Objekt-Orienteret Programmering

Arrays

Aslak Johansen asjo@mmmi.sdu.dk
Peter Nellemann pmn@mmmi.sdu.dk

September 13, 2023

SDU

Part 0:
Resumé

Resumé ▷ Variable

Variable er navne i vores kildekode som vi bruger til at referere til værdier.

```
1    int lewis_carroll;
2    int douglas_adams = 42;
3    lewis_carroll = 10; ⇐
```

Namespace:

`douglas_adams`
`lewis_carroll`

Bindinger:

`lewis_carroll`	$\mapsto$	10
`douglas_adams`	$\mapsto$	42

Type	*Størrelse*	*Beskrivelse*	*Default Værdi*
`boolean`	? bit	Sandhedsværdi	false
`byte`	8 bit	Heltal	0
`short`	16 bit	Heltal	0
`int`	32 bit	Heltal	0
`long`	64 bit	Heltal	0L
`float`	32 bit	Kommatal	0.0f
`double`	64 bit	Kommatal	0.0d
`char`	16 bit	Tegn	\u0000

Resumé ▷ Primitive Typer

En primitiv datatype indeholder en enkelt værdi.

Exampler på heltal: `1`, `2`, `100`, `-1`

Exampler på kommatal: `1.0`, `2.7`, `3.14`, `1.4e-3`

Exampler på booleans: `true`, `false`

Det giver ikke meget mening at dele disse op i mindre dele.

Part 1:
Arrays

En kompleks type kan indeholde flere værdier.

Af og til har vi brug for typer som kan indeholde mere end én værdi.

- For at samle sekvenser af data af en bestemt type i én struktur.
  - Fx heltal.
- For at skabe typer som er beskrevet ved mere end én (type af) dataværdi.
  - Fx kunne en `Person` bestå af en `int` for fødselsår og en `String` for navn.

Disse typer kalder vi under ét for *komplekse typer*.

I dag skal vi se på en datatype der repræsenterer sekvenser af data, der har samme type, i én struktur. Den datatype hedder "array".

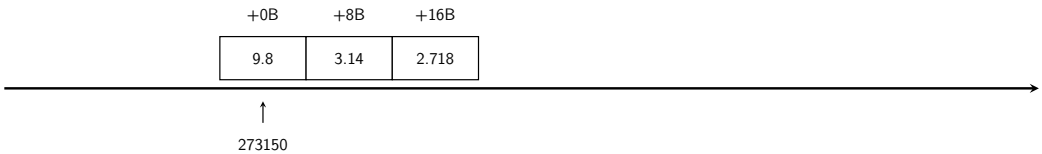
Arrays ▷ Komplekse Typer i Hukommelsen

En computer kan kun arbejde på én værdi af gangen*.

Derfor refererer en variabel af en kompleks type til den adresse i hukommelsen hvor *samlingen* af værdier ligger.

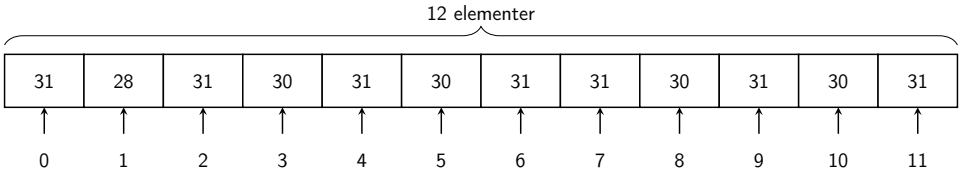
- Værdien af en komplekst typed variabel er derfor en adresse, **ikke** den logiske samling af værdier.

$a \mapsto 42$ (primitiv type: `int`)
$b \mapsto 273150$ (kompleks type: 3 `double`)



Arrays ▷ Definition

*Et array er en **data struktur** som indeholder en sekvens af andre stykker data, **alle af den samme bestemte type**.*



Elementerne er lagt fortløbende ud i hukommelsen.

Elementerne tilgås via et **indeks**.

Arrays indekseres **startende fra 0!**

Arrays ▷ Problemer

When you get the array index wrong

Arrays ▷ Hvorfor Arrays?

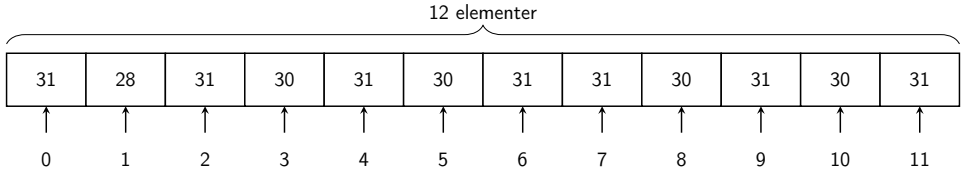
Hvorfor bruge arrays?

- Intet behov for separate variable for associerede værdier.
- Antallet af elementer kan være ukendt ved programmets start.
  - "Hvor mange studerende er der i lokalet lige nu?"
- Evnen til let at gentage handlinger over alle elementer med loops.

Part 2:
Anvendelse

Array Anvendelse ▷ Erklæring og Initialisering

Et array der indeholder en sekvens af `int` værdier kaldes et "int array".



Erklæret ved hjælp af:

```
int[] months;
```

Initialiseret ved hjælp af:

```
months = new int[12];
```

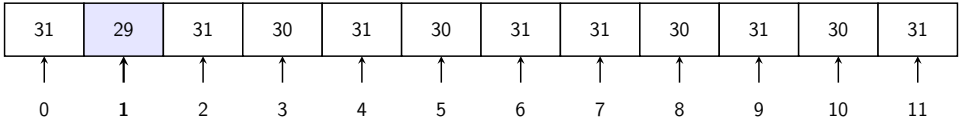
eller – hvis man ønsker selv at specificere startværdier – ved hjælp af:

```
months = {31,28,31,30,31,30,31,31,30,31,30,31};
```

Størrelsen af et array afgøres på initialiseringstidspunktet, og den kan ikke ændres efterfølgende.

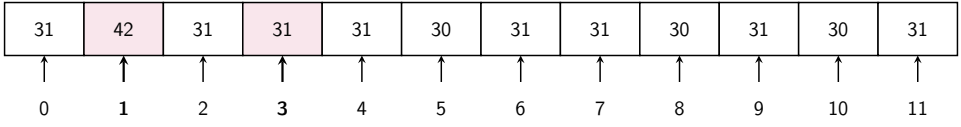
Array Anvendelse ▷ Manipulation

Elementerne af et array kan tilgås og manipuleres via det enkelte elements indeks i sekvensen.



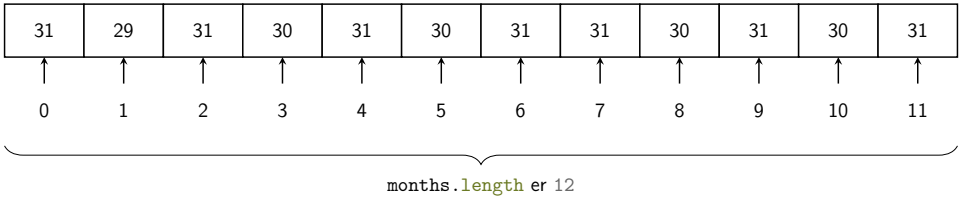
Eksempler:

```
months[1] = 42;
months[3] = months[0];
```



Array Anvendelse ▷ Array Længde

Længden af et array kan aflæses af arrayets **length "attribut"**.




```
int arrayLength = months.length;
System.out.println(arrayLength);
```

```
12
```

Array Anvendelse ▷ Typer i Arrays

Arrays kan indeholde værdier af en vilkårlig type, men alle værdier i et specifikt array har den samme type.

Når et array initialiseres, allokeres der plads til samtlige elementer og disse vil blive tildelt en værdi der afhænger af element typen:

- ***int-array*** Alle elementer i arrayet starter med værdien `0`.
- ***String-array*** Alle elementer i arrayet starter med værdien `null`.
- ***boolean-array*** Alle elementer starter med værdien `false`.

Hvorfor?

Fordi disse er *default* værdierne for typerne.

Part 3:
Traversering af Arrays

Traversering af Arrays ▷ Iteration over et String-Array

Eksempel:

```
String[] stringArray = {
  "This", "is", "an", "example", "of", "a", "String", "array"
};

for (int i = 0; i < stringArray.length; i++) {
  System.out.print(stringArray[i] + " ");
}
```

Output:

```
This is an example of a String array
```

Traversering af Arrays ▷ Iteration over et int-Array

Eksempel:

```
int[] months = {31,28,31,30,31,30,31,31,30,31,30,31};

for (int i=0 ; i<months.length ; i++) {
  System.out.println("Month "+(i+1)+" has "+months[i]+" days");
}
```

Output:

```
Month 1 has 31 days
Month 2 has 28 days
Month 3 has 31 days
Month 4 has 30 days
...
Month 12 has 31 days
```

Traversering af Arrays ▷ Iteration med Foreach

Eksempel:

```
int[] months = {31,28,31,30,31,30,31,31,30,31,30,31};

for (int monthLength : months) {
  System.out.println(monthLength + " days");
}
```

Output:

```
31 days
28 days
31 days
30 days
...
31 days
```

Traversering af Arrays ▷ Større Eksempel

```
double[] doubleArray = new double[12];
double total = 0;
double average;
int i;

// insert code to fill up doubleArray

for (i = 0; doubleArray.length; i++) {
  total += doubleArray[i];
}

average = total / doubleArray.length;
```

Part 4:
Reference Manipulation

Reference Manipulation

Vi afgør dynamisk hvad en variabel refererer til.

```
int[] monthsNormal = {31,28,31,30,31,30,31,31,30,31,30,31}
int[] monthsLeap   = {31,29,31,30,31,30,31,31,30,31,30,31}

for (int i=0 ; i<2020 ; i++) {
  int[] months = monthsNormal;
  if(i%4==0){
    months = monthsLeap;
  }
  System.out.println("In year "+i+" February is "+months[1]+
                     " days long");
}
```

Reference Manipulation

Vi afgør dynamisk hvad en variabel refererer til.

```
int[] monthsNormal = {31,28,31,30,31,30,31,31,30,31,30,31}
int[] monthsLeap   = {31,29,31,30,31,30,31,31,30,31,30,31}

for (int i=0 ; i<2020 ; i++) {
  int[] months = (i%4==0 ? monthsLeap : monthsNormal);
  System.out.println("In year "+i+" February is "+months[1]+
                     " days long");
}
```

Part 5:
Arrays af Arrays

Arrays af Arrays ▷ Introduktion

Vi har set at man kan erklære arrays således:

```
int[]     is = {1,2,3,4};
double[]  ds = {1.0,2.0,3.0,4.0};
boolean[] bs = {false,true,false,true};
```

Hvis man kan lave et array af en vilkårlig type, kan man så også lave et array af arrays?

Måske sådan her?

```
int[][] as = {{1,2,3,4}, {2,3,4,5}, {3,4,5,6}, {4,5,6,7}};
```

Og svaret er: Ja!

Et array er også en (kompleks) type, og arrays kan indeholde både simple og komplekse typer.

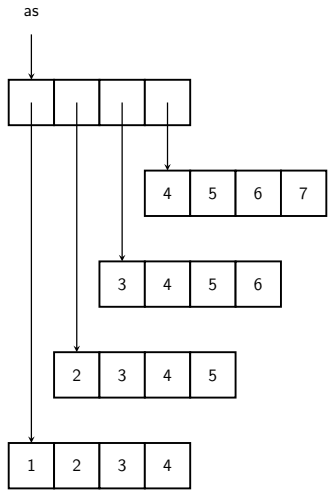
Arrays af Arrays ▷ Layout i Hukommelsen

Eksempel:

```
int[][] as = {
  {1,2,3,4},
  {2,3,4,5},
  {3,4,5,6},
  {4,5,6,7}
};
```

For at manipulere en værdi skal vi først slå op i det yderste array, og dernæst det inderste.

```
System.out.println(as[2]);
System.out.println(as[2][3]);
```



Arrays af Arrays ▷ Tabeller

Arrays af arrays kan (ofte) ses som tabeller.

```
int[][] array = new int[4][6];

// fill out array

intArray[3][2] == 17
```

	0	1	2	3	4	5
0	13	7	33	54	-5	-1
1	-3	0	8	42	18	0
2	44	78	90	79	-5	72
3	43	-6	17	100	1	-12

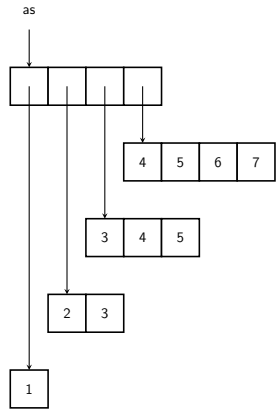
Arrays af Arrays ▷ Eksempel

```
double[][] profit = new double[25][12];
double totalProfit; // Company's total profit in 2014.
int store, month; // variables for looping through stores and months

// code for filling out profit

for ( store=0 ; store<25 ; store++ ) {
  for ( month=0 ; month<12 ; month++ ) {
    totalProfit += profit[store][month];
  }
}
```

Arrays af Arrays ▷ Jagged Arrays

Alle indre arrays behøver ikke at have samme længde: Pas på når du indekserer!

```
int[][] as = {
  {1},
  {2,3},
  {3,4,5},
  {4,5,6,7}
};

for (int y=0 ; y<as.length ; y++) {
  for (int x=0 ; x<as[0].length ; x++) {
    System.out.println("as["+y+"]["+x+
                       "] = "+as[y][x]);
  }
}
```

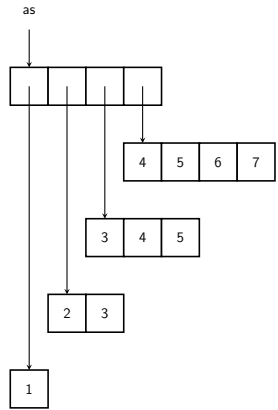


Nope!

Arrays af Arrays ▷ Jagged Arrays

Alle indre arrays behøver ikke at have samme længde: Pas på når du indekserer!

```
int[][] as = {
  {1},
  {2,3},
  {3,4,5},
  {4,5,6,7}
};

for (int y=0 ; y<as.length ; y++) {
  for (int x=0 ; x<as[1].length ; x++) {
    System.out.println("as["+y+"]["+x+
                       "] = "+as[y][x]);
  }
}
```

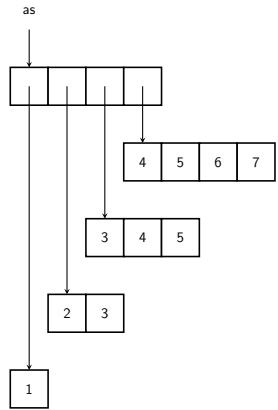


Nope!

Arrays af Arrays ▷ Jagged Arrays

Alle indre arrays behøver ikke at have samme længde: Pas på når du indekserer!

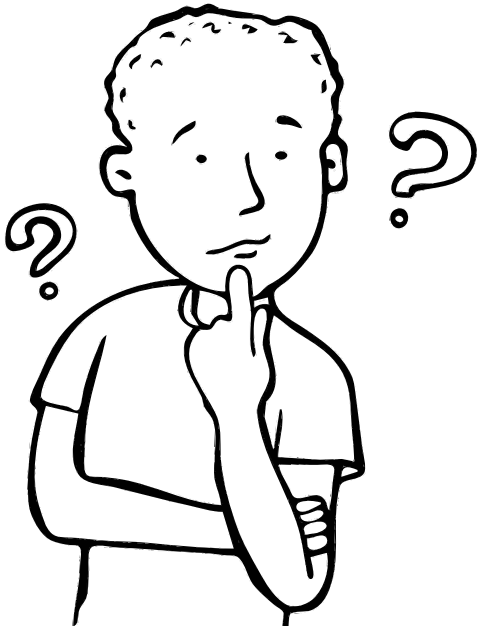
```
int[][] as = {
  {1},
  {2,3},
  {3,4,5},
  {4,5,6,7}
};

for (int y=0 ; y<as.length ; y++) {
  for (int x=0 ; x<as[y].length ; x++) {
    System.out.println("as["+y+"]["+x+
                       "] = "+as[y][x]);
  }
}
```



Yay!

Fristelsen

Questions?



https://openclipart.org/detail/236687/boy-thinking-of-question