ex5matrix

Joachim von Hacht

Matriser

```
// Declare and initialize (instantiated automatically)
int[][] m = {       // An array of arrays
       { 1,2,3,},          // m[0]
       { 4,5,6,},          // m[1]
       { 7,8,9,},          // m[2]
   };

int[][] m;          // Just a single variable
m = new int[3][3];  // Instantiate object, default values 0
m = new int[2][2]{  // Instantiation and initialization
    {11, 12},
    {21, 22}
};
```

2

I Java kan man ha arrayer av godtycklig dimensioner (< 255)

Tvådimensionella array:er är vanligt

- Tekniskt en array av array:er
- Vi kallar 2D arrayer för matriser
- En matrisvariabel deklareras med dubbla [ ] -parenteser efter typen
  - Först index anger vilken array (rad), …
  - … andra anger, element i aktuell array (kolumn)
  - Rad och kolumnindex börjar som vanligt på 0.
- Vi använder normalt bara rektangulära matriser i laborationerna (man kan använda “ragged 2D arrays”)

Indexering på Matriser

```
int[][] m = {
    { 1,2,3,},
    { 4,5,6,},
    { 7,8,9,}
};

// [row][col]
m[0][2] = 99;  // {{ 1,2,99,},{ 4,5,6,},{ 7,8,9,}}
m[2][1] = m[0][2]; // {{ 1,2,99,},{ 4,5,6,},{ 7,99,9,}}

// Exception
m[0][3] = 45;
```


För att komma åt enskilda element används som tidigare indexering (men nu med två index)

- Tänk rad, kolumn!
- Undvik att tänka x och y
    - Kan bli förvirrade, speciellt blanda inte x/y och rad/kolumn.

Traversera Matris

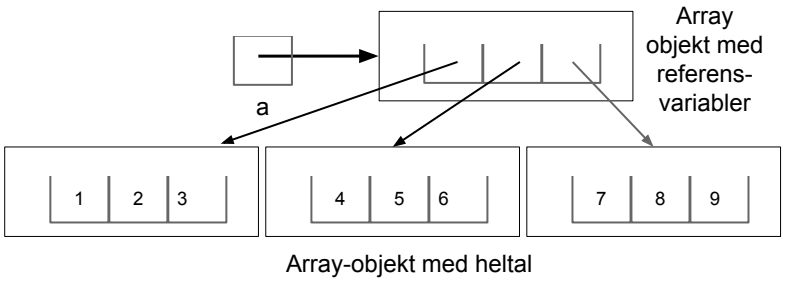
```
int[][] m = { { 1,2,3,}, { 4,5,6,}, { 7,8,9,} };

// Traverse (will work for ragged matrices)
for(int row = 0 ; row < m.length ; row++){
   for( int col = 0 ; col < m[row].length; col++){
       out.print( m[row][col]);
   }
   out.println();
}
```


Traversering använder nästlade for-loopar och length.

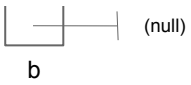
Matris-Objekt

```
int[][] a = { { 1,2,3,}, { 4,5,6,}, { 7,8,9,} };
```



Array-objekt med heltal

```
// NO matrix object
int[][] b = null;
```



En deklaration av en matrisvariabel ger, på samma sätt som en array, en referensvariabel

- Objektet referensen pekar på innehåller i sin tur variabler av referenstyp
- Som tidigare anger vi null om variabeln inte skall peka på något objekt.
- Tilldelning och likhet som för arrayer.

Matrisparametrar

```
void transpose(int[][] m) {
   for (int row = 0; row < m.length; row++) {
       for (int col = row + 1; col < m[row].length; col++) {
           int tmp = m[row][col];
           m[row][col] = m[col][row];
           m[col][row] = tmp;
       }
   }
}
```

Transponerar matrisen, rader blir kolumner.

- Parametrar och returvärden som för arrayer.

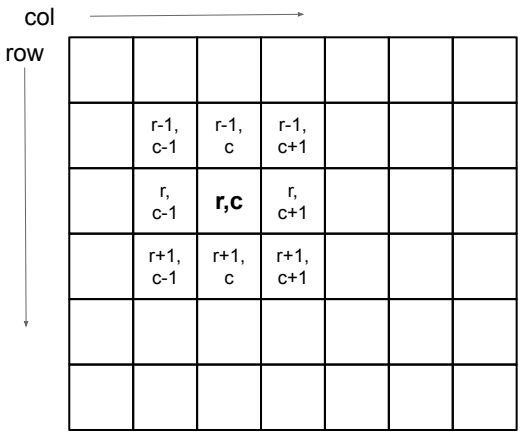
Testning av Matriser

```
int[][] m = {
        {0, 1, 0,},
        {1, 1, 1,},
        {0, 1, 0,},
};

out.println(Arrays.toString(m[0]).equals("[0, 1, 0]"));
// m[1] false!
out.println(Arrays.toString(m[1]).equals("[1, 0, 1]"));
out.println(Arrays.toString(m[2]).equals("[0, 1, 0]"));
```


Ungefär som för arrayer fast vi får skriva ut alla rader.

Matris som Datastruktur

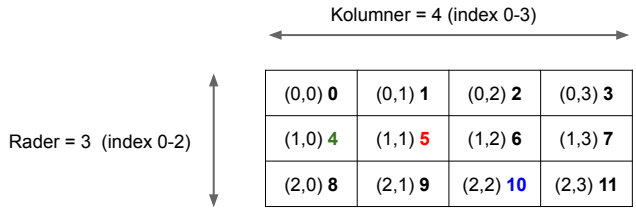


8

En matris ger oss också en datastruktur

- Fler möjligheter: vänster/höger/över/under/snett över/snett under ...

Byte mellan Array och Matris



Array -> Matris:

radindex = index / kolumner (5 / 4 = 1)

kolumnindex = index % kolumner (10 % 4 = 2)

Matris -> Array:

index = radindex * kolumner + kolumnindex (1 * 4 + 0 = 4)

9

Omvandling mellan array och en matris kan behövas

- Datan är oförändrad det är bara strukturen som ändras.
- Ibland lättare att arbeta med array-format …
- … ibland lättare med matris.