Java traineeship

Flow control

Leerdoelen

- Het aanmaken en gebruik van if, if-else, ternary en switch statements
- Het aanmaken en gebruik van loops: while, do-while, for en enhanced for
- Geneste constructies maken voor selectie- en iteratie-instructies
- Vergelijking van de do-while, while, for en verbeterde for loop-constructies
- Break en continue statements gebruiken

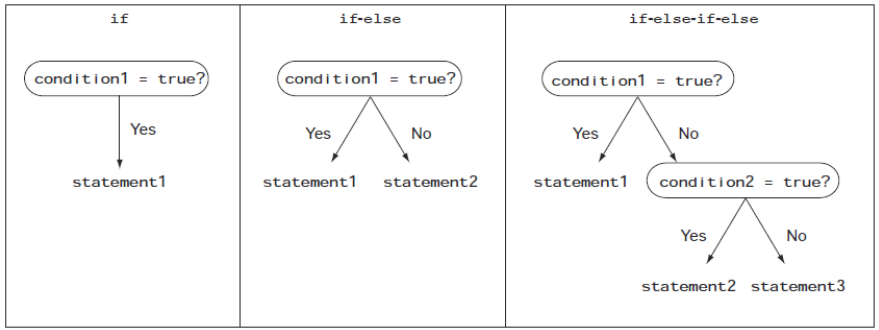
If, If-else en ternary statements

Met een if-constructie kan je een reeks instructies in je code uitvoeren op basis van het resultaat van een voorwaarde.

Deze voorwaarde moet altijd een booleaan zijn of een booleaanse waarde opleveren. Je kan een set instructies specificeren die moeten worden uitgevoerd wanneer deze voorwaarde waar of onwaar is.

Je hebt meerdere smaken if-statements:

- If
- If-else
- if-else-if-else



Selectie statments: if

Met behulp van een **if-statement** kun je code laten uitvoeren indien een bepaalde bewering **waar** is.

```
if (bewering) {
    // Deze code wordt uitgevoerd als de expressie van de if-bewerking waar is
}
```

Indien de bewering niet waar is, vervolgt de uitvoering van de code na het if-statement bijvoorbeeld:

```
if (radius >= 0) {
    oppervlakte = Math.PI * radius * radius;
    System.out.println("De oppervlakte van de cirkel: " + oppervlakte);
}
```

Selectie statments: if-else

Met behulp van een **else-statement** is het mogelijk om een stuk code uit te laten voeren indien de bewering **onwaar** is.

```
if (bewering) {
    // uitgevoerd als bewering true oplevert
}
else {
    // uitgevoerd als bewering false oplevert
}
```

Voorbeeld:

```
if (radius >= 0) {
    oppervlakte = Math.PI * radius * radius;
    System.out.println("De oppervlakte van de cirkel: " + oppervlakte);
}
else {
    System.out.println("De oppervlakte van de cirkel is ongeldig");
}
```

Meervoudige selectie statements: else-if

Je kunt een **if-** en **else-statement** ook combineren tot een **if-else-statement**.

Let op: Indien je geen accolades plaatst, behoort enkel de eerstvolgende regel bij het if of else-statement.

```
if (aantalMensen < 50)
    System.out.println("Er zijn minder dan 50 mensen");
else if (aantalMensen < 100)
    System.out.println("Er zijn minstens 50 en minder dan 100 mensen");
else if (aantalMensen < 200)
    System.out.println("Er zijn minstens 100 en minder dan 200 mensen");
else
    System.out.println("Er zijn meer dan 100 mensen");
```

Implicaties van de aan- en afwezigheid van {} in if-else constructies

Je kan een enkele instructie of een blok met instructies uitvoeren wanneer een if-voorwaarde waar of onwaar is.

Een if-blok wordt gemarkeerd door een of meer instructies tussen een paar accolades {} te plaatsen.

Een if-blok voert daarentegen ook een enkele regel code uit als er geen accolades zijn, maar voert een onbeperkt aantal regels uit als ze in een blok staan (gedefinieerd met accolades). De accolades zijn optioneel als er maar één regel in de if-instructie staat.

Voorbeeld

```
String naam = "Bob";
int score = 100;
if (naam.equals("Bob"))
    score = 200;
```

```
String naam = "Bob";
int score = 100;
if (naam.equals("Bob"))
    score = 200;
    naam = "Jan";
```

In het eerste voorbeeld wordt de score van Bob naar 200 gezet.
Maar wat gebeurt er in het tweede voorbeeld?

Ongeacht wat de naam is in de eerste regel, zal de naam altijd veranderen naar Jan, omdat deze niet meer onderdeel is van de if-statement.

Voorbeeld (2)

```
String naam = "Bob";
int score = 100;
if (naam.equals("Bob"))
    score = 200;
    naam = "Jan";
else:
    score = 142;
```

```
String naam = "Bob";
int score = 100;
if (naam.equals("Bob"))
    score = 200;
naam = "Jan";
else:
    score = 142;
```

Wat denk je dat er gebeurt bij deze twee stukken code?

De code compileert niet. Dit komt omdat de ‘else’ altijd onderdeel moet zijn van de if-statement. Dat is het in dit geval niet.

Voorbeeld (3)

```
String naam = "Bob";
int score = 100;
if (naam.equals("Bob")) {
    score = 200;
    naam = "Jan";
}
else:
    score = 142;
```

Door accolades toe te voegen geven we aan dat de if-statement meerdere regels bevat.

Vuistregel: ongeacht de aantal regels na de if-statement, gebruik altijd de accolades.

Nested If

```
public class NestedIf {
    public static void main(String[] args) {
        int eenNummer = 7;
        if (eenNummer >= 0)
            if (eenNummer == 0)
                System.out.println("eerste string");
        else
            System.out.println("tweede string");
        System.out.println("derde string");
    }
}
```

```
tweede string
derde string
```

Een **else-statement** bindt zich aan de dichtbijzijndede **if-statement**

Deze code is dus misleidend ingesprongen.

Ternary statements

Het komt regelmatig voor in programma's dat je de waarde van één variabele wilt toekennen aan de hand van een conditie.

```
Lijn lijn;
Punt middelpunt;

/* Allerlei magie hier */

if(lijn != null)
    middelpunt = lijn.middelpunt();
else
    middelpunt = new Punt(0, 0);
```

```
int a = 9;
int b = 3;
int min = 0;
if(a > b)
    min = b;
else
    min = a;
```

Tenary if

Om deze constructies om te vormen tot one-liners is er in veel programmeertalen de volgende expressie aanwezig:

```
conditie ? waarde als waar : waarde als onwaar
```

```
int a = 9;
int b = 3;
int min = (a > b) ? b : a;
```

Dit kan ook zonder haakjes:

```
int a = 9;
int b = 3;
int min = a > b ? b : a;
```

Tenary if genest

Je kunt ook meerdere van deze expressies in elkaar stoppen. Bedenk je alleen wel of dat de leesbaarheid van de code ten goede komt.

```
int a = 3;
int b = 4;
int c = 5;
int max = (a > b ? (a > c ? a : c) : (b > c ? b :
c));
```

```
if (a > b) {
    if (a > c) {
        max = a;
    } else {
        max = c;
    }
} else {
    if (b > c) {
        max = b;
    } else {
        max = c;
    }
}
```

If

Opdracht:
Op veel scholen gebruiken ze letters om cijfers bij te houden. Schrijf een methode die op basis van een input berekent wat dat cijfer is in letters. Gebruik het volgende schema:
- Onder 25 - F
- 25 tot 45 - E
- 45 tot 50 - D
- 50 tot 60 - C
- 60 tot 80 - B
- Boven 80 - A

Bonus:

Schrijf een methode genaamd *fibonacci*(int n) die gebruikmaakt van **recursie** en een **ternary if** om fibonacci cijfers te berekenen.

De methode mag slechts één regel lang zijn.

Switch statement

Een andere manier van conditionele logica binnen veel programmeertalen is het werken met switch statements. Sinds Java 7 zit dit ook in Java.

```
if (expressie == x) {
    // code blok
} else if (expressie == y) {
    // code blok
}
...  // enzovoort
```

If-statement

```
switch(expressie) {
    case x:
      // code block
      break;
    case y:
      // code block
      break;
    default:
      // code block
}
```

Switch statement

Meervoudige selectie statements: switch-case-statement

```
aantal = 0
switch (aantal) {
    case 0: System.out.println("nul");
        break;
    case 1: System.out.println("een");
        break;
    default: System.out.println("veel");
        break;
} // prints nul
```

```
aantal = 0
switch (aantal) {
    case 0: System.out.println("nul");
    case 1: System.out.println("een");
    default: System.out.println("veel");
} // prints nul, een, veel
```

Let op: De break is optioneel

Verschil tussen switch en if-statement

```
String dag = "ZO";
if (dag.equals("MA") || dag.equals("DI")||
    dag.equals("WO") || dag.equals("DO"))
    System.out.println("Tijd om te werken");
else if (dag.equals("VR"))
    System.out.println("Bijna weekend");
else if (dag.equals("ZA") || dag.equals("ZO"))
    System.out.println("Weekend!");
else
    System.out.println("Geen dag?");
```

```
switch (dag) {
    case "MA":
    case "DI":
    case "WO":
    case "DO":
        System.out.println("Tijd om te werken");
        break;
    case "VR":
        System.out.println("Bijna weekend");
        break;
    case "ZA":
    case "ZO":
        System.out.println("Weekend!");
        break;
    default:
        System.out.println("Geen dag?");
        break;
}
```

Switch statement

Opdracht:

Schrijf het gegeven if-statement om naar een switch case

```
if (browser == 'Edge') {
    System.out.println("Je gebruikt Edge");
}
else if (browser == 'chrome' || browser == 'firefox' || browser == 'opera') {
    System.out.println("Je gebruikt een moderne browser");
}
else if (browser == 'IE') {
    System.out.println("Waarom gebruik je Internet Explorer?");
}
else {
    System.out.println("Je gebruikt een niet-ondersteunde browser");
}
```

De for-loop

Met behulp van loops kunnen we een stuk code meerdere keren laten uitvoeren

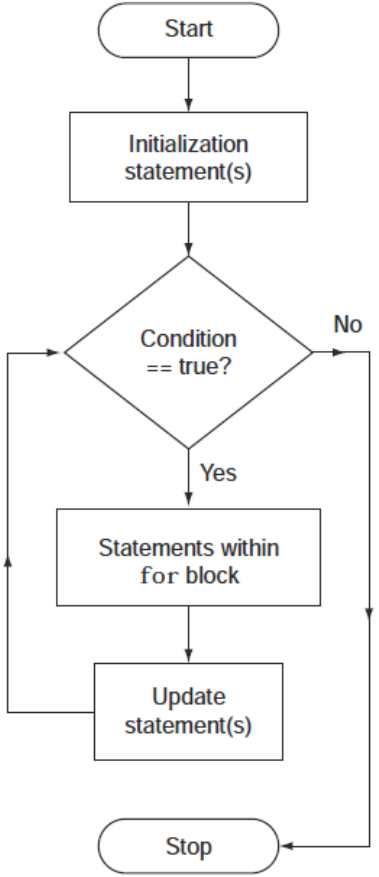
De **for-loop** gebruik je als je van te voren exact weet hoeveel iteraties je wilt gebruiken

```
for(initialisatie; conditie; stap){
    // inhoud van de for-loop
}
```

Herhaling met de for-loop

```java
/*
 * Bereken som van getallen 1 t/m 10
 */
int som = 0;
for (int i = 1; i <= 10; i++) {
    som += i;
}
System.out.println(som);
```

55



Herhaling met de for-loop vanaf 0

```java
/*
 * vijf keer
 */
int k = 1;
for (int i = 0; i < 5; i++) {
    System.out.println(i + " " + k);
    k = k * 2;
}
```

```
0 1
1 2
2 4
3 8
4 16
```

De geneste for-loop

Ook for-loops kan je nesten. Een voorbeeld hiervan is bijvoorbeeld werken met een klok. Dan heb je te maken met minuten en uren. Beide krijgen dan hun eigen for-loop:

```
for (int uur = 1; uur <= 6; uur++) {
    for (int min = 1; min <= 60; min++) {
        System.out.println(uur + ":" + min);
    }
}
```

Hiernaast worden dubbele for-loops veel gebruikt voor multidimensionale arrays.

De geneste for-loop

Opdracht:

Schrijf een programma dat de volgende patroon genereert:

```
Tafelmaker: Geef de grootte aan: 10
 * |  1   2   3   4   5   6   7   8   9  10
-----------------------------------------------
 1 |  1   2   3   4   5   6   7   8   9  10
 2 |  2   4   6   8  10  12  14  16  18  20
 3 |  3   6   9  12  15  18  21  24  27  30
 4 |  4   8  12  16  20  24  28  32  36  40
 5 |  5  10  15  20  25  30  35  40  45  50
 6 |  6  12  18  24  30  36  42  48  54  60
 7 |  7  14  21  28  35  42  49  56  63  70
 8 |  8  16  24  32  40  48  56  64  72  80
 9 |  9  18  27  36  45  54  63  72  81  90
10 | 10  20  30  40  50  60  70  80  90 100
```

De enhanced for-loop

De enhanced for-loop wordt ook wel de for-each-loop genoemd en biedt enkele voordelen ten opzichte van de reguliere for-loop.

Met behulp van een **foreach loop** kun je over alle elementen uit een array itereren:

```
for(type naam : lijst) {
    // het element staat nu toegankelijk via de variabele
"naam"
}
```

Foreach vs For-loop

```java
int[] lijst = {1,2,3,4};
int som1 = 0, som2 = 0;

for(int i = 0; i < lijst.length; i++) {
    som1 = som1 + lijst[i];
}

for(int getal : lijst) {
    som2 = som2 + getal;
}

System.out.println("som1: " + som1); // som1:
10
System.out.println("som2: " + som2); // som2:
10
```

While en do-while loops

Beide loops voeren een reeks instructies uit zolang hun voorwaarde als **true** wordt geëvalueerd.

We onderscheiden op dit moment drie soorten loops:

- For-loop: als je van te voren exact weet hoeveel iteraties je hebt.
- While-loop: als je niet van te voren weet hoeveel iteraties je hebt, maar mogelijk geen.
- Do-while-loop: als je niet van te voren weet hoeveel iteraties je hebt maar minstens één

Kun je voor ieder type een loop een situatie bedenken?

De while-loop

De code in de while-loop wordt uitgevoerd, zolang de evaluatie van de conditie true oplevert.

```
while(conditie) {
    // code die uitgevoerd wordt, zolang conditie == waar
}
```

```
int count = 1;
while(count < 11) {
    System.out.println("Teller: " + count);
    count++;
}
```

Aan welk for-loop is dit equivalent?

De do-while-loop

De code in de **do-while-loop** wordt altijd **minstens één maal** uitgevoerd, en vervolgens net zolang herhaald zolang de evaluatie van de conditie **true** oplevert.

```
do {
    // code
} while(conditie);
// code die uitgevoerd wordt, zodra conditie != waar
```

Kunnen we een eenvoudig programmaatje verzinnen dat hier gebruik van maakt?

De do-while-loop

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

    // Expressie declareren en initialiseren
    int i = 1;

    // Do-while loop
    do {

        // Body van do-while loop
        // Print statement
        System.out.println("Hallo Wereld");

        // Update expression
        i++;
    }

    // Test expression
    while (i < 6);
}
```

While en do-while loops

Opdracht:

Transformeer de code naar een while en een do-while loop

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

for (int i = 0; i < 5; i++) {
    list [i] = i + 2;
}
```

Loop statements: break en continue

Met behulp van een spronginstructie kunnen we het gedrag van een loop beïnvloeden:

- **Break**: Stopt de uitvoering van de loop

- **Continue**: Spring naar de volgende iteratie van de loop

Break

```
int i;
String str = "Hallo Hi Hallo";

for(i = 0; i < str.length(); i++) {
    if(str.charAt(i) == 'l')
        break;
}

System.out.println("Eerste l gevonden op positie: " +
i);
```

Continue

```
int aantalHs = 0;
String str = "Hallo Hi Hallo";

for(char karakter : str.toCharArray()) {
    if (karakter != 'H')
        Continue;

    // we slaan dit deel over als het karakter geen H
is
    aantalHs = aantalHs + 1;
    System.out.println("H gevonden!");
}

System.out.println("Aantal gevonden Hs: " + aantalHs);
```

Geneste loops: Wat zal er gebeuren?

```
for(outerinit; outerLoopCondition; outerUpdate) {
    // A;
    for(innerInit; InnerLoopCondition; innerUpdate) {
        //B;
        if(breakCondition)
            break;
        //C;
    }
    // D;
}
```

Eindopdracht

- Gegeven een invoer van de gebruiker, doe het volgende:
  - Print het aantal karakters
  - Print het aantal woorden dat in de invoer zat
  - Print het aantal klinkers (a, e, i, o, u, y)

  - Bepaal of de invoer een palindroom is
  - Maak een staafdiagram dat weergeeft hoe vaak een bepaalde karakters gezien zijn.

```
java eindopgave
Voer een zin in: nee weg t kaatsnet en staak t geween

Aantal karakters: 36
Aantal woorden:   8
Aantal klinkers:  12
Palindroom?       True



    *                                   
    *                                   *
    *                                   *
    *                                   *
*   *         *     *                   *
*   *         *     *                   *
*   * *   *   *   * *   *               *
*   * *   *   *   * *   *               *
a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9
```

Leerdoelen

- ✓ Het aanmaken en gebruik van if, if-else, ternary en switch statements
- ✓ Het aanmaken en gebruik van loops: while, do-while, for en enchaned for
- ✓ Geneste constructies maken voor selectie- en iteratie-instructies
- ✓ Vergelijking van de do-while, while, for en verbeterde for loop-constructies
- ✓ Break en continue statements gebruiken

Vragen?

- E-mail mij op voornaam.achternaam@code-cafe.nl!

- Join de Code-Café community op discord!