Java – cheat sheet

Zie ook de Mini-java pagina's op de parallel website (http://parallel.vub.ac.be/education/java/miniJava/MiniJavaDoc.html)

Primitieve Types

byte	8-bit geheel getal. -128 tot en met 127
short	16-bit geheel getal. -32768 tot en met +32767
int	32-bit geheel getal. -2^{31} tot en met $2^{31}-1$
long	64-bit geheel getal. -2^{63} tot en met $2^{63}-1$ of 0 tot en met $2^{64}-1$
float	32-bit floating point getal (voor meer info google IEEE 754)
double	64-bit floating point getal
boolean	kan slechts twee waarden aannemen (**true** of **false**)
char	16-bit unicode karakter (1 letter of speciaal karakter) bv 'a', 'b', ... (1 letter en enkele quotes)
String	0 of meer 16-bit unicode karakters, bv "Hallo wereld!" (dubbele quotes)

Om in Java een variabele aan te maken moet je altijd aangeven van welk type de variabele is. Dit type is vast.

```
int mijnInteger; // maak een variabele van type int aan
mijnInteger = 1; // geef mijnInteger de waarde 1
int tweedeInteger = 2; // maak een int variabele aan en stel ze onmiddelijk gelijk aan 2
tweedeInteger = 3.1415; // Type mismatch: cannot convert from double to int
```

Objecten

Alles in Java is een Object (ook de primitieve datatypes). Om een variabele aan te maken moet je normaal gezien altijd het keyword **new** gebruiken. De enige uitzonderingen hierop zijn de primitieve datatypes.

!!!
Het aanmaken van een variabele zonder er ook effectief een object aan toe te wijzen kan leiden tot een *NullPointerException*. Dit is een veel voorkomende fout bij beginnende programmeurs. Wanneer je geen waarde toekent aan de variabele zal er standaard **null** in staan. De variabele wijst dus letterlijk nergens naar, dus je kan er niks mee doen.
!!!

Arrays

Een array is een object dat een vast aantal waarden van één enkel type kan bevatten. De lengte van de array wordt bepaald wanneer hij wordt aangemaakt en deze lengte kan achteraf niet veranderd worden.

Elke waarde in een array wordt *element* genoemd en elk element kan worden geaddresseerd door zijn numerieke *index*. Geldige indexen gaan van *0 tot en met de lengte van de array-1*. Vragen naar een index buiten deze range zal een *IndexOutOfBoundsException* veroorzaken.

```
int[] mijnArray; // maak variabele aan van type "array van integers"
mijnArray = new int[3]; // maak een nieuwe int-array van lengte 3 en stop die in mijnArray
mijnArray[0] = 5;
mijnArray[1] = 6;
mijnArray[2] = 7;
mijnArray[3] = 8; // runtime error: IndexOutOfBoundsException
```

Operatoren

=	toekenning, geeft een waarde aan een variabele

+	optelling (wordt ook gebruikt om Strings aaneen te plakken)
-	aftrekking
*	vermenigvuldiging
/	deling (de deling wordt uitgevoerd met de datatypes van de getallen waarop ze werkt) 7/2 = 3 (int / int = int) 7.0/2 = 3.0 (double / int = double)
%	modulo operator (rest bij deling)
^	XOR operator, je zal die niet snel nodig hebben. Indien je dit nodig hebt, zoek dan eens op Google wat het doet, maar onthoud alvast dat deze operatie **GEEN machtverheffing** is.

Unaire operatoren (op 1 variabele):

+	Geeft een positieve waarde aan (maar dat is standaard al zo)
-	Maakt een expressie negatief: bv -1 of -(5+7)
++	Increment operator, vermeerdert een waarde met 1
--	Decrement operator, verminderd een waarde met 1
!	Logisch complement, keert de waarde van een **boolean** om

Vergelijkings operatoren

==	gelijkheid
!=	ongelijkheid
>	groter dan
>=	groter dan of gelijk aan
<	kleiner dan
<=	kleiner dan of gelijk aan

Conditionele operatoren

&&	logische AND (twee expressies moeten waar zijn)
||	logische OR (een of beide expressies moeten waar zijn)

Expressies, statements, blokken

Een expressie is een combinatie van variabelen en operatoren die iets als resultaat heeft.
Bijvoorbeeld: 1+2, “Hallo”+”Wereld”, 7/2, etc. zijn expressies.
Een statement is een “zin” of een “commando” en eindigt altijd met een puntkomma “;”.
Dat kan gaan over het maken van een nieuwe variabele, het uitvoeren van een functie, etc.

Enkele voorbeelden van statements:

```
int a = 1;
a++;
System.out.println("Hallo wereld");
double[] mijnArrayVanDoubles = new double[10]:
double[0] = 1;
```

Een blok is een groep van 0 of meerdere statements omringd door gekrulde haakjes { }. Blokken worden gebruikt voor control-flow (if-then-else en loops).

Control flow

if-then-else

```
String resultaat;
if(score > 10){
   resultaat = “geslaagd”;
}else{
   resultaat =  “niet geslaagd”;
}
```

while-loop

```
int i = 3;
while( i >= 0 ){
   System.out.println("i = " + i);
   i--;
}
```

for-loop

```
int[] mijnArray = {1,2,3,4,5};
for (int idx =  0; idx < mijnArray.length; idx++) {
   System.out.println(mijnArray[idx]);
}
```

In sommige gevallen kan je ook een verkorte for-loop gebruiken:

```
int[] mijnArray = {1,2,3,4,5};
for (int i : mijnArray) { // lees als voor elke integer i in mijnArray
   System.out.println(i);
}
```

Object oriented programming (classes & objects)

Class vs object

- Een klasse is een verzameling van data (variabelen) en functies/methods en vormt een blauwdruk/bouwplan.
- Een object is een instantie (instance) van een klasse.

Stel dat je een klasse auto hebt:

```
class Auto{
   String merk;
   int aantalDeuren;
   int benzine;
   int maxBenzine;

   void geefGas();
   void afremmen();
}
```

De klasse bevat variabelen die een auto kunnen beschrijven en een aantal methodes die functionaliteit implementeren dat een auto zou moeten hebben.

Vanuit die klasse zou je dan verschillende objecten kunnen aanmalen met het keywoord **new**. Zo zou je een auto van het merk "Ford" kunnen aanmaken met een benzinetank van 45 liter of een "BWM" met een benzinetank van 42 liter, etc.

Inheritance

Wanneer je verschillende klassen kan het zijn dat je voor veel klassen dezelfde variabelen en functies aan het gebruiken (of zelf copy-pasten) bent. Overerving (inheritance) is een mechanisme dat toelaat om functionaliteit in een parent-klasse te definiëren en andere klassen die deze parent dan extenden nemen dan automatisch al die variabelen en methodes over.

Stel dat je een meetkunde programma zou schrijven, dan zou je een volgende opdeling kunnen maken. Meetkundige figuren hebben doorgaans een oppervlakte en omtrek. Ruimtefiguren hebben daarboven ook nog een volume. Vervolgens heb je dan ook enkele specifieke figuren zoals Cirkels, Rechhoek etc. die hun eigen voorstelling hebben (straal, hoogte, breedte) en hun eigen formules om daaruit de oppervlakte en omtrek te berekenen. Maar het blijven wel allemaal VlakkeFiguur objecten dus voor de 'buitenwereld' zien ze er qua gedrag/interface gelijkaardig uit.

```
class Figuur {..}
class VlakkeFiguur extends Figuur {...}
class Cirkel extends VlakkeFiguur {...}
class Rechthoek extends VlakkeFiguur {...}
class Veelhoek extends VlakkeFiguur {...}
class RuimteFiguur extends Figuur {...}
class Sfeer extends RuimteFiguur {...}
```