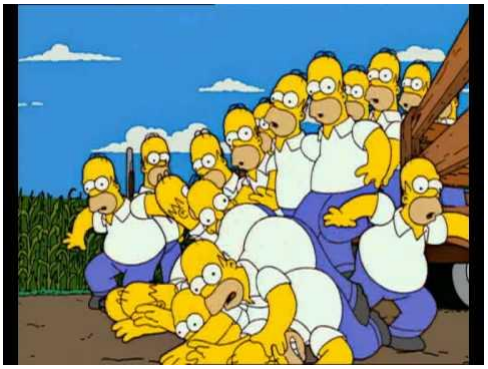
ex3arraysfor

Joachim von Hacht

Många av Samma



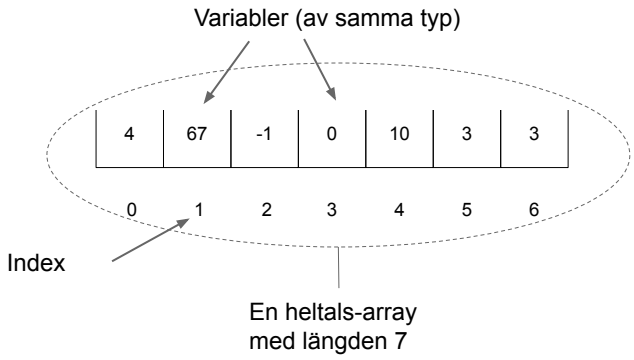
Literaler och variabler kan användas för enstaka värden.

- Men vad händer om vi behöver många värden …
- … 10 heltalsvariabler, eller 100, eller 100000 …. ?
- Orimligt att deklarera dessa en och en.
- Lösning: Om vi behöver <u>många variabler av samma typ</u> kan vi använda en array …
- En array ger dessutom en struktur som är användbar, mer nedan.

Finns tyvärr inget bra svenskt namn, “vektor” och “fält” förekommer.

- Vi använder det engelska namnet array.

Array: Konceptuell bild



3

En array är en konsekutiv (inga tomrum) följd av variabler.

- Antal variabler bestäms en gång för alla, förändras aldrig!
- Längden för arrayen är antalet variabler.
- Varje variabel har ett index.
- Index börjar på 0 och slutar på längden-1.

Array-typer

```
// Arrays types
int[] ia;
double[] da;
boolean[] ba;
```

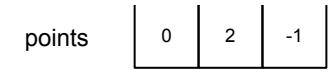

Array-typer skapas utifrån någon existerande typ genom att lägga till hakparenteser efter grundtypen.

- Vi säger t.ex. int-array för typen int[]

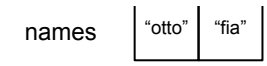
Deklaration och Initiering

```
// Declare and initialize array variable "points"
int[] points = {0, 2, -1};
```

Array-typer




```
String[] names = {"Otto", "Fia"};
```



5

För att använda en array måste vi som alltid deklarera variabel för den.

Deklaration görs som vanligt med: Typ, namn och ev. initiering

- Namnet gäller för hela arrayen (de enskilda variablerna i array:en har inga namn).
- Man initierar genom att ange värdena för de enskilda variablerna i en lista då vi deklarerar arrayen.
    - Enskilda variablerna skapas och initieras med värden från listan i skriven ordning (från vänster till höger)
    - Array:en kommer att innehålla lika många variabler som värden vi angav
    - Mer senare

Utskrift av Array:er

```
int[] points = { 1, 2, 3, 4, 5 };

// Will print like [I@330bedb4
out.println(points);

// Better, use "built in" helper
String s = Arrays.toString(points);
out.println(s);  // Will print [1, 2, 3, 4, 5]
```


Att direkt använda out.println() ger en (normalt) oanvändbar/obegriplig utskrift t.ex. [I@7f31245a

- Uttrycket Arrays.toString(), omvandlar arrayen till en sträng (text) som sedan kan skrivas ut.

Indexering

```
int[] points = {0, 0, 0};

points[0] = 4;                  // { 4, 0, 0 }
points[2] = 1;                  // { 4, 0, 1 }
points[1] = points[2];          // { 4, 1, 1 }
points[0] == points[2];         // false
int i = 1;
points[0] > points[i];          // true

points[6] = 3;                  // Exception
points[-1] = 3;                 // Exception
```

7

Indexering innebär att komma åt de enskilda variablerna, **elementen**, i en array

- Indexering skrivs: *array-namn* [*index*]
  - [ ] (hakparenteser) kallas indexeringsoperatorn
  - Förvirrande: [ ] används även vid deklaration men betyder då bara array-typ
- Index måste var ett heltalsuttryck (en variabel går bra).
- Vi måste själva se till att index inte hamnar utanför array:en, …
  - … om utanför så, undantag (exception) vid körning, ett **exekveringsfel**

Längden av en Array

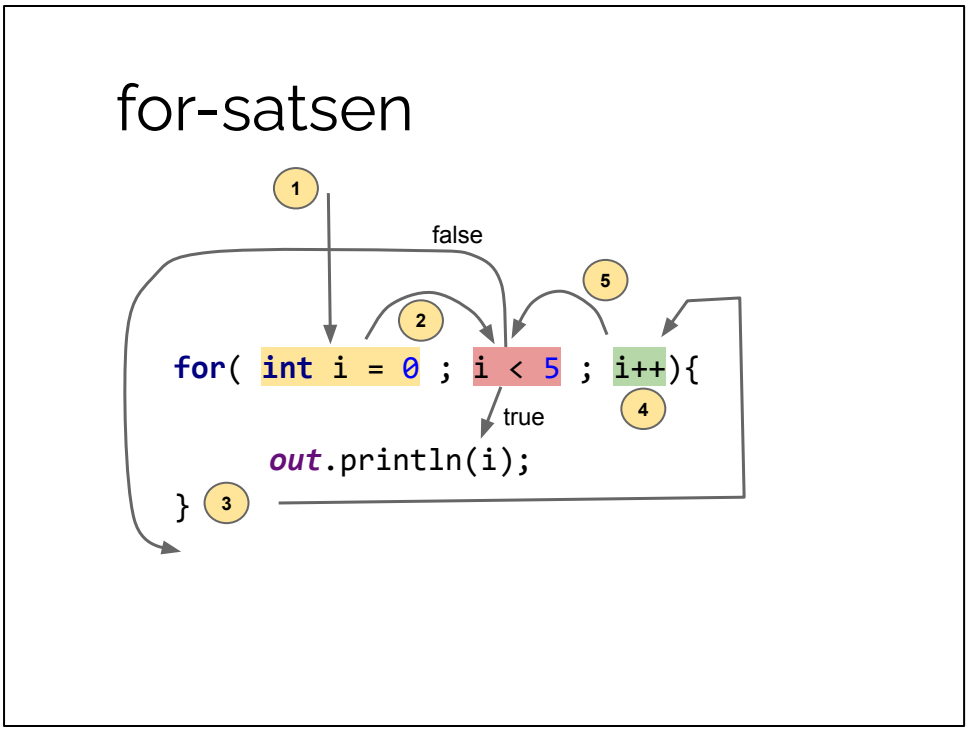
```
int[] points = { 0, 5, 2, 0, 9 };

out.println(points.length);   // 5

// Last value!
out.println(points[points.length-1]); // 9
```


Man kan komma åt längden av en array på ett fördefinierat sätt (inbyggt i språket)

- Genom att skriva: *array-namn*.length



For-satsen är ett annat sätt att skriva iterationer.

En for-loopen består av 3 delar (åtskilda med ";") inom parentesen och ett efterföljande block

- Första delen av parentesen är en initieringsdel (körs bara en gång, första gången). Använda för att deklarera och initiera loop-variabeln. Variabelns synlighetsområde är bara inom loopen (parentesen och blocket efter)
- Andra delen är villkoret
- Sista delen ändrar loop-variabeln. Denna del körs automatiskt sist i loopen trots att den står på första raden.

Analys av koden i bilden

1. Räknaren (variabeln i) för antal varv deklareras och initieras
2. Villkoret evalueras
    a. Om sant
        - Blocket efter parentesen exekveras
        - När slutparentes i blocket nås sker ett hopp till sista delen av parentesen (3)
        - Räknaren ökas/minskas (4)
        - Därefter evalueras villkoret igen (5)
    b. Om falsk

- Hela blocket hoppas över

OBS!

- Skall vara “;” mellan delarna i parentesen annars konstiga fel
- Som tidigare: Undvik att förändra räknaren i loopen (den skall bara räknas upp/ner i sista delen)

while eller for?

compute the largest power of 2 less than or equal to n	`int power = 1;`
	`while (power <= n/2)`
	`  power = 2*power;`
	`System.out.println(power);`

compute a finite sum (1 + 2 + … + n)	`int sum = 0;`
	`for (int i = 1; i <= n; i++)`
	`  sum += i;`
	`System.out.println(sum);`

10

I Java är while och for logiskt utbytbara, vi kan klara oss med t.ex. bara while dock ...

Vi gör följande:

- while-satsen betecknar konceptet upprepa-tills … vi vet inte på förhand hur många gånger vi skall upprepa
- Ibland vet man på förhand hur många gånger man skall upprepa, i dessa fall använder vi for-satsen (for-loopen)

Traversering av Array:er

```
int[] arr = { 1, 2, 3, 4, 5, 6 };


for( int i = 0 ; i < arr.length ; i++){
   out.print(arr[i]);    //123456
}


for( int i = arr.length - 1 ; i >= 0 ; i--){
   out.print(arr[i]);   //654321
}
```

11

Traversering

- Ofta vill man **traversera** (genomlöpa) en array d.v.s. komma åt alla variabler i tur och ordning
  - Från vänster till höger eller tvärtom.
- Traversering görs vanligen med en for-sats (eftersom vi vet längden = antal varv)
- Som villkor använder man:
  - i < array.length (vänster till höger) D.v.s. i skall vara strikt mindre än längden eftersom sista index är ett mindre än längden
  - eller i >= 0 (höger till vänster), större eller lika med eftersom första index är 0.

Inläsning till Array

```
String[] names = new String[3];

out.print("Input 3 names (enter after each) > ");
for (int i = 0; i < names.length; i++) {
   names[i] = sc.nextLine();
}
```


Måste använda en loop.

- Finns ingen genväg

Mer om Initiering

```
int[] points1;    // Declare arrays variables
int[] points2;

// Use variable and initialize array, must use new-operator
// because initialization not done at declaration
points1 = new int[]{2, 9, 0, 1, -4};

// No value list, must supply length (3), default values 0
points2 = new int[3];  // [0, 0, 0]
points2 = new int[points1.length]; // Or same length

// Same for strings
String[] names;
names = new String[]{“Otto”, “Fia”,“Pelle”, “Siv”};
```

13

Man kan skapa och initiera arrayer på flera platser i programmet (inte bara vid variabeldeklarationen)

För att skapa och initiera en array på någon annan plats än vid deklarationen

- Använd operatorn **new** + array-typen + [ ] + en lista med värden.
- Om ingen lista med värden anges måste man ange ett värde för arrayens längd inom hakparenteserna
    - Variablerna får då förbestämda värden beroende på typ, int ger t.ex. 0.

Ändra storlek för Arrayer

```java
// Would like to delete last element from this array
int[] arr1 = {1, 2, 3};

// Create new shorter array
int[] arr2 = new int[arr1.length - 1];

// Copy values
arr2[0] = arr1[0];
arr2[1] = arr1[1];

out.println(Arrays.toString(arr2));
```


Går inte att ändra storlek.

- Måste skapa en ny array och kopiera över värden.