sheets Programmeren 1 — Java

college 6, Walter Kosters

De sheets zijn gebaseerd op met name hoofdstuk 13 en 14 van:
D. Bell en M. Parr, Java voor studenten,
Prentice Hall, 2002

`http://www.liacs.nl/home/kosters/java/`

Java — **arrays**

Je kunt eenvoudig een geordende rij (*array*) met getallen maken in Java:

```
int[ ] serie = new int[5];
```

Nu hebben we de volgende vijf `int`'s: `serie[0]`, `serie[1]`, `serie[2]`, `serie[3]` en `serie[4]`. Het getal tussen `[ ]` heet een *array-index*, terwijl `serie[3]` een *array-element* is.

Let er goed op dat de array-indices altijd binnen de perken blijven. Zo bestaat in bovenstaand voorbeeld `serie[5]` niet!

Als je een array doorgeeft aan een functie, kan het array wel degelijk gaan veranderen — dit in tegenstelling tot bijvoorbeeld `int`'s!

Java — **afdrukken**

Stel dat we de inhoud van een array af willen drukken. Dat gaat als volgt:

```
public void drukAf (int[ ] array) {
  int i;
  for ( i = 0; i < array.length; i++ )
    System.out.println (array[i] + " ");
}  // drukAf
public void vulArray (int[ ] A) {
  int i;
  for ( i = 0; i < A.length; i++ )
    A[i] = (int) ( 100 * Math.random ( ) ) + 1;
}  // vulArray
public static void main (String[ ] args) {
  int[ ] ar = new int[30];
  vulArray (ar);
  drukAf (ar);
}  // main
```

Let op de verschillende namen (`A`, `array`, `ar`) die het array kan hebben. Merk ook op dat de functie `vulArray` het array wijzigt!

Java — **afdrukken (vervolg)**

Of in een applet:

```
public void drukAf2 (int[ ] arr, Graphics g) {
  int i; int y = 30;
  for ( i = 0; i < arr.length; i++ ) {
    g.drawString (arr[i] + " ",50,y);
    y = y + 20;
  }  // for
}  // drukAf2
public void paint (Graphics g) {
  int[ ] arr = new int[100];
  ... vul arr als eerder ...
  drukaf2 (arr,g);
}  // paint
```

Je kunt in de body van de `drukAf2`-functie overigens ook `g` "ophalen" via

```
Graphics g = getGraphics ( );
```

en zo "zonder" `paint` tekenen!

Java — **sorteren**

We gaan nu een array oplopend sorteren met behulp van *bubblesort*:

```
public void bubblesort (int[ ] array) {
  int i, j;
  int temp;
  for ( i = 1; i < array.length; i++ )
    for ( j = 0; j < array.length-i; j++ )
      if ( array[j] > array[j+1] ) {
        temp = array[j];
        array[j] = array[j+1];
        array[j+1] = temp;
      }  // if
}  // bubblesort
```

In iedere ronde wisselen we buren die verkeerd om staan, van links naar rechts door het array heen. Zo wordt 3 5 8 2 7 9 1 na de eerste ronde 3 5 2 7 8 1 9: het grootste getal staat nu zeker achteraan.
Aanroep: `bubblesort (serie);`

Java — **zoeken**

En vervolgens willen we weten op welke plek getal `X` in het array staat:

```
public int zoeken (int[ ] array, int X) {
  int i = 0;
  while ( i < array.length ) {
    if ( X == array[i] )  // bingo
      return i;
    i++;
  }  // while
  return -1;  // X niet gevonden
}  // zoeken
```

Dit heet wel *lineair zoeken*.
Aanroep: `nummer = zoeken (serie,42);`

Het kan sneller (met *binair zoeken*) als de rij oplopend gesorteerd is: je kijkt naar het middelste getal, en als dat niet `X` is ga je of links of rechts verder, al naar gelang het middelste getal groter of kleiner dan `X` is.

Java — **de grootste**

En wat is het grootste getal in het array?

```
public int grootste (int[ ] array) {
  int i;
  int groot = array[0];
  for ( i = 1; i < array.length; i++ ) {
    if ( array[i] > groot )
      groot = array[i];
  }  // for
  return groot;
}  // grootste
```

Probeer zelf eens het op-een-na-grootste array-element te vinden.

Java — **de grootste (vervolg)**

Maar waar staat dat grootste getal in het array:

```
public int plekgrootste (int[ ] array) {
  int i;
  int index = 0;
  int groot = array[0];
  for ( i = 1; i < array.length; i++ ) {
    if ( array[i] > groot ) {
      groot = array[i];
      index = i;
    }  // if
  }  // for
  return index;
}  // plekgrootste
```

Als je deze functie zo aanpast dat je vanaf een bepaalde plek de grootste zoekt, kun je een eenvoudige sorteermethode maken door herhaald de grootste "van de rest" op te sporen en die te wisselen met het "achterste" element.

Java — **tellen**

En hoe vaak komt `X` in een 2-dimensionaal array (matrix) voor?

```
public int tellen (int[ ][ ] array, int X) {
  int i, j;
  int teller = 0;
  for ( i = 0; i < array.length; i++ )
    for ( j = 0; j < array[0].length; j++ )
      if ( X == array[i][j] )
        teller++;
  return teller;
}  // tellen
```

Let er op dat hier `array.length` het aantal rijen van `array` is, en `array[0].length` het aantal kolommen van `array`.

```
2  3  4  5  2        2 rijen, 5 kolommen
3  5  2  7  8        2 komt drie keer voor
```

Java — **exceptions**

Hoe vang je fouten op? Hoe kun je bijvoorbeeld controleren of iemand in een tekstveld daadwerkelijk een getal invoert? Dat verloopt via *exceptions*:

```
public void actionPerformed (
    ActionEvent event) {
  try {
    int getal = Integer.parseInt (
        tekstVeld.getText ( ));
    getal = 2 * getal;
    anderVeld.setText ("Verdubbeld " + getal);
  }  // try
  catch (NumberFormatException e) {
    anderVeld.setText ("Fout ... nog eens: ");
  }  // catch
}  // actionPerformed
```

Je kunt exceptions ook nog doorgeven met behulp van `throws`-commando's.

Java — **voor de zekerheid**

Voor de zekerheid nog één keer de twee soorten functies, dit maal voor machtsverheffen:
$2^{10} = 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 = 1024$.

```
public int macht (int x, int n) {
  int res = 1; int i;
  for ( i = 1; i <= n; i++ )
    res = res * x;
  return res;
}  // macht
public void macht2 (int x, int n) {
  int res = 1; int i;
  for ( i = 1; i <= n; i++ )
    res = res * x;
  System.out.println (" " + res);
}  // macht2
```

Met aanroepen:

```
int iets = macht(2,10);
System.println (" " + iets);  // 1024
macht2 (2,10);  // ook 1024
```