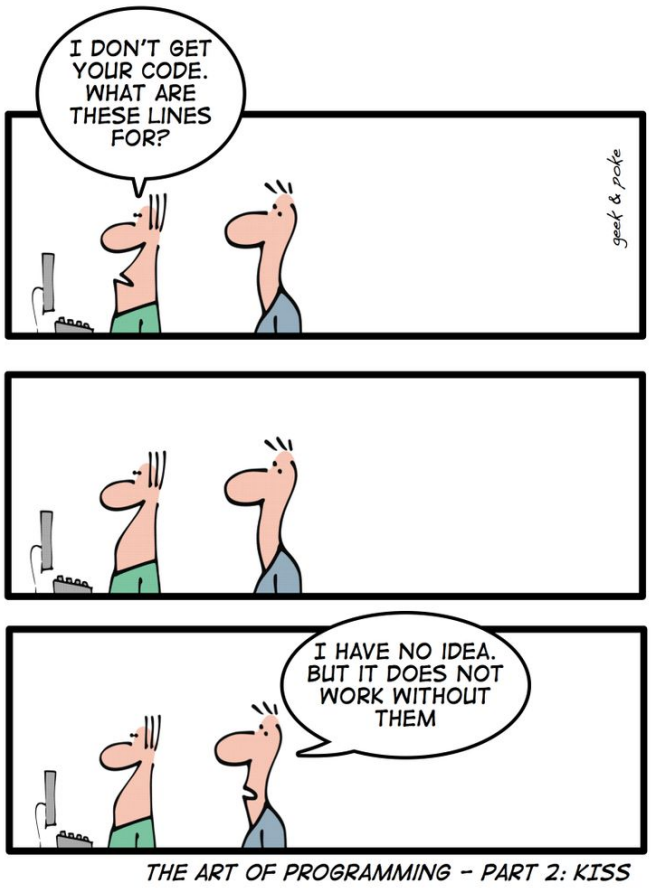
# Universiteit van Amsterdam
## Inleiding Programmeren

## Opgave 1

José Lagerberg - Robin de Vries - Marco Brohet - Youri Voet

# Introductie




**Deadline: zaterdag 7 september 2019, 18:00**

# 1 Leerdoelen

Aan het einde van deze opdracht kun je:

- een eenvoudig programma schrijven in Java,
- gebruik maken van de Scanner en for-loops,
- een consistente en nette programmeerstijl toepassen,
- code voorzien van zinnig en helder commentaar.

Neem ook de rubric door die je kunt vinden bij de opdracht op Canvas. In deze rubric wordt aangegeven hoe de opgave beoordeeld wordt.

# 2 Java en Linux

Neem voordat je aan de opdracht begint de "Linux introductie" en de "Tips & Tricks" op Canvas door. Deze kun je vinden onder de "Algemene informatie"-module.

Maak vervolgens een map aan voor Inleiding Programmeren. In deze map maak je dan nog zeven aparte submappen, voor elke opgave één. Gebruik hiervoor de terminal. In Linux is dit namelijk een van de belangrijkste en snelste manieren om je computer te beheren.

# 3 Je eerste programma

Om te controleren of je werkomgeving in orde is, volgt nu het schrijven, compileren en uitvoeren van een testprogramma dat de tekst "Hallo wereld!" print. Dit programma hoeft niet te worden ingeleverd.

Maak in je Inleiding Programmeren-map de submap **opgave0** aan. Open je favoriete teksteditor, neem de code over van Voorbeeld 1 en sla het op in de submap als `Opgave0.java`.

```java
public class Opgave0 {
    public static void main(String[] args) {
        System.out.println("Hallo wereld!");
    }
}
```

Voorbeeld 1: Opgave0.java

Compileer het programma met de Java compiler, **javac** met het commando '**javac Opgave0.java**'. Gebruik het commando **ls** om te zien of dit een '.class' bestand heeft opgeleverd. Experimenteer nu met de volgende handelingen:

- Het uitvoeren van een Java programma:

  ```
  $ java Opgave0
  ```

- Resultaten naar een bestand (**results0**) schrijven:

  ```
  $ java Opgave0 > results0
  ```

- Een bestand kopiëren:

  ```
  $ cp Opgave0.java backup0.java
  ```

# 4 Voordat je begint

De code die je inlevert moet aan enkele eisen voldoen. Lees voor het inleveren van de eerste opgave goed de zogeheten 'styleguide' door. Hierin staan de regels en richtlijnen voor de layout van je code. Probeer je aan deze regels te houden, maar blijf vooral consistent in je stijl van programmeren.

De code moet netjes, leesbaar en overzichtelijk zijn en er moet voldoende commentaar aanwezig zijn. Commentaar is bedoeld om de lezer van de code te helpen te begrijpen wat er gebeurt. Overbodig commentaar is commentaar dat niets toevoegt aan de code, zoals "hier wordt iets geprint". Dat kan je dus beter weglaten. De keuze van goede namen voor variabelen en methoden helpt ook om een programma leesbaar te maken.

Boven elke methode en lastigere stukken code moet altijd commentaar worden geplaatst. Daarnaast moet er bovenaan ieder bestand een kort stukje commentaar staan, de zogenaamde *header*. Zorg dat jouw persoonlijke informatie altijd in de eerste vijf regels van jouw header staat en dat hier nooit andere informatie in komt te staan. In de header zet je:

- je naam
- je UvAnetID (studentnummer)
- je studie
- een korte omschrijving van de functionaliteit van de code in dit bestand en/of een korte omschrijving van het probleem dat aan de orde is en hoe dit is opgelost.

In Voorbeeld 2 staat een voorbeeld van een goede header.

```
/*
 *  Naam     : K. Klaassen
 *  UvAnetID : 12345678
 *  Studie   : BSc Informatica
 *
 *  Voorbeeld.java:
 *  -  Dit programma berekent de wortels van de vergelijking
 *     ax^2 + bx + c = 0
 *     Deze oplossing wordt berekend met de a-b-c formule
 *  -  De coefficienten moeten in de command line opgegeven worden
 *  -  Als de discriminant kleiner is dan 0, is er geen oplossing
 */

public class Voorbeeld
{

}
```

Voorbeeld 2: Voorbeeld.java

# 5 Opgave 1

De eerste opgave bestaat uit vier verschillende onderdelen die ieder in een apart '.java'-bestand geïmplementeerd moeten worden. Het eerste onderdeel moet worden gemaakt in het bestand **Deel1.java**, het tweede in **Deel2.java**, etc.

## 5.1 Deel 1

In het eerste onderdeel moet je met een aantal **print** statements je naam printen in ASCII-art. ASCII-art is een kunstvorm waarbij afbeeldingen worden samengesteld uit enkel ASCII-tekens. Vooral in niet-grafische omgevingen zoals bepaalde computerterminals, e-mail, SMS en chatomgevingen wordt dit wel gebruikt om tekst toch van vaak zeer eenvoudige illustraties te kunnen voorzien. Een voorbeeld hiervan is:

```
     __
    |__|____ ___  _______
    |  \__  \\  \/ /\__  \
    |  |/ __ \\   /  / __ \_
/\__|  (____  /\_/  (____  /
\______|    \/           \/
```

`ASCII-art Java in graffity font`

Zie `https://nl.wikipedia.org/wiki/ASCII-art`. Op de website `http://www.patorjk.com/software/taag` kun je je naam omzetten naar ASCII-art. Gebruik hiervoor het *graffity* font, en kopieer het resultaat. Dit moet je printen in **Deel1.java**.

## 5.2 Deel 2

In **Deel2.java** moet via de terminal een geheel positief getal opgegeven worden. Het getal moet opgevraagd worden met behulp van de **nextInt**-methode uit de Scanner-klasse. Zorg ervoor dat verkeerde invoer goed wordt afgehandeld door het programma. Vervolgens moet de som van alle even getallen van 1 tot en met het opgegeven getal worden berekend. Ook moet de som van alle oneven getallen van 1 tot en met het opgegeven getal worden berekend. Als laatste moet het verschil van deze twee sommen worden geprint. Een voorbeeld van de uitvoer is:

```
$ java Deel2
Geef een geheel positief getal: 10
som van oneven getallen tot en met 10 is 25
som van   even getallen tot en met 10 is 30
verschil tussen twee sommen is -5

$ java Deel2
Geef een geheel positief getal: 11
som van oneven getallen tot en met 11 is 36
som van   even getallen tot en met 11 is 30
verschil tussen twee sommen is 6

$ java Deel2
Geef een geheel positief getal: -128
Dit is geen geheel positief getal
```

## 5.3 Deel 3

In dit onderdeel moet je een Java programma schrijven dat een interactief raadspelletje speelt met de gebruiker. In het programma moet een willekeurig getal tussen 1 en 10 gegenereerd worden, dat de gebruiker met drie pogingen kan raden. Na elke poging moet het programma aangeven of de gebruiker het goed geraden heeft, of te hoog of te laag zit.

```
$ java Deel3
Geef een getal tussen 1 en 10, je mag drie keer raden
Eerste keer: 20
Getal was niet tussen 1 en 10, dan stop ik

$ java Deel3
Geef een getal tussen 1 en 10, je mag drie keer raden
Eerste keer: 3
te klein
Tweede keer: 8
te groot
Derde keer: 6
gewonnen

$ java Deel3
Geef een getal tussen 1 en 10, je mag drie keer raden
Eerste keer: 9
te groot
Tweede keer: 4
te groot
Derde keer: 2
te groot
verloren, het getal was 1
```

## 5.4 Deel 4

Het programma vraagt de gebruiker een natuurlijk getal $n$ en print vervolgens de eerste $n$ zogenaamde Lucas-getallen. Deze getallen worden gegeven door de reeks:

```
2 1 3 4 7 11 18 ...
```

Het eerste Lucas-getal is 2, het tweede is 1. Daarna krijg je het volgende getal telkens door de twee voorgaande getallen bij elkaar op te tellen. In je programma moet je testen of het door de gebruiker ingetypte getal wel positief is. Verder kunnen de getallen van de Lucas-reeks zo groot worden dat ze niet meer passen in een **int**. Bouw in je programma een test in, zodat bij een te grote waarde van $n$ niets geprint wordt. Uitvoer zou kunnen zijn:

```
$ java Deel4
Geef een natuurlijk getal: 5
De eerste 5 Lucas-getallen:
2 1 3 4 7

$ java Deel4
Geef een natuurlijk getal: -6
Getal negatief, fout

$ java Deel4
Geef een natuurlijk getal: 60
Getal te groot, past niet
```

# 6 Inleveren

Stop je bestanden in een archief en lever deze in op Canvas. Inpakken kan bijvoorbeeld zo:

```
$ tar zcvf mijn_opgave1.tar.gz Deel?.java
```