FACULTEIT ECONOMIE
EN BEDRIJFSKUNDE

# OBJECTGEORIËNTEED PROGRAMMEREN

F000887

UNIVERSITEIT
GENT

# INHOUD

- Inleiding tot programmeren met Java
- **Basisbewerkingen Java**
- Flow of Control: selectie
- Flow of control: herhaling
- Definieren van klassen en methodes
- Arrays
- Overerving, Polymporhisme en Interfaces
- GUI: Frames, Figuren, Labels, Tekstvakken, Knoppen, Dialoogvensters en events

FACULTEIT ECONOMIE
EN BEDRIJFSKUNDE

VAKGROEP BELEIDSINFORMATICA EN OPERATIONEEL BEHEER
BUSINESS INFORMATICS RESEARCH GROUP

# BASISBEWERKINGEN

## Hoofdstuk 2

UNIVERSITEIT
GENT

# OUTLINE

- Variabelen en expressies
  - Variabelen, Constanten en Data types
  - Java identifiers
  - Toekenningsstatements
  - Type casting
  - Wiskundige operatoren
  - Increment en decrement operatoren
- De String klasse
- Keyboard en Scherm I/O
- Documenteren van code en stijl
- GUI: JFrame en JOptionPane klasse

# VARIABELEN

- Variabelen worden gebruikt voor het opslaan van data zoals getallen en karakters
- Worden geïmplementeerd als geheugen adressen
- De data die wordt opgeslagen in een variabele noemt men zijn waarde
  - De waarde wordt opgeslagen op de geheugenlocatie dat overeenkomt met de variabele
  - De waarde van een variabele kan gewijzigd worden

# CASE: VARIABELEN

- **Demo EierMandje:**

  **https://youtu.be/luozYiWv43Y**

```java
public class EierMandje {

  /**
   * @param args the command line arguments
   */
  public static void main(String[] args) {
    int numberOfBaskets, eggsPerBasket, totalEggs;
    numberOfBaskets = 10;
    eggsPerBasket = 6;
    totalEggs = numberOfBaskets * eggsPerBasket;
    System.out.println ("Indien je beschikt over");
    System.out.println (eggsPerBasket + " eieren per mandje en");
    System.out.println (numberOfBaskets + " mandjes, dan");
    System.out.println ("is het totaal aantal eieren " + totalEggs);
  }
}
```

UNIVERSITEIT GENT

# DECLAREREN VAN VARIABELEN

- Kies een naam met betekenis, bijvoorbeeld **snelheid** of **aantal**, maar niet **s** of **a**
- Een variabele wordt gedeclareerd door specificeren van naam en type
  `int numberOfBaskets,eggsPerBasket;`
- Het type bepaalt het soort data dat men kan opslaan in de variabele (`int`, `double`, `char`, etc.).
- Een variabele moet altijd eerst gedeclareerd worden alvorens deze wordt gebruikt.

# SYNTAX EN VOORBEELDEN

– Syntax

```
type variable_1, variable_2, …;
```

– Voorbeelden:

……

……

……

UNIVERSITEIT GENT

# SOORTEN DATATYPES

- Een klasse datatype wordt gebruikt voor het opslaan van een object van deze klasse en heeft zowel data als methodes
  - `"Java is fun"` is een waarde voor het klasse type `String`
- Een primitief datatype wordt gebruikt voor het opslaan van eenvoudige, onsplitsbare waarden zoals een getal of karakter
  - `int`, `double` en `char` zijn primitieve datatypes.

# PRIMITIEVE DATATYPES

| Type Name | Kind of Value | Memory Used | Range of Values |
|---|---|---|---|
| `byte` | Integer | 1 byte | –128 to 127 |
| `short` | Integer | 2 bytes | –32,768 to 32,767 |
| `int` | Integer | 4 bytes | –2,147,483,648 to 2,147,483,647 |
| `long` | Integer | 8 bytes | –9,223,372,036,8547,75,808 to 9,223,372,036,854,775,807 |
| `float` | Floating-point | 4 bytes | $\pm 3.40282347 \times 10^{+38}$ to $\pm 1.40239846 \times 10^{-45}$ |
| `double` | Floating-point | 8 bytes | $\pm 1.79769313486231570 \times 10^{+308}$ to $\pm 4.94065645841246544 \times 10^{-324}$ |
| `char` | Single character (Unicode) | 2 bytes | All Unicode values from 0 to 65,535 |
| `boolean` | | 1 bit | True or false |

# WAARDEN PRIMITIEVE DATATYPES

- Integer types:

  .....

- Floating-point types:

  .....

- Character type:

  .....

- Boolean type:

  .....

UNIVERSITEIT GENT

# JAVA IDENTIFIERS

- Een *identifier* is een naam, zoals de naam van een variabele
- Identifiers mogen enkel bestaan uit
  - Letters
  - Cijfers (0 tot 9)
  - Underscore karakter (_)
  - Het dollar teken ($) dat een speciale betekenis heeft.
- Het eerste karakter mag geen cijfer zijn.

# JAVA IDENTIFIERS

- Identifiers mogen niet bestaan uit spaties, punten (.), sterretjes (*):

  `7-11  netscape.com  util.*`

- Identifiers kunnen heel lang zijn
- Java is hoofdlettergevoelig, bijgevolg zijn `stuff`, `Stuff`, en `STUFF` verschillende identifiers.

# JAVA KEYWORDS

- Woorden zoals `if` noemt men *keywords* omdat ze een speciale voorgedefinieerde betekenis hebben:
  - Ze kunnen niet als identifiers gebruikt worden
  - Zie Appendix 1 voor een complete lijst van alle Java keywords.
- Voorbeelden:

  .....

  .....

# NAAM CONVENTIES

- Klasse datatypes beginnen met een hoofdletter (e.g. `String`).
- Primitieve types beginnen met een kleine letter (e.g. `int`).
- Variabelen van zowel klassen als primitieve types beginnen met een kleine letter
  (e.g. `myName`, `myBalance`).
- Namen die bestaan uit meerdere woorden maken gebruik van hoofdletters om het begin van een nieuw woord aan te duiden.

# WAAR VARIABELEN DECLAREREN

- Declareer een variabele
  - net voor deze gebruikt wordt
  - In het begin van een blok binnen je code dat wordt begrensd aan de hand van `{ }`.

```
public static void main(String[] args)
{ /* declareer variabelen hier */
      . . .
}
```

# TOEKENNINGSSTATEMENT

- Een toekenningsstatement wordt gebruikt voor een waarde toe te kennen aan een variabele

```
answer = 42;
```

- De ”is gelijk aan” operator is de toekenningsoperator.
- Dus in het voorbeeld is de waarde 42 toegekend aan de variabele met als naam `answer`

# TOEKENNINGSSTATEMENT

- Syntax

  **`variabele = expressie`**

  waarbij **`expressie`** een andere variabele kan zijn, een letterwaarde of constante, of een complexere expressie dat verschillende variabelen, letterwaarde of constanten combineert aan de hand van een operator.
- Voorbeelden:

  .....

  .....

# INITIALISEREN VARIABELEN

- Een variabele kan gedeclareerd zijn maar nog nog niet geinitialiseerd.
- Niet geinitialiseerde klasse variabelen hebben de waarde `null`.
- Niet geinitialiseerde primitieve variabelen kunnen een standaardwaarde hebben.
- Het is beter om steeds zelf te initialiseren in plaats van te werken met standaardwaarden.

# INITIALISEREN VARIABELEN

- syntax

```
type variable_1 = expression_1, variable_2 =
expression_2, …;
```

- Voorbeelden:

```
.....

.....

.....
```

UNIVERSITEIT GENT

# EVALUATIE TOEKENNINGSSTATEMENT

- De expressie aan de rechterzijde van de toekenningsoperator (=) wordt eerst geëvalueerd.
- Het resultaat wordt gebruikt als waarde voor de variabele aan de linkerzijde van de toekenningsoperator.

```
score = numberOfCards +  handicap;
eggsPerBasket = eggsPerBasket - 2;
```

# CONSTANTEN

- Letterwaarden zoals `2, 3.7,` or `'y'` noemt men constanten.
- Integer constanten kunnen vooraf gegaan worden door een `+` of `-` teken, maar kunnen geen komma's bevatten.
- Floating-point constanten kan men schrijven door
  - Toevoegen van cijfers na komma
  - Door gebruik te maken van de wetenschappelijke notatie

# WETENSCHAPPELIJKE NOTATIE

- Voorbeelden
  - `865000000.0` kan men schrijven als `8.65e8`
  - `0.000483` kan men schrijven als `4.83e-4`
- Het getal voor de `e` mag een decimaal punt bevatten, maar dit moet niet.

# ONNAUWKEURIGHEID FLOATING-POINT GETALLEN

- Floating-point getallen zijn benaderingen aangezien ze worden opgeslagen aan de hand van een eindig aantal bits
- Bijgevolg is
  - `1.0/3.0` kleiner dan 1/3.
  - `1.0/3.0 + 1.0/3.0 + 1.0/3.0` kleiner dan 1.

# BENOEMDE CONSTANTEN

- Java maakt het mogelijk om …
  - Een variabele te declareren
  - Een waarde toe te kennen die niet kan wijzigen

```
final Type Variable = Constant;
```

  - Voorbeeld PI:

    ……

# TOEKENNINGSCOMPATIBILITEIT

- Java is sterk getypeerd
  - Men kan geen floating point waarde toekennen aan een variabele die gedeclareerd is om een int waarde op te slaan.
- Soms zijn conversies echter wel mogelijk

  `doubleVariable = 7;`

  is mogelijk ondanks dat `doubleVariable` van het type `double` is.

# TOEKENNINGSCOMPATIBILITEIT

- Een waarde van een type kan toegekend worden aan een variabele van het type rechts van hem

  **`byte --> short --> int --> long`**

  **`--> float --> double`**

  - Maar niet omgekeerd
- Men kan ook een waarde van het type **`char`** toekennen aan een variabele van type **`int`**.

# TYPE CASTING

- Een type cast verandert tijdelijk de waarde van een variabele van het gedeclareerde type naar een ander type
- Syntax:

  **(type) naam_variabele**

- Voorbeelden:

  .....

  .....

UNIVERSITEIT GENT

# WISKUNDIGE OPERATOREN

- Wiskundige expressies kan men vormen aan de hand van de operatoren `+`, `-`, `*`, en `/` in combinatie met de variabelen en getallen, operands genaamd.
  - Wanneer de beide operands van hetzelfde type zijn, dan is het eindresultaat van het zelfde type
  - Wanneer één van de operands een floating point type is en het ander een integer type, dan is het resultaat een floating point type.

# WISKUNDIGE OPERATOREN

- Expressies met twee of meerdere operatoren kan men beschouwen als een serie van stappen, elk bestaande uit slechts 2 operands.
  - Het resultaat van een bepaalde stap wordt gebruikt in de volgende stap
- voorbeeld

```
balance + (balance * rate)
```

# WISKUNDIGE OPERATOREN

- Wanneer een van de operands een floating point en de andere operands integers, dan zal het resultaat een floating point zijn.
- Het resultaat van een berekening zal overeenkomen met het type dat het meest rechts staat in de lijst van types dat voorkomt in de expressie

```
byte --> short --> int --> long
--> float --> double
```

# DE DELING OPERATOR

- De deling operator (`/`) werkt zoals verwacht indien één van de operands een floating point waarde is.
- Wanneer de twee operands integers zijn, wordt het resultaat afgekapt en niet afgerond.
  - Dus `99/100` is gelijk aan `0`.

# DE **MOD** OPERATOR

- De `mod` (`%`) operator wordt gebruikt om de rest van een integer deling te bepalen
- 14 gedeeld door 4 is 3 *met als rest 2.*
  - dus `14 % 4` is gelijk aan `2.`
- Heeft veel toepassingen, zoals
  - Bepalen of een getal even of oneven is
  - Bepalen of getal deelbaar is door ander getal.

# HAAKJES EN VOORRANGSREGELS

- Haakjes worden gebruikt voor bepalen volgorde waarin wiskundige bewerkingen worden uitgevoerd.
- Voorbeelden:

```
(cost + tax) * discount
(cost + (tax * discount)
```

- Zonder haakjes worden de voorrangsregels gevolgd.

# VOORRANGSREGELS

| EERSTE |
| --- |
| UNAIRE OPERATOREN: +, -, ++, --,  ! |
| BINAIRE WISKUNDIGE OPERATOREN: *, / EN % |
| BINAIRE WISKUNDIGE OPERATOREN: + EN - |
| LAATSTE |

# VOORRANGSREGELS

- Wanneer unaire operatoren dezelfde voorrang hebben, wordt de operator het meest rechts het eerst uitgevoerd.
- Haakjes kunnen het lezen van code ook vereenvoudigen

```
balance + (interestRate * balance)
```

- Spaties maken het lezen van code ook duidelijker

```
balance + interestRate*balance
```

maar hebben geen invloed op de volgorde .

# SPECIALE TOEKENNINGSOPERATOREN

- Toekenningsoperatoren kan men combineren met wiskundige operatoren

```
amount = amount + 5;
```

kan men schrijven als

```
amount += 5;
```

# INCREMENT EN DECREMENT OPERATOREN

- Worden gebruikt om een waarde van variabele te vermeerderen of te verminderen met 1
- Eenvoudig in gebruik
- De increment operator

  `count++` of `++count`
- De decrement operator

  `count--` of `--count`

# INCREMENT EN DECREMENT OPERATOREN

– equivalente operaties

```
count++;
++count;
count = count + 1;

count--;
--count;
count = count - 1;
```

# INC EN DEC OPERATOREN IN EXPRESSIES

- Na uitvoering van

```
int m = 4;
int result = 3 * (++m)

Waarde Result en m ???
```

- Na uitvoering van

```
int m = 4;
int result = 3 * (m++)

Waarde Result en m ???
```

# DEMO WISSELAUTOMAAT

- Vereisten
  - De gebruiker geeft een waarde in tussen 1 en 99
  - Het programma bepaalt een combinatie van muntstukken dat gelijk is aan dit bedrag.
  - BVB, 55 eurocent komt overeen met twee stukken van 20 cent, 1 van 10 cent en 1 van 5 cent.

# DEMO WISSELAUTOMAAT

- https://youtu.be/eb24kpVngXI

```
run:
Geef een bedrag tussen 0 en 100
Ik zal een combinatie van munten bepalen
dat overeenkomt met dit bedrag
89
89 eurocent kan gewisseld worden in de volgende munten :
1 stukken van 50 cent
1 stukken van 20 cent
1 stukken van 10 cent
1 stukken van 5 cent
2 stukken van 2 cent
0 stukken van 1 cent
BUILD SUCCESSFUL (total time: 10 seconds)
```

# OUTLINE

- Variabelen en expressies
- **De String klasse**
- Keyboard en Scherm I/O
- Documenteren van code en stijl.
- GUI: JFrame en JOptionPane

# DE `STRING` KLASSE

- Reeds op verschillende plaatsen hebben we gebruik gemaakt van constanten van het type `String` :

```
"Geef een getal tussen 1 en 99"
```

- Een waarde van het type `String` is een
  - Sequentie van karakters
  - Wordt beschouwd als 1 enkel item

# STRING CONSTANTEN EN VARIABELEN

- Declaratie en initialisatie

```
String greeting;
greeting = ”Hallo!”;
```

of

```
String greeting = ”Hallo!”;
```

of

```
String greeting = new String(”Hallo!");
```

- Printing

```
System.out.println(greeting);
```

# AANEENSCHAKELING VAN STRINGS

- Twee strings plakt men aan elkaar door gebruik te maken van de + operator.

```
String greeting = ”Hallo";
String sentence;
sentence = greeting + " professor";
System.out.println(sentence);
```

- Onbeperkt aantal Strings kunnen aan elkaar geplakt worden door gebruik te maken van de + operator.

# AANEENSCHAKELEN VAN STRINGS EN INT

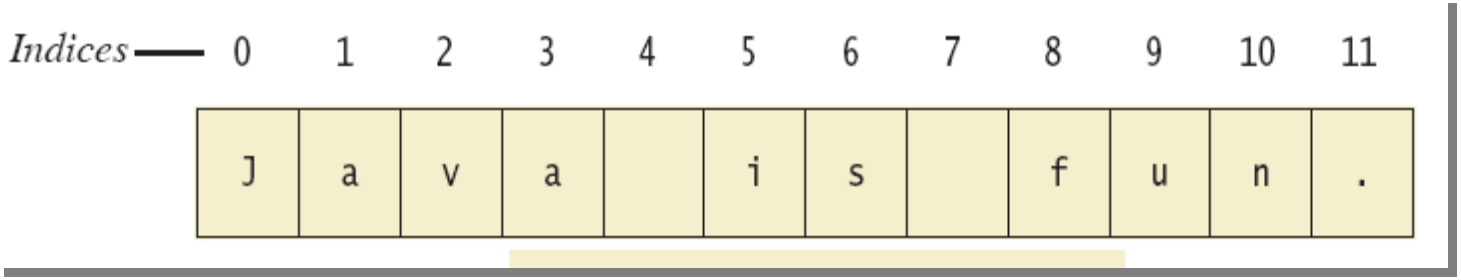
```
String solution;
solution = ”Het antwoord is " + 42;
System.out.println (solution);
```

# STRING METHODEN

- Een object van de klasse `String` bevat als data een sequentie van karakters
- Objecten hebben naast data ook methoden
- De `length()` methode retourneert het aantal karakters van het `String` object.

```
String greeting = ”Hallo";
int n = greeting.length();
```

# STRING INDICES




- Positie start bij 0 en niet bij 1
  - De 'J' in "Java is fun." staat op positie 0
- Een positie wordt een index genoemd:
  - De `'f'` in `"Java is fun."` bevindt zich op index 8.

# `STRING` METHODEN

- charAt(Index)
- compareTo(A_String)
- concat(A_String)
- equals(Other_String)
- equalsIgnoreCase( Other_String)
- indexOf(A_String)
- lastIndexOf(A_String)
- length()
- toLowerCase()
- toUpperCase()
- Replace(OldChar, NewChar)
- subString(Start)
- subString(Start, End)
- trim()

# ESCAPE CHARACTERS

- Hoe zou je het volgende weergeven

```
"Java" is een programmeertaal.
```

- Men moet aan de compiler duidelijk maken dat de dubbele aanhalingstekens (") niet het einde van de String aanduiden, maar moeten weergegeven worden.

```
System.out.println(
"\"Java\" is een programmeertaal."
```

# ESCAPE CHARACTERS

- \" : dubbele aanhalingstekens
- \': enkele aanhalingstekens
- \\: backslash
- \n: nieuwe lijn (ga naar begin van de volgende lijn)
- \r: cariage return (ga naar begin van de huidige lijn)
- \t: tab (spaties toevoegen tot volgende tabstop)

!!! Elke escape sequentie komt overeen met één karakter ondanks dat er twee symbolen worden gebruikt.

# DE LEGE STRING

- Een string kan 0 of meerdere karakters bevatten
- Een String met 0 karakters noemt men de lege String
- De lege String wordt veel gebruikt en kan men als volgt creëren:

```
String s3 = "";
```

# OUTLINE

- Variabelen en expressies
- De string klasse
- **Keyboard en Scherm I/O**
- Documenteren van code en programmeerstijl
- GUI: JFrame en JOptionPane

# SCHERM OUTPUT

- Wat meer uitleg over scherm output
- `System.out` is een object dat deel uitmaakt van Java.
- `println()` is een methode dat beschikbaar is voor het `System.out` object.

# SCHERM OUTPUT

- Alternatief, gebruik `print()`

```
System.out.print("One, two,");
System.out.print(" buckle my shoe.");
System.out.println(" Three, four,");
System.out.println(" shut the door.");
```

  eindig met `println()`.

# KEYBOARD INPUT

- Java beschikt over heel wat mogelijkheden ter ondersteuning van keyboard input.
- De klasse die hiervoor gebruikt wordt is de `Scanner` klasse van de `java.util` package.
- Een *package* is een bundeling van klassen

# GEBRUIK SCANNER KLASSE

- In het begin van het programma de juiste klasse importeren:

```
import java.util.Scanner;
```

- Maak een object van de Scanner Klasse

```
Scanner keyboard =
  new Scanner (System.in)
```

- Lees data (bvb een `int` of een `double`)

```
int n1 = keyboard.nextInt();
double d1 = keyboard,nextDouble();
```

# DEMO KEYBOARD INPUT

- **https://youtu.be/nltxWGcjkTs**

```
run:
Geef twee gehele getallen in
gescheiden door één of meerdere spaties
45 66
Je gaf de volgende getallen in: 45 en 66
Geef nu twee kommagetallen in.
12.5 3.5
Je gaf de volgende getallen in:  12.5 en 3.5
Geef nu twee woorden in
Fred Help
Je gaf de volgende woorden in: "Fred" en "Help"
Geef nu een lijn tekst in
Hallo wie ben ik
Je gaf de volgende tekst in: "Hallo wie ben ik"
BUILD SUCCESSFUL (total time: 32 seconds)
```

# SCANNER CLASS METHODEN

- Scanner_object.next()
- Scanner_Object.nextLine()
- Scanner_Object.nextInt()
- Scanner_Object.nextDouble()
- Scanner_Object.nextFloat()
- Scanner_Object.nextLong()
- Scanner_Object.nextByte()
- Scanner_Object.nextShort()
- Scanner_Object.nextBoolean()
- Scanner_Object.useDelimiter(Delimeter_word)

# `nextLine()` METHODE

- De `nextLine()` methode leest
  - De rest van de huidige lijn
  - Ook indien deze leeg is

# `NEXTLINE()` METHODE

– Voorbeeld– gegeven de volgende declaraties

```
int n;
String s1, s2;
n = keyboard.nextInt();
s1 = keyboard.nextLine();
s2 = keyboard.nextLine();
```

– Veronderstel de volgende input

`n` is `42`

maar `s1` is gelijk

aan de lege string

```
42
and don't
you
forget it.
```

UNIVERSITEIT GENT

# OUTLINE

- Variabelen en expressies
- De string klasse
- Keyboard en Scherm I/O
- **Documenteren van code en programmeerstijl.**
- GUI: JFrame en JOPtionPane

# COMMENTAAR

- Een commentaar begint met //.
- Alles na dit symbool tot en met het einde van de lijn wordt beschouwd als commentaar en wordt genegeerd

```
double radius; //in centimeters
```

# COMMENTAAR

- Een commentaar kan ook beginnen met `/*` en eindigen met `*/`
- Alles tussen deze symbolen wordt beschouwd als commentaar en wordt genegeerd door de compiler

```
/**
This program should only
be used on alternate Thursdays,
except during leap years, when it should
only be used on alternate Tuesdays.
*/
```

# COMMENTAAR

- Een javadoc commentaar begint met `/**` en eindigt met `*/`.
- Javadoc commentaar kan automatisch geëxtraheerd worden uit het programma.

```
/** method change requires the number of
  coins to be nonnegative */
```

# INSPRINGEN CODE

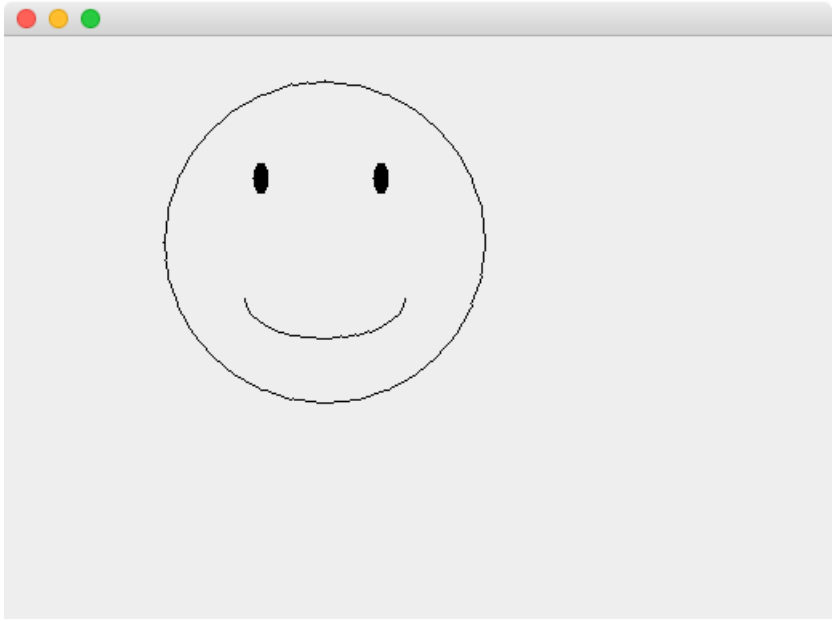
- Het inspringen van code heeft geen invloed op het gedrag van een programma.
- Het inspringen van code helpt bij het lezen van code aangezien het duidelijk de verschillende blokken weergeeft.
- Het inspringen van code moet consistent toegepast worden.
- Inspringen wordt ook gebruikt voor statements die niet op één lijn passen

# OUTLINE

- Variabelen en expressies
- De string klasse
- Keyboard en Scherm I/O
- Documenteren van code en programmeerstijl.
- **GUI: JFrame en JOPtionPane**

# DEMO JFRAME KLASSE

- HappyFaceJFrame
- https://youtu.be/szQWFjs2c9g

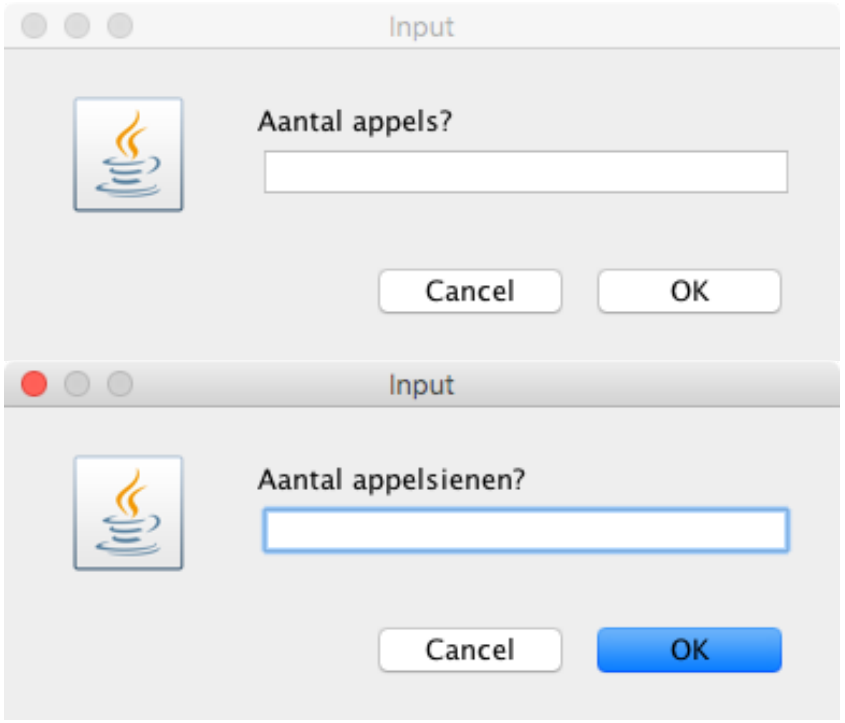


# UITLEG CODE

- HappyFaceJFrame extends JFrame
- setSize(600,400)
  - bepaalt de breedte en hoogte van het JFrame window
  - Methode van JFrame klasse
- setDefaultCloseOperation(EXIT_ON_CLOSE)
  - Bepaalt wat er gebeurt indien men window sluit
  - Methode van JFrame klasse
- main methode
  - Maakt JFrame object adhv constructor
  - Zorgt ervoor dat JFrame zichtbaar is

# DEMO JOPTIONPANE

- https://youtu.be/tH4dZTvPm5Q




# UITLEG CODE

- Dialoogvenster met input:
  JOptionPane.showInputDialog(“Aantal appels?”)
- Tekst omzetten naar getal:
  Integer.parseInt(appelString);
- Dialoogvenster zonder input:
  JOptionPane.showMessageDialog(null, “…”)
- GUI Programma afsluiten
  System.exit(0)