2IP65

Inleiding Programmeren

Kees Huizing
instructeurs & studentassistenten

Fall10

Overview

- 8 lectures
- 15 instructions by instructors and student assistants
- 8 homework assignments (HAs) *count for 30%*
- 1 examination *counts for 70%*
- you have to make at least 6 HAs
- deadline for HAs is usually around next lecture
- you have to make some designated exercises from the instructions

Other info

Kees Huizing, HG 5.39, tel. 4120
contact2ip65@gmail.com
Please don't use other addresses to contact me
your instructor, one of
Michael Franssen – Kees Hemerik – Meilof Veeningen
your student assistants

- instruction on Wednesday and Thursday
- HAs via Peach: peach.win.tue.nl

Locations & Groups

groepjes	Wednesday hour 7+8	instructeur	Thursday hour 3+4	student-assistent	student-assistent
1+2	Ma 1.44	Michael Franssen	Helix west 1.91	Tom Boshoven	Nick van der Veeken
3+4	Pav L10	Kees Hemerik	Helix oost 4.91	Tom Vrancken	Ron Vanderfeesten
5+6	IPO 0.98	Meilof Veeningen	Helix oost 3.91	Thorstin Crijns	Paul Wagener

Resources

- slides of lectures (put on studyweb after lecture)
- reader (put on studyweb after lecture)
- NetBeans program (on your computer, can be downloaded)
- Exercises (handed out at instruction, on studyweb)
- Homework assignment (on Peach)
- book for further reading (optional):
  Learning Java by Niemeyer and Knudsen. O'Reilly.

What to do on/before first instruction

- Register with Peach (http://peach.win.tue.nl)
- Register with Studyweb
- Download NetBeans if not on your computer

Programming is difficult

- computers don't speak
- formal and precise

your main challenge

- prepare for “all” possibilities
- complexity of size
- flexibility (maintenance)

} Software Engineering

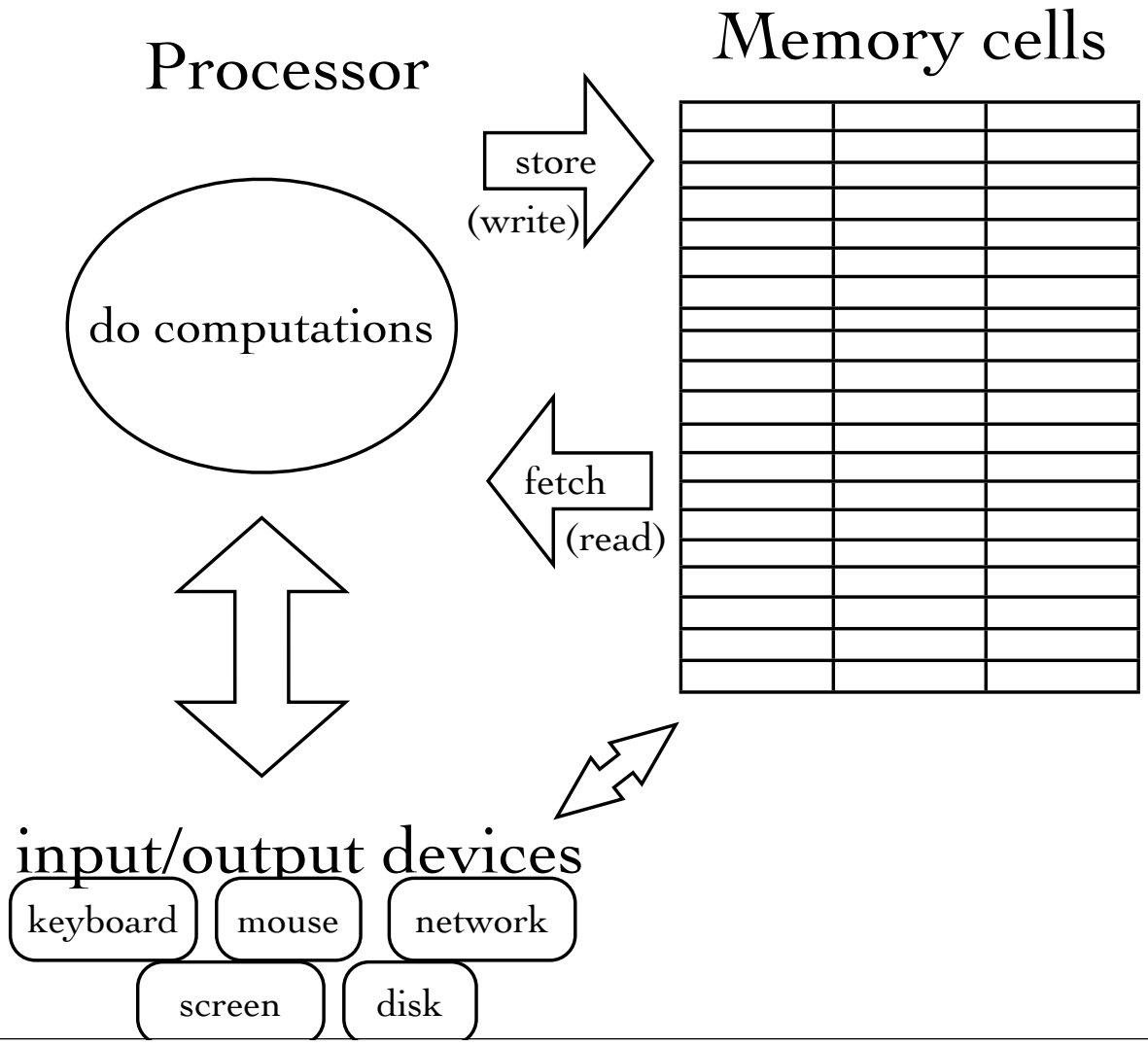
Three T's

- Taal
  - how do you write something in Java?
- Tools
  - how do you make it into a working program
  - what is there that you can use?
- Toil
  - you have to do it yourself!

Parts of a computer

- several input/output devices
  (*randapparatuur*)
  display, keyboard, disk, network, ...
- CPU (processor): the thing that computes
- memory (RAM, *werkgeheugen*)

How does it work?



Only a calculator?

- Processor has a limited set of *instructions* fixed in the electronic circuits

  How to perform complex tasks beyond the limited set of instructions?

- Instructions are made for computers, not for humans, e.g.,
  read memory cell 12345, add 1 to it, and write it back

Two Tricks

1. The program trick: store long sequences of instructions in memory

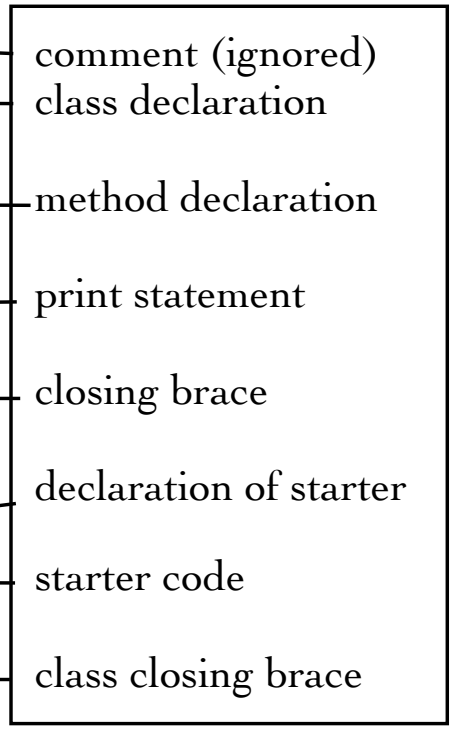
2. The translation trick: translate human readable programs *automatically* into sequences of machine readable instructions

Java

- Offers a high-level view on this basic computer model (memory, instructions); we call this type of languages *imperative* languages
- Offers a good way to organize your code (i.e., programs), it is *object-oriented*

A Java Program

```
// een eenvoudig Java programma
public class Groeter {

  void groet() {

    System.out.println("Hallo daar");

  }

  public static void main(String[] args){
    new Groeter().groet();
  }
}
```



This program puts the text "Hallo daar" on the screen

Running a Java program

1. type in text, containing at least one class, e.g. Groet
2. save in file with the name of the class: Groet.java
3. compile file (creates class file Groet.class)
4. run (= execute) class file

- the compiler does the following:
  - it checks your program for errors
  - it translates your source code into byte-code, a more efficient format

What you need

1. An editor to write Java programs
2. A compiler to translate to byte code
3. A Java system ("Virtual Machine") to run the code

- Luxurious editor integrated with 2 and 3: IDE (integrated development environment)
- Recommended: NetBeans

Calculations

- example *expressions*:

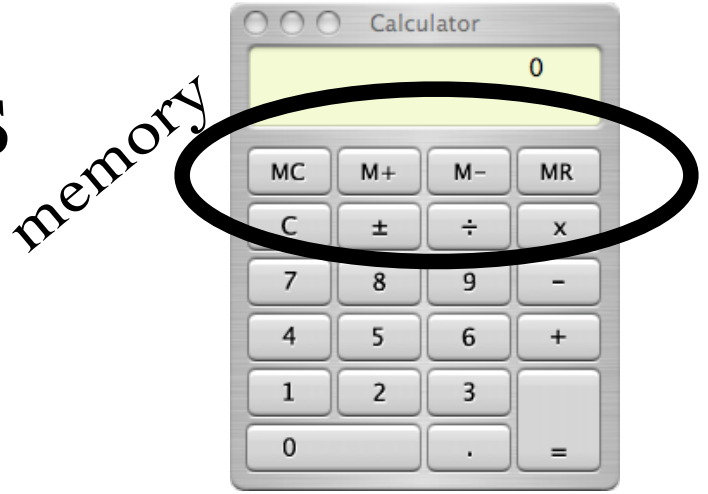
```
10*3 + 6 - 8/4          (= 34)

"apen" + "kool"         (= "apenkool")

Math.sqrt(36)           (= 6)
```

Arithmetic Operators

	description	example expression	result
*	multiplication	5 * 2	10
/	integer division	5 / 2	2
	ordinary division	5.0 / 2	2.5
+	arithmetic addition	1+1	2
	string concatenation	"1"+"1"	"11"
–	subtraction	5-2	3
	minus	-(5-2)	-3
%	remainder or modulo (rest bij deling)	10 % 5 7 % 5	0 2

- for x^y (machtsverheffen), use Math.pow(x,y)
- for logical operators, see later

Variables



- named places to store values
- have a *type*: int, double, String
- have to be declared:

```
int x;
int y;
```

- used in expressions by name: `x + 3`
- changed by *assignment*: `y = x + 3;`

Variables

```
public class Variables {
  double bedrag;
  double rente;   //percentage

  void rente() {
    bedrag = 1200.00;
    System.out.println("bedrag: " + bedrag);

    rente = 3.0;
    bedrag = bedrag * (100+rente)/100;
    System.out.println("na een jaar: " + bedrag);

    bedrag = bedrag * (100+rente)/100;
    System.out.println("na twee jaar: " + bedrag);
  }

  public static void main(String[] args){
     new Variables().rente();
  }
}
```

declaration
declaration

assignment

prints
`bedrag: 1200.0`

prints
`na een jaar: 1236`

prints
`na twee jaar: 1273.8`

creates object and
starts method rente

Input

- input = information from user to executing program
- two alternatives:
  1. reads from the console (the text window where also output appears
  2. create a small window (a *∂ialog*) where user can type some text

Input example 1

necessary to use Scanner

```
import java.util.Scanner;

class InputExample {
   Scanner scanner;
   double bedrag;
   double rente;

   public void renteAndInput() {
      scanner = new Scanner(System.in);
      rente = 3.0;
      System.out.print("beginsaldo? ");

      bedrag = scanner.nextDouble();

      bedrag = bedrag * (100 +rente/100);
      System.out.println("na 1 jaar is uw saldo "+bedrag);
   }

   public static void main(String[] args) {
      new InputExample().renteAndInput();
   }
}
```

create scanner and connect to consele

read a number from input

Console

beginsaldo? **1200**
na 1 jaar is uw saldo 1236.0

typed by user

Input example 2

```
import java.util.Scanner;

class InputExample {
   Scanner scanner;
   String name;
   // String answer; // not needed anymore
   int age;

   public void readNameAndAge() {
      scanner = new Scanner(System.in);
      System.out.print("Name? ");
      name = scanner.nextLine();

      System.out.print("Age? ");
      age = scanner.nextInt();

      System.out.println("Hello "+name+
         ", you are "+age+" springs young");
   }

   public static void main(String[] args) {
      new InputExample().readNameAndAge();
   }
}
```

necessary to use Scanner

create scanner and connect to consele

read next line from input

read an int

Console

```
Name? Kees
Age? 30
Hello Kees, you are 30 springs young
```

Input example (alt)

```
import java.util.Scanner;

class InputExample {
   Scanner scanner;
   String name;
   String answer;
   int age;

   public void readNameAndAge() {
      scanner = new Scanner(System.in);
      System.out.print("Name? ");
      name = scanner.nextLine();
      System.out.print("Age? ");
      answer = scanner.nextLine();
      age = Integer.parseInt(answer);

      System.out.println("Hello "+name+
         ", you are "+age+" springs young");
   }

   public static void main(String[] args) {
      new InputExample().readNameAndAge();
   }
}
```

necessary to use Scanner

declare Scanner object

create scanner and connect to consele

read next line from input

ditto

turn String into an int

Console

```
Name? Kees
Age? 29
Hello Kees, you are 29
springs young
```

Console input – overview

- to use Scanner class: **`import.util.Scanner;`** or **`import.util.*;`**
- declare Scanner object: **`Scanner scanner`**; (or other name)
- create Scanner object and connect to keyboard ("standard input"):
  `scanner = new Scanner(System.in);`
- read words with command depending on data type
- when no input is available (nothing typed or no Return typed yet)
  scanner **`waits`** for input
- when input is finished, **`scanner.hasNext()`** is *false*

Console input – scanner commands

read complete line as String	... = scanner.nextLine();
read word as String	... = scanner.next();
read word as int	... = scanner.nextInt();
read word as double	... = scanner.nextDouble();
check if input is not closed	if (scanner.hasNext())

Input with dialog

```
import javax.swing.*;

public class InputExample {
   String antwoord;
   int leeftijd;

   void readNameAndAge() {
     antwoord = JOptionPane.showInputDialog(
         null, "Hoe heet u?" );

     System.out.println("naam: "+antwoord);

     antwoord = JOptionPane.showInputDialog(
         null, "Hoe oud bent u?" );

     leeftijd = Integer.parseInt(antwoord);

   }

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

necessary to use JOptionPane

turns String into an i

Problem

- We want to have different reactions to different situations, e.g., input values

How?

Choice

- Aim: make a choice between several statements
- Means: if-construct
- Meaning: execute statements only when condition is true

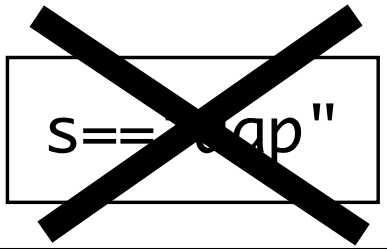
Choice – example

```
int prijs;
System.out.println( "wat is de prijs?" );
prijs = scanner.nextInt();

if ( prijs < 25 ) {
  System.out.println( "Ik koop 'm!");
} else {
  System.out.println( "Te duur" );
}
```

Boolean Operators

operator	description	example expression	result
< >	less than greater than	2 < 3 2 > 2	true false
<= >=	less than or equal to greater than or equal to	2 <= 2 3 >= 5	true false
== !=	equal to not equal to	1 == 1 1 != 1	true false
&&	logical and	0 < 2 && 2 < 3 0 < 2 && 3 < 2	true false
||	logical or	0 < 2 || 3 < 2 2 < 0 || 3 < 2	true false
!	negation, logical not	! true ! (2<0)	false true

- do not mix up `=` (assignment) and `==` (comparison)
- do not use `==` to compare ***Strings***
  use instead: `s.equals("aap")`



Choice – overview (1)

1. general form:

```
if ( condition ) {
   ...   then-part
} else {
   ...   else-part
}
```

2. one branch only:

```
if ( condition ) {
   ...   then-part
}
```

3. alternatives:

```
if ( A ) {
   ... executed if A is true
} else if ( B ) {
   ... executed if B is true and A is not true
} else if (C ) {
   ... executed if C is true and neither A nor B are true
} else {
   ... executed if A nor B nor C is true
}
```

Choice – overview (2)

4. nesting (tree-like branching) :

```
if ( A1 ) {
    if ( B1 ) {
        ... A1 true and B1 true
    } else {
        ... A1 true and B1 false
    }
} else {
    if ( B2 ) {
        ... A1 false and B2 true
    } else {
        ... A1 false and B2 false
    }
}
```

Choice – nesting example

4. nesting (tree-like branching) :

Don't exaggerate, nested ifs get easily very confusing;

use comments as in the example

```
if ( x < 0 ) {
    if ( y == 0 ) {
        ...
    } else {
        ...
    }
} else {      // x >= 0
    if ( x < 5 ) {
        ...  // 0 <= x < 5
    } else {
        ...  // x >= 5
    }
}
```