inn flere objekter med samme nøkkel, er det bare det sist innlagte objektet som blir liggende i tabellen (de andre overskrives):

```
Person p1 = new Person(...);
Person p2 = new Person(...);
Person p3 = new Person(...);
String navn = "Jens";
h.put(navn, p1); // p1 legges inn
h.put(navn, p2); // p2 legges inn og p1 overskrives
h.put(navn, p3); // p3 legges inn og p2 overskrives
```

- Noen ganger må vi konstruere en nøkkel ut fra flere variable for å få entydighet:

```
String lengdegrad = "67.3";
String breddegrad = "53.3";
String posisjon = lengdegrad + ";" + breddegrad;
Fjelltopp fjell = new Fjelltopp(posisjon, "Bjørnefjell");
h.put(posisjon, fjell);
```



Hente objekt fra HashMap – Java 1.4 og 1.5

Java 1.4: For å hente et objekt med utgangspunkt i nøkkelen:

```
// 1.4: Vi vil finne en person ut fra fødselsnummeret:  
Person p = (Person) h.get(fnr);
```

- Legg merke til at vi i 1.4 i starten må skrive i parentes navnet på klassen som objektet tilhører - i dette tilfellet klassen Person.
- Årsaken er at i 1.4 HashMap'en ikke holder rede på hvilken klasse objektene som legges inn har - bare at det er objekter. Når objektene hentes ut må vi derfor "minne Java på" hvilken klasse objektet var av (dette er egentlig et møte med en avansert og svært nyttig mekanisme i objektorienterte språk som kalles *arv* og som blir tatt opp i vårens INF1010).

Java 1.5: For å hente et objekt med utgangspunkt i nøkkelen , nå trenger vi ikke si hvilken klasse objektet har (det har vi jo sagt i deklarasjonen av HashMapen h):

```
// 1.5: Vi vil finne en person ut fra fødselsnummeret:  
Person p = h.get(fnr)
```

- **Merk:** å hente et objekt fra en HashMap slik som over medfører *ikke* at objektet fjernes fra HashMap'en (vi får bare en kopi av en *peker* til objektet).



Fjerne objekt fra HashMap

- For å fjerne et objekt med gitt fødselsnummer som nøkkel:

```
h.remove(fnr);
```

- Dersom det ligger et objekt i HashMap'en med den gitte nøkkelen, blir objektet fjernet og setningen ovenfor returnerer med en peker til objektet som fjernes.
- Dersom det ikke ligger et objekt i HashMap'en med den gitte nøkkelen, returnerer setningen ovenfor verdien **null**.



Løp gjennom alle objekter i HashMap **Java 1.4**

- For å løpe gjennom alle objektene i en HashMap, lager vi en *oppramsing*:

```
Iterator it = h.values().iterator();
```

- Deretter kan vi se på hvert enkelt objekt i HashMap'en ved å gå i løkke:

```
while (it.hasNext()) {  
    Person p = (Person) it.next();  
    System.out.println("Navn: " + p.fåNavn());  
}
```

Løp gjennom alle objekter i HashMap Java 1.5

- For å løpe gjennom alle objektene i en HashMap, lager vi en *oppramsing og låser samtidig det vi skal hente til en bestemt klasse*:

```
Iterator <Person> it = h.values().iterator();
```

- Deretter kan vi se på hvert enkelt objekt i HashMap'en ved å gå i løkke:

```
while (it.hasNext()) {  
    Person p = it.next();  
    System.out.println("Navn: " + p.fåNavn());  
}
```

- **Bedre:** Vi kan også i 1.5 nytte *den nye for-løkka* som automatisk lager en iterator

```
for (Person p: h.values()) {  
    System.out.println("Navn: " + p.fåNavn());  
}
```



To måter å løpe gjennom en HashMap – 1.5

- Løpe gjennom objektene (som på forrige foil):

```
Iterator <Person>it = h.values().iterator();

while (it.hasNext()) {
    Person p = it.next();
    <gjør noe med objektet p>
}
```

- Løpe gjennom nøklene:

```
Iterator <String> it = h.keySet().iterator();

while (it.hasNext()) {
    String nøkkel = it.next();
    <gjør noe med nøkkelen>
}
```




Metoder i HashMap

Metode	Eksempel	Beskrivelse
put	<code>h.put(nøkkel, objekt);</code>	Legg inn objekt med gitt nøkkel
get -1.4 get -1.5	<code>Person p = (Person) h.get(nøkkel);</code> <code>Person p = h.get(nøkkel);</code>	Finn objekt
remove	<code>h.remove(nøkkel);</code>	Fjern objekt
containsKey	<pre>if (h.containsKey(nøkkel)) { // gjør et eller annet }</pre>	Sjekk om nøkkel finnes i tabell
values	<code>Iterator it = h.values().iterator();</code>	Lag oppramsing av objektene
keySet	<code>Iterator it = h.keySet().iterator();</code>	Lag oppramsing av nøklene

Iterator (oppramsing)

	Eksempel	Beskrivelse deklararasjon
hasNext()	<pre>while (it.hasNext()) { < les neste og gjør noe>; }</pre>	returterer true hvis flere objekter igjen i oppramsingen
next() 1.5 next() 1.4	<pre>Person p = it.next(); Person p = (Person) it.next();</pre>	Finn neste objekt
remove()	<pre>Person p = it.next(); if (p.navn.equals("Arne")) it.remove();</pre>	Fjern siste objekt som ble returnert med next()
	<pre>while (it1.hasNext()) { Person p1 = it1.next(); } for (Person p2 : h.values()){ }</pre>	To måter å gå gjennom alle verdiene (objektene) i h

```
import java.util.*;
import easyIO.*;
```

```
class Hasheksempel {
    public static void main(String[] argv) {
```

```
        In tastatur = new In();
```

```
        HashMap <String,Person> personregister = new HashMap <String,Person>();
```

```
        System.out.print("Antall personer som registreres : ");
```

```
        int ant = tastatur.inInt();
```

```
        for (int i = 0; i < ant; i++) {
```

```
            System.out.println("Gi neste person");
```

```
            Person p = new Person(tastatur);
```

```
            personregister.put(p.telefonnr, p);
```

```
        }
```

```
        // Skriv ut alle personobjektene
```

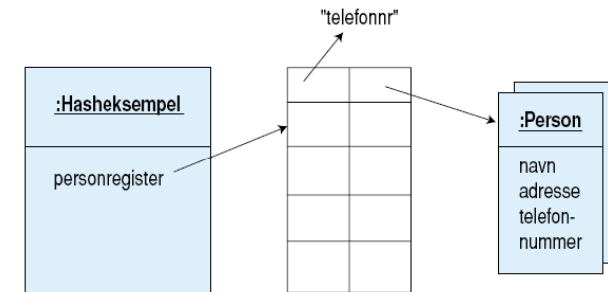
```
        System.out.println("Viser alle personer" +
            "(ukjent rekkefølge):");
```

```
        for (Person p: personregister.values()){
```

```
            p.skrivData();
```

```
        }
```

```
    }
```



```
class Person {
```

```
    String navn, adresse, telefonnr;
```

```
    Person (In tastatur) {
```

```
        System.out.print("Oppgi navn : ");
```

```
        navn = tastatur.inLine();
```

```
        System.out.print("Oppgi adresse : ");
```

```
        adresse = tastatur.inLine();
```

```
        System.out.print("Oppgi telefonnummer : ");
```

```
        telefonnr = tastatur.inLine();
```

```
    }
```

```
    void skrivData() {
```

```
        System.out.println("Navn : " + navn);
```

```
        System.out.println("Adresse : " + adresse);
```

```
        System.out.println("Telefonnummer : “
```

```
            +telefonnr);
```

```
    }
```

```
    String fåNavn() { return navn;}
```

```
}
```

Eksempel fra boka s.189



Sortering

- Lære å løse et vanskelig problem
 - Sortering – mange metoder, her Innstikksortering
 - Sortere hva:
 - Heltall
 - Tekster
- Lære abstraksjon
 - Når vi har løst ett problem, kan lignende problemer løses tilsvarende
- Lære å lage 'proff' programvare ved å lage en generell klasse (en vektøyboks) for sortering
 - Hvordan deklarene en slik klasse
 - Javadoc – lage dokumentasjon
 - Testing
 - Hvordan utvikle programmet



Sortering

- Mange datatyper kan sorteres
 - Tall
 - Tekster (leksikografisk = i samme rekkefølge de ville stått i et leksikon)
 - Tabeller av tekster eller tall
- Vi må ha en algoritme (fremgangsmåte) for sortering
 - Det finns mange titalls (hundretalls) metoder for sortering
 - Dere skal lær den som er raskest når vi skal sortere få elementer, si < 50 elementer



Hvorfor sorterer vi

- For å få noen tall i sortert rekkefølge
 - eks: lotto-tallene
- Sortere tekster (navnelister)
- Sortere noen opplysninger som hører sammen. Sorterer da på en av opplysningene.
 - Eks. Telefonkatalogen: navn, adresse, telefonnummer sortert på navn
 - Eks. Oslo Maraton eller Birken sortert etter :navn, plassering



Vi skal først lære å sortere heltall

- Dette skal vi så med minimale endringer bruke til å sortere:
 - String-arrayer (tekster)



Vi ønsker en klasse med to varianter av sortering:
Heltall og tekster

```
public class ISort {  
  
    public static void sorter(int [] a) {  
  
    }  
    public static void sorter(String [] a) {  
  
    }  
} // end class ISort
```



```
class TestInnstikkSortering
{
```

Test-program for sortering

```
    public static void main ( String[] args) {

        int [] a = {3,1,7,14,2,156,77};
        String [] navn = {"Ola", "Kari", "Arne", "Jo"};

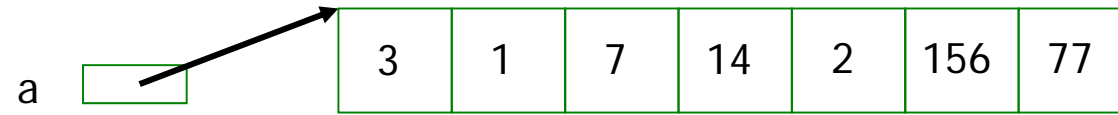
        // sorter heltall - skriv ut
        ISort.sorter(a);
        for (int i = 0; i < a.length; i++)
            System.out.println("a[" + i +"]= " + a[i]);

        System.out.println("\n Test tekst-sortering:");

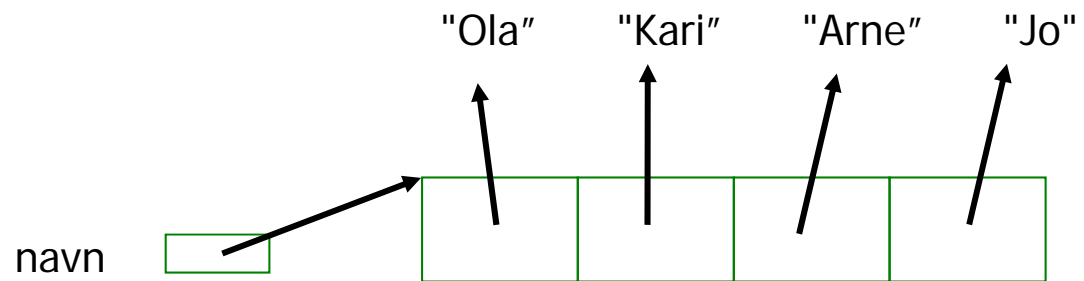
        // sorter Stringer - skriv ut
        ISort.sorter(navn);
        for (int i = 0; i < navn.length; i++)
            System.out.println("navn[" + i +"]= " + navn[i]);

    }}
}
```

heltalls-array



en-dimensjonal
String-array



```
>java InnstikkSortering
```

Test av test-programmet med **tomme** sortering-metoder

```
a[0]= 3
```

```
a[1]= 1
```

```
a[2]= 7
```

```
a[3]= 14
```

```
a[4]= 2
```

```
a[5]= 156
```

```
a[6]= 77
```

Test tekst-sortering:

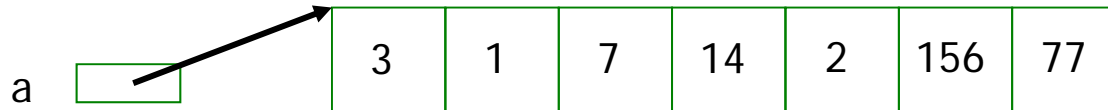
```
navn[0]= Ola
```

```
navn[1]= Kari
```

```
navn[2]= Arne
```

```
navn[3]= Jo
```

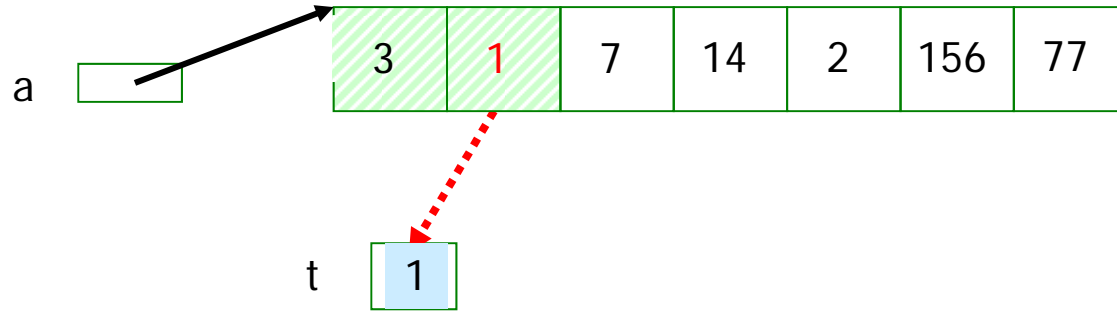
En algoritme for å sortere heltall – innstikksmetoden



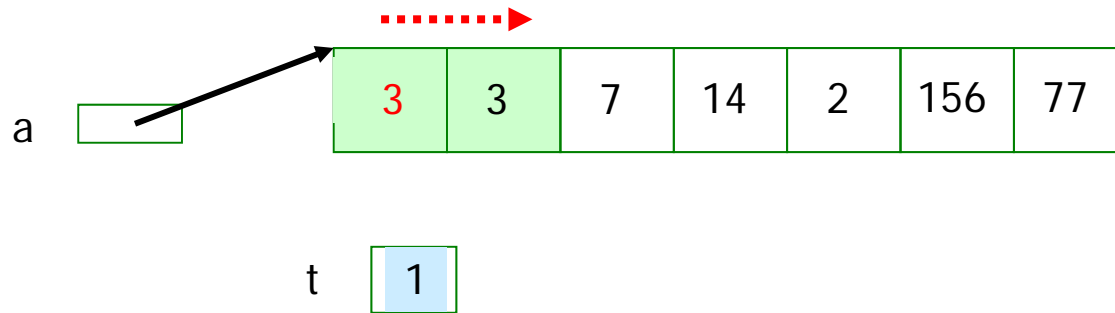
- Se på arrayen *ett for ett* element fra venstre mot høyre
- a ■ Sorterer det vi hittil har sett på, ved :
 - Hvis det nye elementet vi ser på **ikke** er sortert i forhold til de vi allerede har sett på:
 - Ta ut dette elementet (gjem verdien i en variabel **t**)
 - Skyv på de andre elementene vi her sett på en-etter-en, ett hakk høyreover til elementet i **t** kan settes ned på sortert plass.
 - Da er den delen vi har sortert ett element lenger (fra venstre)
 - Når vi har sett på alle elementene, er hele arrayen sortert
 - Observasjon : Det første elementet, er det sortert i forhold til seg selv

Sorter 1 på plass i forhold til 3

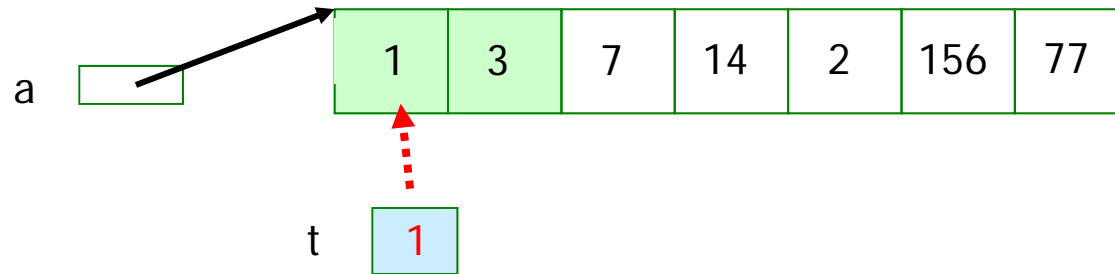
steg 1



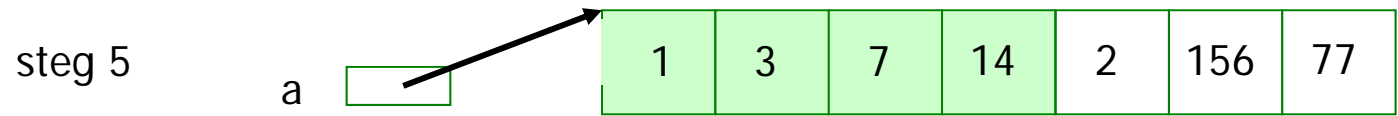
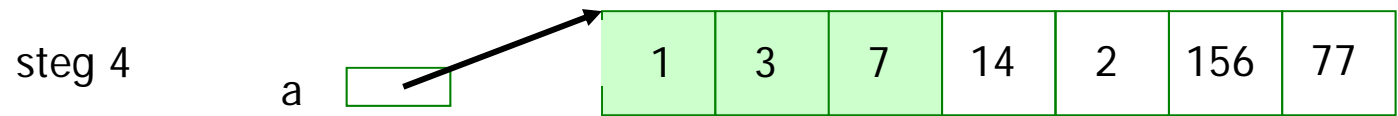
steg 2



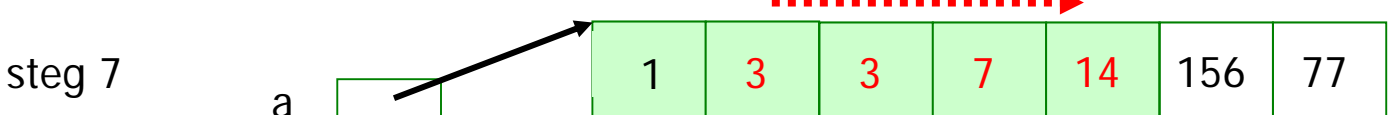
steg 3

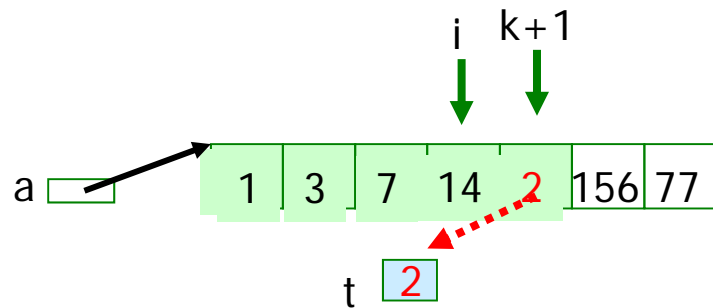


7 og 14 står riktig, Sorter 2 på plass i forhold til : 1,3,7,14



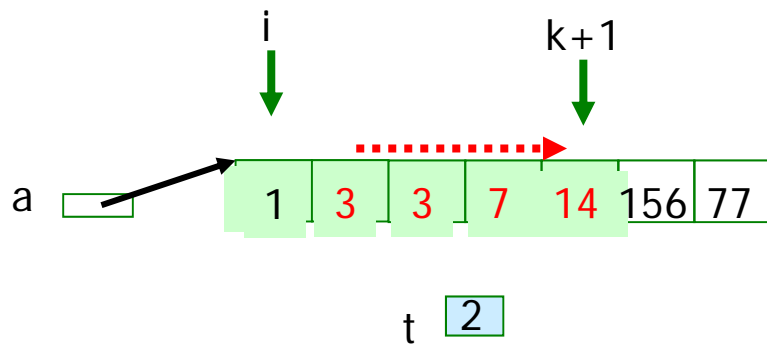
flytt: 14, 7 og så 3 ett hakk til høyre



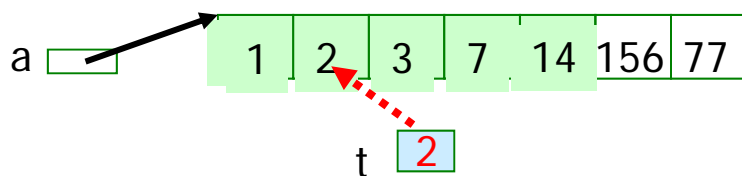


Kode for å flytte **ett** element på plass :

```
// a[k + 1] står på
// feil plass, ta den ut
int t = a[k + 1], i = k;
```



```
// skyv a[i] mot høyre ett hakk til
// vi finner riktig plass til t
while (i >= 0 && a[i] > t) {
    a[i + 1] = a[i];
    i--;
}
```



```
// sett t inn på riktig plass
a[i + 1] = t;
```

```

public class ISort {

    public static void sorter(int [] a) {
        for (int k = 0 ; k < a.length-1; k++) {
            if (a[k] > a[k+1]) {
                // a[k + 1 ] står på feil plass, ta den ut
                int t = a[k + 1], i = k;
                // skyv a[i] mot høyre ett hakk til
                // vi finner riktig plass til t
                while (i >= 0 && a[i] > t) {
                    a[i + 1] = a[i];
                    i--;
                }
                // sett t inn på riktig plass
                a[i + 1] = t;
            }
        }
    } // end heltall-sortering
}

```



```
>java InnstikkSortering
```

```
a[0]= 1
```

```
a[1]= 2
```

```
a[2]= 3
```

```
a[3]= 7
```

```
a[4]= 14
```

```
a[5]= 77
```

```
a[6]= 156
```

Resultat av sortering med heltalls-metoden kodet, den andre uten kode

```
Test tekst-sortering:
```

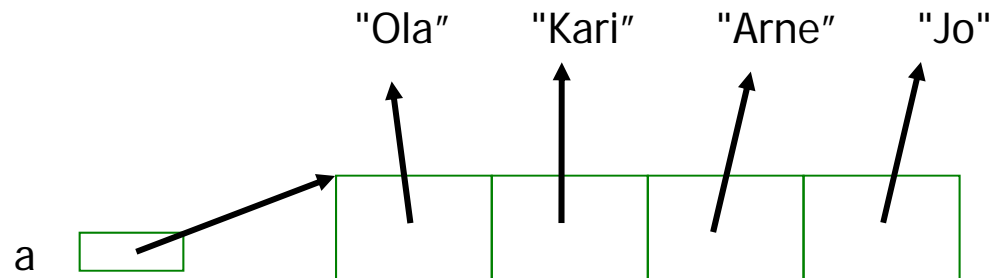
```
navn[0]= Ola
```

```
navn[1]= Kari
```

```
navn[2]= Arne
```

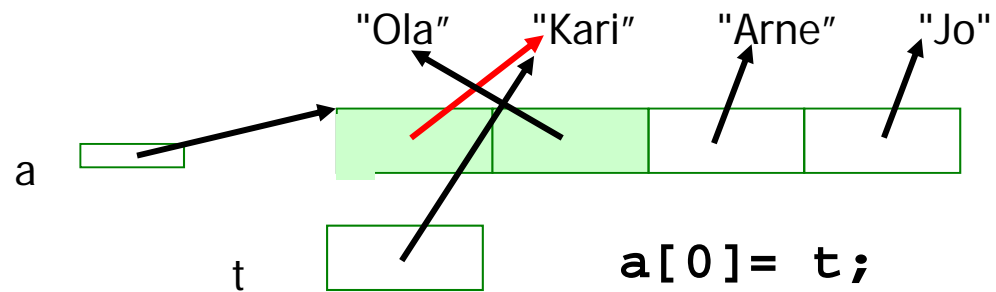
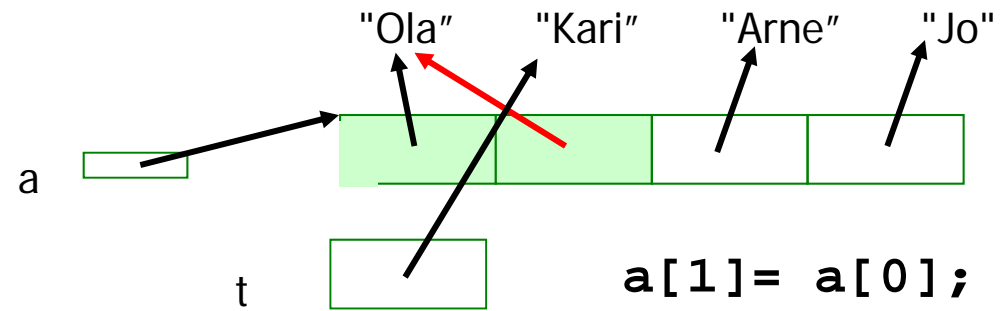
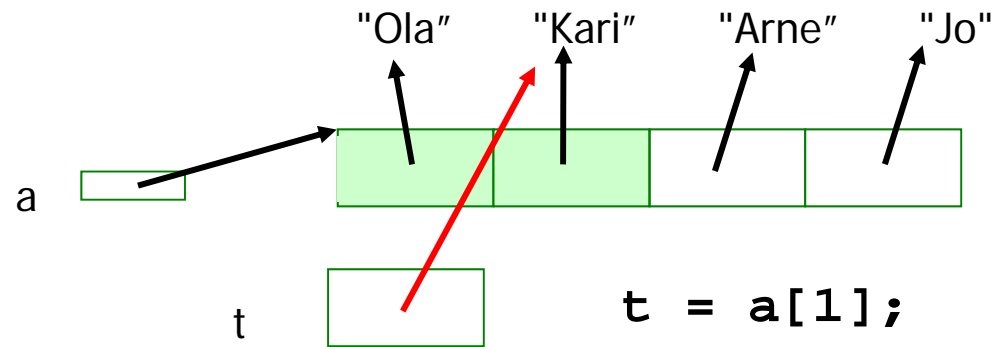
```
navn[3]= Jo
```

Sortering av tekster (String)



- Vi skal sortere denne ved å bytte om på pekerne (la `a[0]` peker på "Arne", ..osv) med innstikkmetoden

Sortere de to første elementene ved å bytte om pekere



```

public static void sorter(int [] a) {
    // Sorterer heltallsarrayaen 'a'.
    for (int k = 0 ; k < a.length-1; k++) {
        if (a[k] > a[k+1]) {
            int t = a[k + 1];
            int i = k;
            while (i >= 0 && a[i] > t) {
                a[i + 1] = a[i];
                i--;
            }
            a[i + 1] = t;
        }
    }
} // end heltall-sortering

public static void sorter(String [] a) {
    // Sorterer String-arrayen 'a'.
    for (int k = 0 ; k < a.length-1; k++) {
        if( a[k].compareTo(a[k+1]) > 0 ){
            String t = a[k + 1];
            int i = k;
            while (i >= 0 && ( a[i].compareTo(t) > 0 ) ){
                a[i + 1] = a[i];
                i--;
            }
            a[i + 1] = t;
        }
    }
} // end String-sortering

```

```
>java InnstikkSortering
```

```
a[0]= 1
```

```
a[1]= 2
```

```
a[2]= 3
```

```
a[3]= 7
```

```
a[4]= 14
```

```
a[5]= 77
```

```
a[6]= 156
```

Test med heltall og enkel String-sortering kodet

Test tekst-sortering:

```
navn[0]= Arne
```

```
navn[1]= Jo
```

```
navn[2]= Kari
```

```
navn[3]= Ola
```



Javadoc – proff dokumentasjon av klassene

- Legg inn spesielle kommentarer i programmet ditt (over hver metode og klasse)
- I disse kommentarene kan man legge HTML-kommandoer (som `
` for å få linjeskift)
- Kjør programmet 'javadoc', og automatisk har du en fin dokumentasjon
- Største fordel: Kode og dokumentasjon vedlikeholdes på samme fil.

```

/**
 * Klasse for sortering etter 'innstikk-metoden', se
 * Rett på Java - kap.5.6.
 * Sortering av heltallsarray, tekster og en to-dimensjonal
 * tekst-array sortert etter verdiene i første kolonne.<br>
 *
 * N.B. Bare velegnet for mindre enn 100 elementer.
 *
 * Copyright : A.Maus, Univ. i Oslo, 2008
 *****/
public class ISort {

    /**
     * Sorterer heltall i stigende rekkefølge.
     * @param a heltallsarrayen som sorteres. <br>
     * Endrer parameter-arrayen.
     *****/
    public static void sorter(int [] a) {
    }

    /**
     * Sorterer String-arrayer i stigende leksikografisk orden.
     * @param a arrayen som sorteres.<br>
     * Endrer parameter-arrayen
     *****/
    public static void sorter(String [] a) {

    }

} // end class ISort

```



Dokumentasjon av klassen og metodene - javadoc

```
M:\INF1000\Isort>javadoc -package ISort.java
Loading source file ISort.java...
Constructing Javadoc information...
Standard Doclet version 1.5.0_02
Building tree for all the packages and classes...
Generating ISort.html...
Generating package-frame.html...
Generating package-summary.html...
Generating package-tree.html...
Generating constant-values.html...
Building index for all the packages and classes...
Generating overview-tree.html...
Generating index-all.html...
Generating deprecated-list.html...
Building index for all classes...
Generating allclasses-frame.html...
Generating allclasses-noframe.html...
Generating index.html...
Generating help-doc.html...
Generating stylesheet.css...
```

```
M:\INF1000\Isort>
```


ISort - Windows Internet Explorer

M:\INF1000\isort\index.html

File Edit View Favorites Tools Help

Class Hierarchy ISort

All Classes
[ISort](#)

Package **Class** [Tree](#) [Deprecated](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#) [FRAMES](#) [NO FRAMES](#)
[SUMMARY: NESTED](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#) [DETAIL: FIELD](#) | [CONSTR](#) | [METHOD](#)

Class ISort

java.lang.Object
└ ISort

```
public class ISort
extends java.lang.Object
```

Klasse for sortering etter 'innstikk-metoden', se Rett på Java - kap.5.7. Sortering av heltallsarray og tekster .
sortert etter verdiene i første kolonne.
N.B. Bare velegnet for mindre enn 100 elementer. Copyright : A.Maus, Univ. i Oslo, 2008

Constructor Summary

[ISort\(\)](#)

Method Summary

static void	sorter (int[] a) Sorterer heltall i stigende rekkefølge.
static void	sorter (java.lang.String[] a) Sorterer String-arrayer i stigende leksikografisk orden.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Local intranet 100%

ISort - Windows Internet Explorer

file:///M:/INF1000/Isort/ISort.html#sorter(int[]) Google

File Edit View Favorites Tools Help

Class Hi... ISort ISort x

Method Detail

sorter

```
public static void sorter(int[] a)
```

Sorterer heltall i stigende rekkefølge.

Parameters:
a - heltallsarrayen som sorteres Endrer parameter-arrayen.

sorter

```
public static void sorter(java.lang.String[] a)
```

Sorterer String-arrayer i stigende leksikografisk orden.

Parameters:
a - arrayen som sorteres Endrer parameter-arrayen

Package **Class** [Tree](#) [Deprecated](#) [Index](#) [Help](#)

PREV CLASS NEXT CLASS

SUMMARY: NESTED | FIELD | [CONSTR](#) | [METHOD](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

DETAIL: FIELD | [CONSTR](#) | [METHOD](#)

Local intranet 100%

Class Hierarchy - Windows Internet Explorer

M:\INF1000\Isort\overview-tree.html

File Edit View Favorites Tools Help

Package Class **Tree** Deprecated Index Help

PREV NEXT [FRAMES](#) [NO FRAMES](#) [All Classes](#)

Hierarchy For All Packages

Class Hierarchy

- o java.lang.Object
 - o [ISort](#)

Package Class **Tree** Deprecated Index Help

PREV NEXT [FRAMES](#) [NO FRAMES](#) [All Classes](#)

Local intranet 100%

Index - Windows Internet Explorer

M:\INF1000\Isort\index-all.html

Google

File Edit View Favorites Tools Help

Index

Package Class **Tree** Deprecated **Index** Help

PREV NEXT [FRAMES](#) [NO FRAMES](#) [All Classes](#)

[IS](#)

I

[ISort](#) - Class in [<Unnamed>](#)
Klasse for sortering etter 'innstikk-metoden', se Rett på Java - kap.5.7.

[ISort\(\)](#) - Constructor for class [ISort](#)

S

[sorter\(int\[\]\)](#) - Static method in class [ISort](#)
Sorterer heltall i stigende rekkefølge.

[sorter\(String\[\]\)](#) - Static method in class [ISort](#)
Sorterer String-arrayer i stigende leksikografisk orden.

[IS](#)

Package Class **Tree** Deprecated **Index** Help

PREV NEXT [FRAMES](#) [NO FRAMES](#) [All Classes](#)

Local intranet 100%