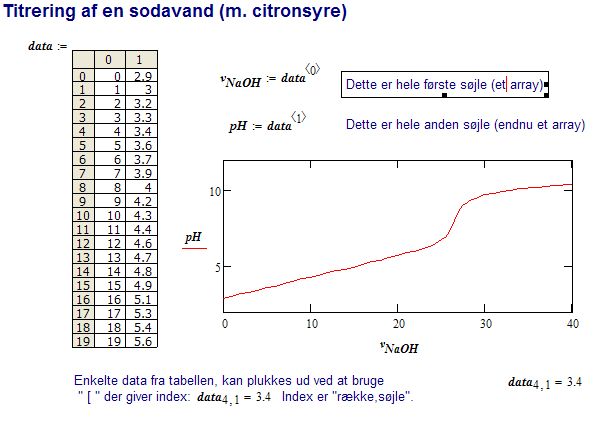
## 13.0 Arrays

Arrays er en gruppe af sammenhængende hukommelsespladser, der alle har samme navn og er af samme type.

Fra MathCad kender du **typen data**. Den bruger   
du til at lægge en række målte tal­værdier ind i. (Svarer til at vælge to søjler i Excel eller regneark i Geogebra)

Sådan en datatabel er et   
eksempel på en 2D array.

På billedet er vist 20 elementer nedad og 2 elementer henad.  
Bemærk at nummereringen starter ved 0.

Arrays er praktiske til at opbevare informationer i og til enkelt at håndtere disse. F.eks. kunne man ønske at sortere dem og gemme dem på harddisken.

### 13.1 Oprettelse af arrays.

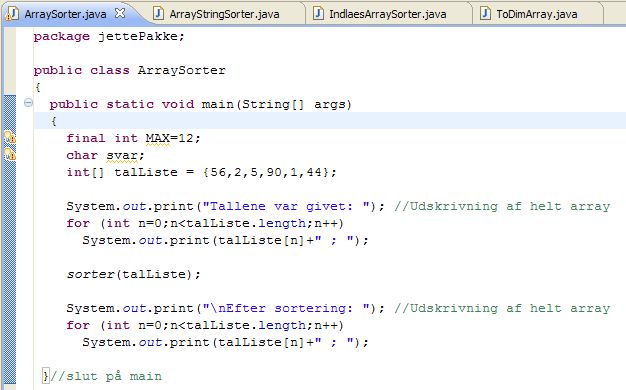
Arrays skal erklæres, så der kan reserveres den nødvendige plads. Når først et array er dannet, kan længden ikke ændres[[1]](#footnote-1). Her laves en liste med plads til 4 heltal.  
 ***int****[] talListe =* ***new*** *int[4];*

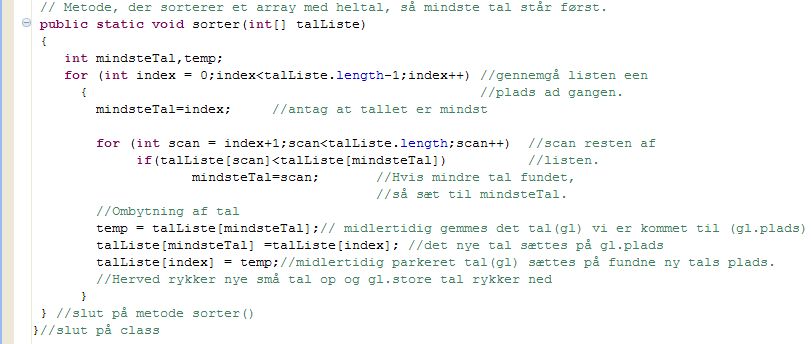
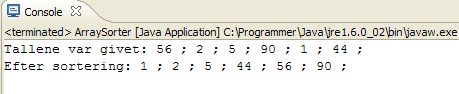
Der er ikke lagt nogen værdier ind, men default vil der blive lagt 0 ind i tal-arrays. I arrays bestående af objekter ligger default ***null*** på alle pladser.

Alternativt kan et array laves sådan: ***int****[] talListe = {4,89,16,11};*   
Her lægges værdier ind med det samme og antallet fremgår af element-antallet. Et array indeholder nogle metoder, så man f.eks. kan få oplyst længden af et array: ***int*** længde = talListe.length;

På næste side er vist klassen ”ArraySorter”, der sorterer nogle givne tal og udskriver dem. Læg mærke til hvor let mange tal udskrives ved at bruge løkken ***for****(startværdi, stopværdi, tilvækst).* Sorterings-metoden (linie) kræver din særlige opmærksomhed, hvis du ønsker at forstå den fuldstændig.

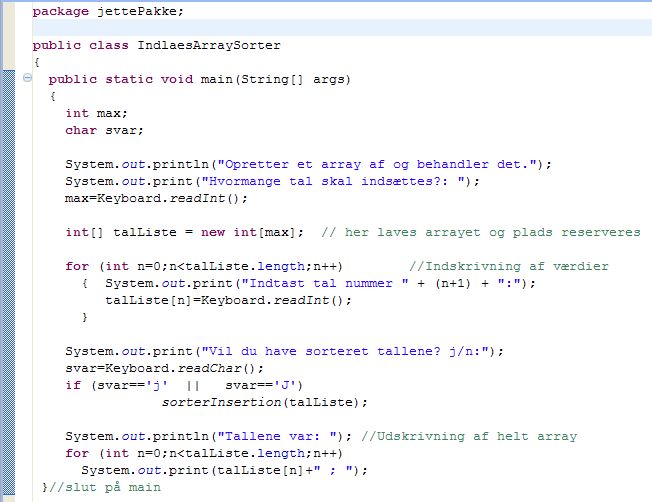
Se kommentarer i selve koden. Følgende program udskriver nogle givne tal, sorterer dem og udskriver dem igen:

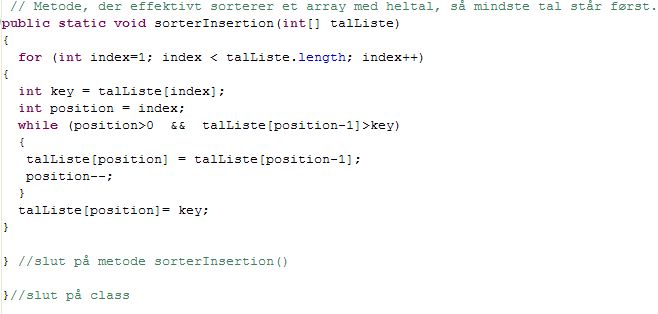
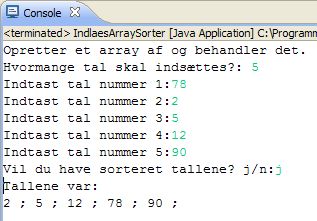




Det ser således ud, når programmet afvikles:

## 13.2 Effektive sorteringsrutiner

Der findes en lidt mere effektiv sorteringsrutine, hvor kun mindre tal flyttes, så der ikke bruges unødig tid og plads. I nedenstående klasse sorteres en række indtastede tal efter den nye sorteringsmetode:



Programmeringsmæssigt er det meget vigtigt at sorteringsrutinerne er så effektive som muligt. Når vi arbejder med vores små opgaver, hvor vi flytter rundt på 10-20 tal eller tekststrenge, ser alt tilforladeligt ud.   
Men hvad hvis det nu f.eks. er patientjournaler på et hospital der tælles i mange tusinder om året og nye patienter kommer til i hundredevis hver dag?  
Det må kun tage et split-sekund at gemme og hente journalen. Derfor skal rutinerne være nøje gennemtænkt, ikke bruge overflødige variable og ikke flytte på allerede sorterede datamængder.  
Studér emnet i faglitteraturen, hvis du har brug for at lave programmer, der skal sortere store datamængder. Det er udenfor disse lektioners ramme at behandle emnet i detaljer, men se f.eks. følgende link:

<https://www.toptal.com/developers/sorting-algorithms>

<http://cs.au.dk/~gerth/dADS1-12/slides/sorting.pdf> (teori Aalborg Uni)

<http://people.math.aau.dk/~olav/dmge04/slilek3.pdf>

<http://www.imada.sdu.dk/~kudhal08/Noter.pdf>

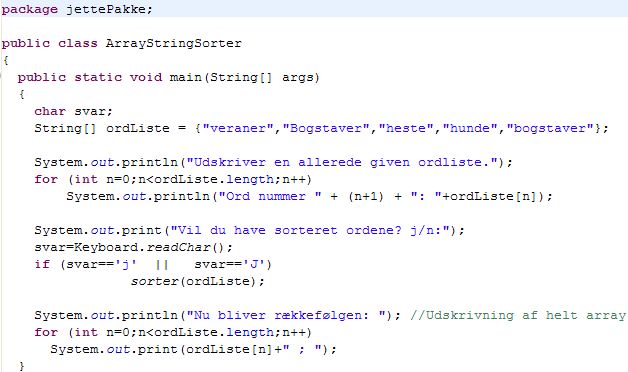
<http://www2.compute.dtu.dk/courses/02105+02326/2015/>

## 13.3 Arrays af objekter

Arrays af objekter

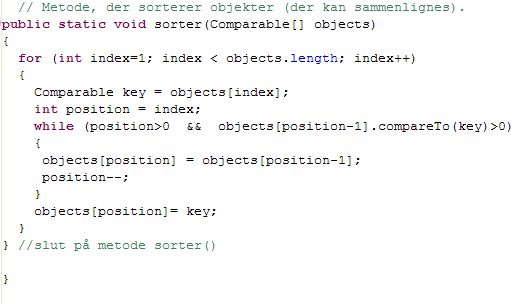
Et array kan også indeholde hele objekter. Faktisk er *String* ikke en variabel, men et objekt. Forudseende programmører har sørget for at dette objekt indeholder metoden *compareTo(object key*). Den sorterer alfa­be­tisk og ord med store bogstaver før små bogstaver.

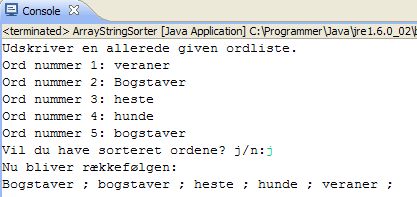
Her er vist et program-eksempel, der håndterer stringobjekter:



Metoden, der kaldes i linie 17, og som sorterer, er vist på næste side.

Her sorteres objekter (i dette eksempel tekststrenge) ved at bruge objekterne indbyggede metode til at sammenligne med.



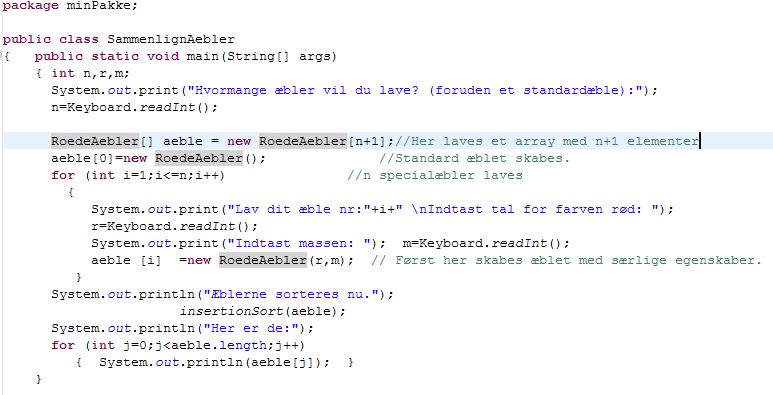


Her er afviklingen:

## 13.4 Comparable objekter.

Hvis du selv skaber nogle objekter, f.eks. et Moent-objekt eller et Terning-objekt, er det op til dig at lave en metode, der bestemmer hvordan objekterne skal rangeres.   
Ved terningen er det oplagt at 1 øje er lavere end 6 øjne, men hvad er ”lavest” af plat eller krone?  
Er det bedste æble det sureste, det rødeste, det rundeste, det tungeste?  
Kun skaberen af objektet ved, hvilke egenskaber objektet besidder og hvordan de skal rangeres. For at gøre objekter sammenlignelige skal de implementere interfacet *Comparable* det vil sige at programmøren skal lave metoden *compareTo(object)*., der returnerer heltal < 0 hvis objektet, der sammenlignes med, rangerer lavere, returnerer 0 hvis objekterne har samme rang, og returnerer heltal>0 hvis objektet har højere rang.

Her under laves nogle virtuelle æbler, der skal sammenlignes. Der er to klasser: En ”driver-klasse” der laver et array af æbler og sorterer dem og en klasse, der laver æbler og definerer høj/lav rang.



9  
10

16

19

## SammenlignInsertion Arrayet til æblerne laves i linie 9 og første æble skabes i linie 10 (default æble), resten af æblerne (bruger-indtastede) i linie 16. I linie 19 sorteres hele arrayet, ved at kalde en metoden insertionSort(aeble).

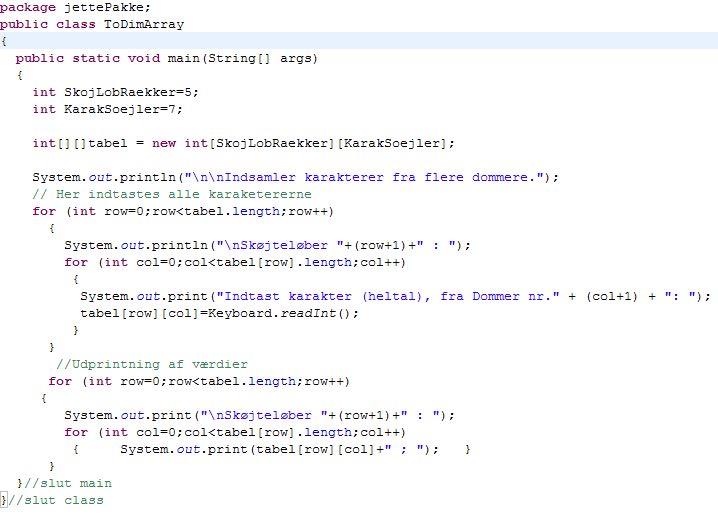
## Sorteringen foregår ved indsættelse, så objekter med højere rang rykker mod højre. Lavere rangsobjekter indsættes til venstre. Herved står det objekt, der har laveste rang helt til venstre/først/øverst, og derefter vokser rangen, jo længere mod højre/ned du kommer.SorterÆblerRun1Her kan du se udklip af programafviklingen.

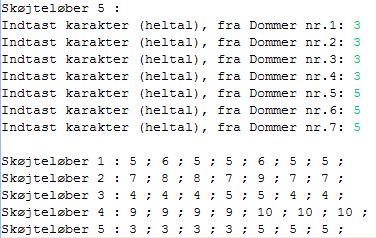
## SorterÆblerRun2

## RoedeAeblerMen hvordan skal æbler sorteres? Det ved objektet selv, se klassen RoedeAebler herunder. Bemærk at klassen lover at lave interfacet Comparble, dvs. metoden compareTo() skal laves og returner et heltal (<0,=0,>0), alt efter hvad programmøren vælger, der skal rangeres højest. Her har jeg valgt, at de æbler med lavest rødtal (0-255) står først. Altså de mindst røde først og de mest røde tilsidst. Hvis de er lige røde, skal letteste æbler først og de tungeste tilsidst.

## 13.5 Multi-dimensionale arrays.

Da det er muligt at lægge objekter ind i et array, kan man lægge et nyt array ind på hver plads. Hermed er det muligt at danne n-dimensionale arrays.   
I denne lektion nøjes vi med at se et eksempel på et 2-dimensionalt array.

Programmet indlæser dommerkarakterer til nogle skøjteløbere.   
Det 2-dimensionale array skabes (instantieres) i linie 9.

Programafvikling:  
Her er sidste del af indtastningen  
samt udprintningen.

------ slut på lektion ------

1. Dog kan man lave et nyt array med ny længde og lægge det gamle over i dette. Der findes også en type ”Vector” som man kan ændre størrelse på. Den er også en array, men sørger selv for at lave ny kortere/længere array efter behov. [↑](#footnote-ref-1)