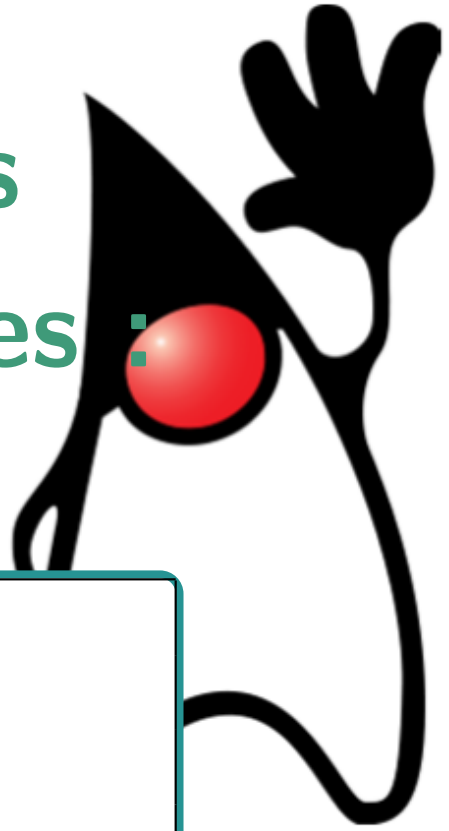
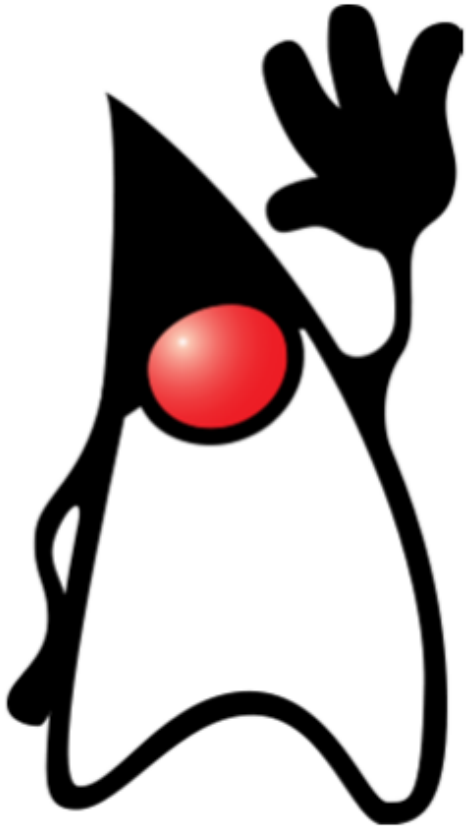
Programming Fundamentals
Arrays in meerdere dimensies :



Klasgroep	1EO-ICT
Opleiding	Bachelor Elektronica-ICT
Theorie	DI: 8:45 - 9:45
	WOE: 11:45 - 12:45
(Werk)Labo	DI: 9:45 - 12:45 + 13:30 - 14:30
	WOE: 13:30 - 15:30 + 15:30 - 17:30
Docent	Katja Verbeeck
Contact	katja.verbeeck@odisee.be

1. Wat is een array?
2. declaratie, instantiatie en initialisatie
3. Een array is een object
4. Arrays als parameters en returntypes
   - Parameters van de main methode
5. foreach vs for



Wat is een array?

- Een array is een datastructuur, deze kan een collectie van variabelen bijhouden en er via 1 variabelenaam naar refereren.
- Een array kan alleen een collectie van homogene data bijhouden, m.a.w. elk element in de array moet van hetzelfde type zijn.
- Een array in Java is geimplementeerd als een object (en moet dus aangemaakt worden via new)
- Een array kan meerdere dimensies hebben (je kan arrays van arrays van arrays ... maken)

Een voorbeeld : lotto

Via 1 variabele kunnen de lotto cijfers bijgehouden worden :

```
int[] trekking;
trekking = new int[6];

for (int idx = 0; idx < 6; idx++) {
    trekking[idx] = (int)(Math.random() * 45) + 1;
}
```

Een voorbeeld : klinkers

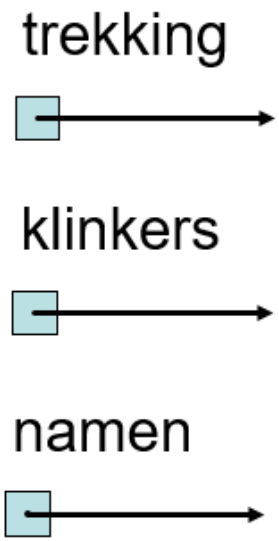
Via 1 variabele kunnen alle klinkers bijgehouden worden :

```
char [] klinkers;
klinkers = new char[5]

klinkers[0] = 'a';
klinkers[1] = 'e';
klinkers[2] = 'i';
klinkers[3] = 'o';
klinkers[4] = 'u';

char[] klinkers = {'a', 'e', 'i', 'o', 'u'};
```

declaratie

Declaratie kent een variabele een type toe. Voor een array doe je dit door het type van de elementen van de array te laten volgen door [ ].
Wanneer een variabele gedeclareerd is, is er echter nog geen plaats aan toegekend in het geheugen!

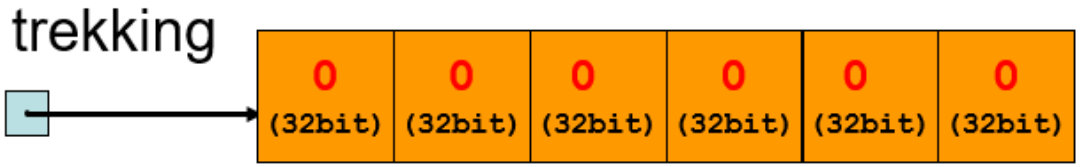
```
//enkele declaraties
int[] trekking;
char[] klinkers;
String[] namen;

System.out.println(trekking);
// trekking wijst op dit
    moment naar de null
    pointer
// compiler geeft error
    omdat er nog niet
    geinitialiseerd is
```



instantiatie

Tijdens deze stap ga je plaats in het geheugen reserveren voor de array aan de hand van de **new** operator. Daarom geef je mee van welk type je elementen zijn en hoeveel elementen je wil aanmaken. Dit moet vooraf gekend zijn ! Elk element in de array wordt standaard ingesteld op een default waarde (volgens type - voor int is dit 0):
Declaratie en instantiatie kan op 1 lijn.

```
int[] trekking = new int[6];
```



lengte van een array

Van zodra een array plaats heeft in het geheugen kan je zijn lengte opvragen. Dit is het aantal elementen waarvoor je plaats reserveerde. Je doet dit met de **length** op te vragen. Let op, dit is geen methodeoproep, maar data (en dus geen haakjes!)

```
int lengte = trekking.length;
System.out.println(trekking.length);
// output naar het scherm : 6

System.out.println(trekking);
//output naar het scherm : [I@3e25a5


// laatste element van de trekking array
int laatste = trekking[5];
int laatste = trekking[lengte - 1]
```

Wat er gebeurt ...

... als je vergeet te instantiëren

```
String[] namen;
System.out.println(namen);
// output naar het scherm : (compiler geeft
   een fout, want namen heeft nog geen
   waarde gekregen )
System.out.println(namen.length);
// output naar het scherm : idem
   compilererror
```

java: variable namen might not have been initialized

initialisatie via for-lus

Maak en vul een array met de getallen 1 tot en met 10

```
int[] getallen = new int[10];
getallen[0] = 1;
getallen[1] = 2;   ...
getallen[9] = 10;

int[] getallen = {1,2,3,4,5,6,7,8,9,10};

// efficienter:
int[] getallen = new int[10];
for (int i = 0; i < getallen.length; i++) {
    getallen[i] = i+1;
}
```

initialisatie via for-lus

Let op! ArrayIndexOutOfBounds ...

```
// fout !
int[] getallen = new int[10];
for (int i = 1; i <= getallen.length; i++) {
    getallen[i] = i;
}
```

initialisatie via initializer syntax

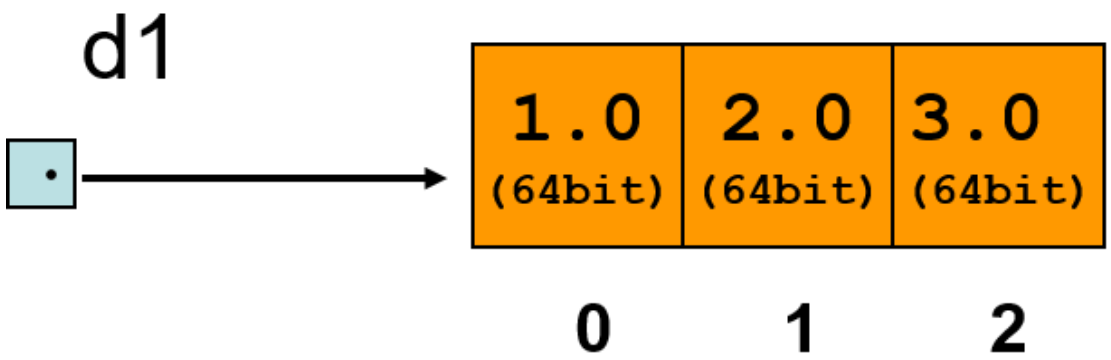
Via initializer syntax kan declaratie, instantiatie en initialisatie gebundeld worden in 1 java instructie.

```
int[] getallen = {1,2,3,4,5,6,7,8,9,10};
char[] klinkers = {'a', 'e', 'i', 'o', 'u'};
int[] priemgetallen = {2, 3, 5, 7, 11, 13, 17, 19};

String[] dagen = {"maandag", "dinsdag", "woensdag",
          "donderdag", "vrijdag", "zaterdag",
              "zondag"};
```

Let op! Je kan dit niet opsplisten, declaratie, initialisatie en instantiatie moet in 1 instructie (op 1 lijn dus) gebeuren.

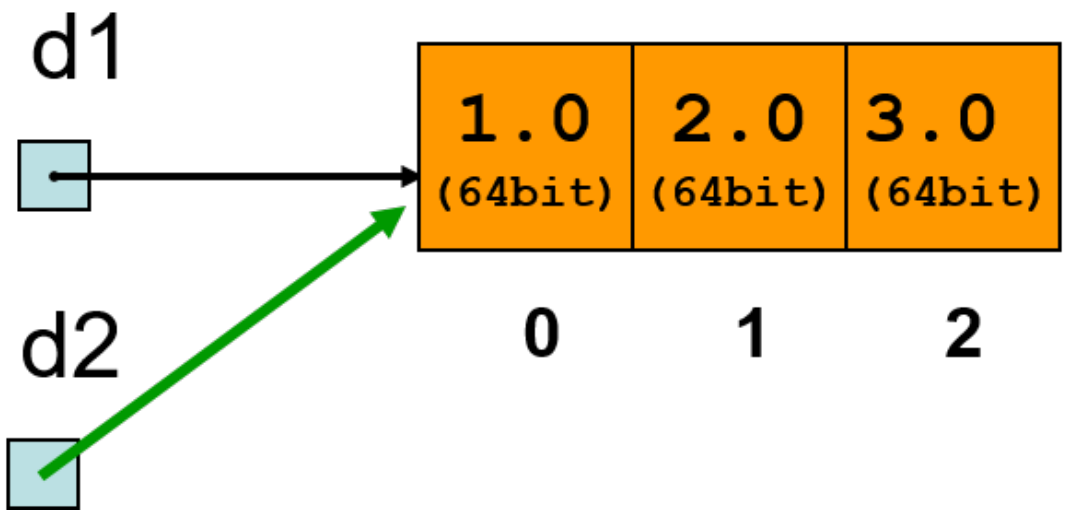
Een array is een object

```
double[] d1 = {1.0, 2.0, 3.0};
```



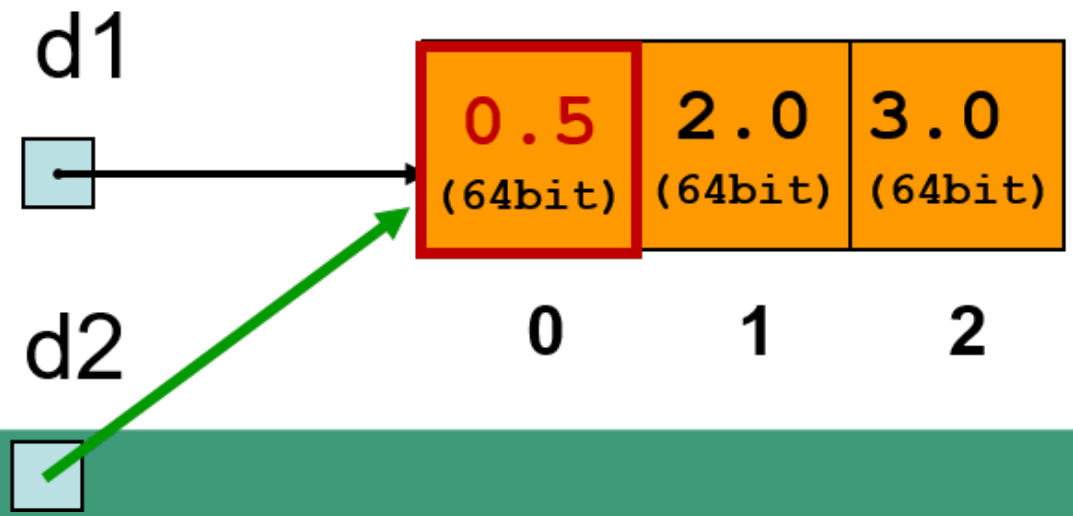
Een array is een object

```
double[] d1 = {1.0, 2.0, 3.0};
double[] d2;
d2 = d1;
```



Een array is een object

```
double[] d1 = {1.0, 2.0, 3.0};
double[] d2;
d2 = d1;
d2[0] /= 2;
System.out.println(d1[0]);
// output naar het scherm : 0.5
```



Methoden en Arrays

Een array is een object type en dus kan je ook arrays meegeven als parameter van een methode (zie main methode!), maar je kan ook een array als resultaat teruggeven van een methode. Op deze manier kan je dus meerdere antwoorden (wel allen van hetzelfde type !) teruggeven als resultaat van een methode.

Een array als parameter van een methode

```
public int geefSom(int[] getallen) {
    int som = 0;

    for (int i = 0; i < getallen.length; i++) {
     som += getallen[i];
    }
    return som;
}
```

Een array als parameter van een methode

```
public boolean zoek(char[] rij, char c) {
    for (int i = 0; i < rij.length; i++) {
        if(rij[i] == c) {
          return true;
        }
    }

    return false;
}
```

Parameters van de main methode

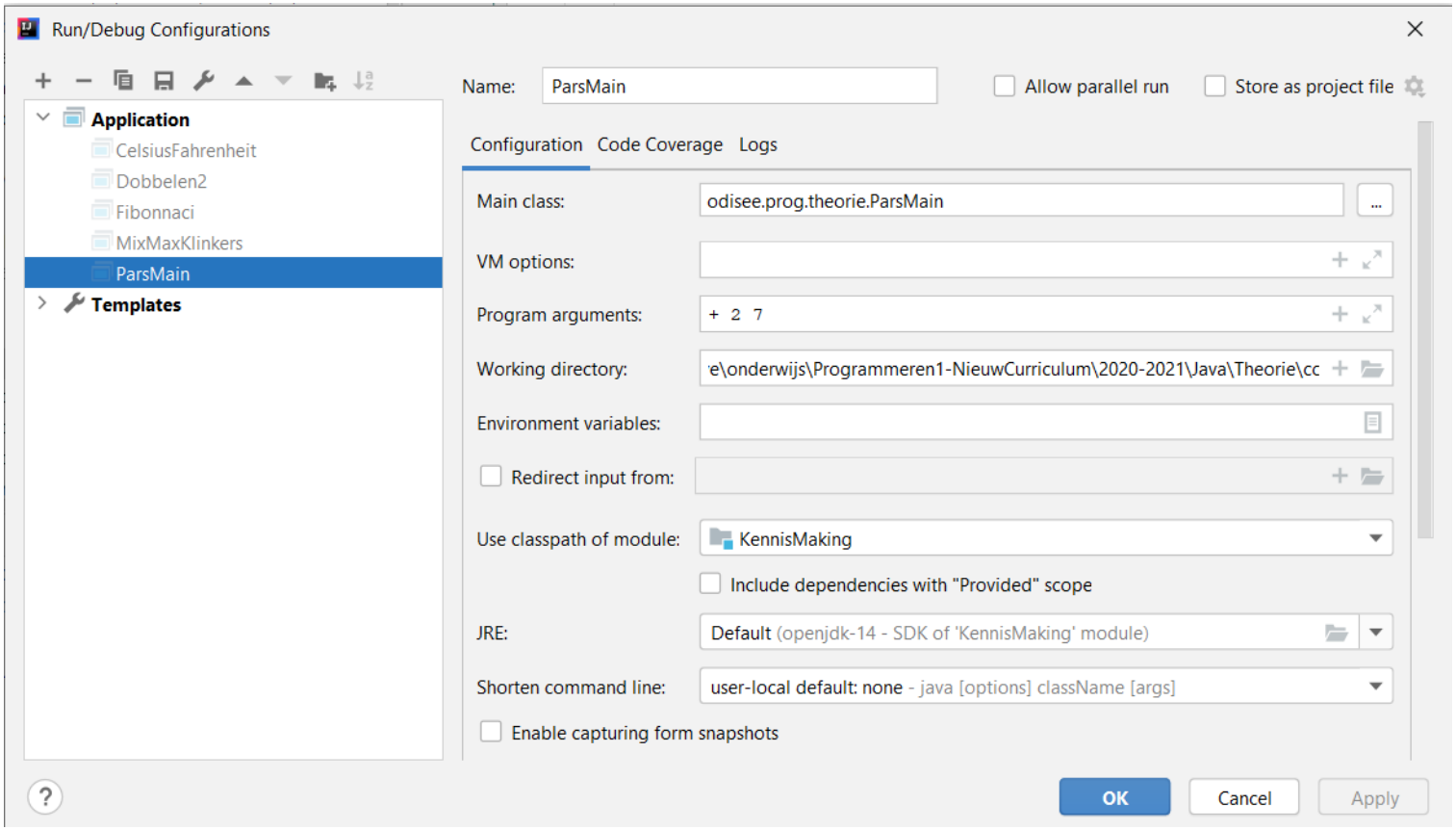
Deze worden doorgegeven via een array : args[]
`public static void main(String[] args)`
In de body van de main kan je deze parameterwaarden (indien er zijn) uitlezen :

- Je kan de lengte van de array opvragen : `args.length`
- Wanneer je weet hoeveel er zijn kan je ze afzonderlijk uitlezen : `args[0]`, `args[1]`, `args[2]` enz
- Deze argumenten zijn van type String maar kunnen altijd omgezet worden naar het eigenlijke primitieve type, bvb. bij getallen : `Integer.parseInt(String s)` of `Double.parseDouble(String s)`

Parameters van de main methode

```
public static void main(String[] args) {

    if (args.length != 3) {
        System.out.println("Gebruik het programma als
            volgt:");
        System.out.println("java.exe <klasse> <operator>
            <eerste getal> <tweede getal> ");
        return;
    }

    char operator = args[0].charAt(0);
    int getal1 = Integer.parseInt(args[1]);
    int getal2 = Integer.parseInt(args[2]);

    int resultaat =
         RekenMachine.doeBewerking(getal1,getal2,operator);
    System.out.println("Het resultaat is: " + resultaat);
}
```

Parameters meegeven via IntelliJ

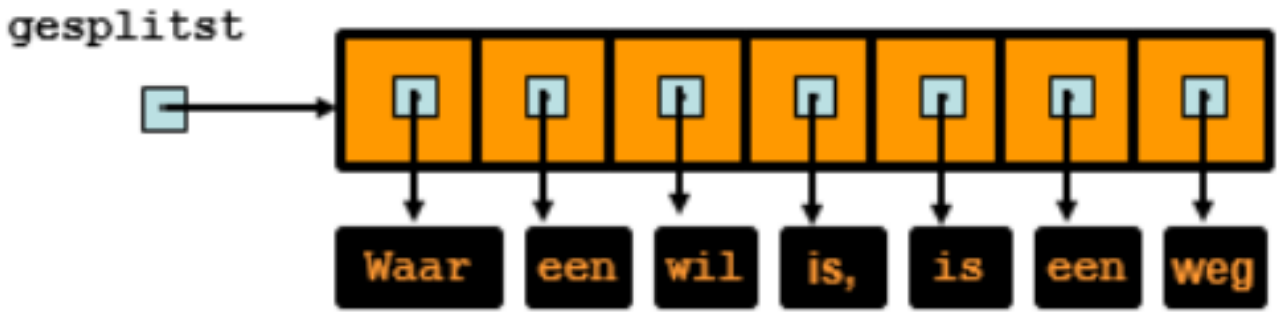


Een array als resultaat van een methode

```
public int[] genereerGetallen(int aantal) {
  int[] getallen = new int[aantal];

  for (int i = 0; i < getallen.length; i++) {
      getallen[i] = i + 1;
   }

   return getallen;
}
```

Een array als resultaat van een methode

```java
// methode split van de klasse String
public String[] split(String s);
String mijnTekst = "Waar een wil is, is een weg";
String[] gesplitst  = mijnTekst.split(" ");
```



Een array als resultaat van een methode

```
String[] gesplitst = mijnTekst.split(", ");
```

for(each)

```
char[] graden = {'A','B','C'};
for(int i = 0; i < graden.length; i++)
    // for i goes from 0 to graden.length
     System.out.print(graden[i]);
    // print graden[i]
```

```
for(char graad : graden)
    //foreach (voor elke) graad in graden
     System.out.print(graad);
     // print de graad
```

Zie vervolg in Java OO Collections