2ip90
Lecture 3

I/O

Arrays

for loop

for-each loop

do-while loop

ArrayList

I/O

- Input/Output (I/O) is communication from program with "outside world", in particular the user (≠ programmer)
  - console: special text window or pane that is used by Java machine
  - dialogs: windows created for a particular communication
  - GUI: graphical user interface (later)

Output

- console output:
  - `System.out.println("x is "+x);`
  - without new line: `System.out.print("x is "+x);`
  - numeric arguments are automatically converted to their textual decimal representation
- dialog output:

```
javax.swing.JOptionPane.showMessageDialog
    (null, "boe!");
```

Console Input

- scanner.next(); scanner.nextInt(); scanner.nextDouble();

  every next-command reads an input word and advances to the next word

  - import java.util.Scanner;
  - Declare variable: Scanner scanner; (or any other name)
  - Create and connect object: scanner = new Scanner(System.in);
  - scanner.hasNext() is true when there is input left (always true in interactive situation, not always true when reading from file (such as in Peach)
  - scanner.hasNextInt(), hasNextDouble() etc. are true when such a number is on top of input

Scanner

next() reads next word (whitespace separated) and gives a String, without surrounding whitespace

nextInt() ditto, gives an int

nextDouble(), next... ditto, gives a double (or type ...)

hasNext() is true if there is still more than whitespace on input (never false when reading from keyboard, usually)

hasNextInt() is true if there is still a word on input and that word is an integer number

hasNextDouble(), hasNext... ditto, gives a double (or type ...)

Scanner (ctd)

- Scanner object splits input into tokens
- token is a "word": maximal sequence of non-whitespace characters
- only after user types an Enter, the input line is offered to Scanner
- when user types ahead of program, input is buffered
- NB: what separates the words (tokens) can be defined by programmer

Dialog input

- String s = javax.swing.OptionPane.showInputDialog(null, “question”);
- what user types in dialog will be put into s

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 int

Input

- console input: scanner.next(); scanner.nextInt(); scanner.nextDouble();

  every next-command reads an input word and advances to the next word

  - import java.util.Scanner;
  - Declare variable: Scanner scanner; (or any other name)
  - Create and connect object: scanner = new Scanner(System.in);
  - scanner.hasNext() is true when there is input left (always true in interactive situation, not always true when reading from file (such as in Peach)
  - scanner.hasNextInt(), hasNextDouble() etc. are true when such a number is on top of input

- dialog input: JOptionPane.showInputDialog(null, "type wat");

Scanner

next() reads next word (whitespace separated) and gives a String, without surrounding whitespace

nextInt() ditto, gives an int

nextDouble(), next... ditto, gives a double (or type ...)

hasNext() is true if there is still more than whitespace on input (never false when reading from keyboard, usually)

hasNextInt() is true if there is still a word on input and that word is an integer number

hasNextDouble(), hasNext... ditto, gives a double (or type ...)

NB: what separates the words (tokens) can be defined by programmer

Remarks

idea of += works with many operators, such as *=, -=, and %=

- i++ and i+=1 are shorthand for i=i+1
- don't mix them! i=i++ is wrong!
- It is in some cases allowed to write the body of a loop or the branch of an if without braces ({ }).
  Don't do this. It easily leads to errors.

Control and data

- Two fundamental concepts of programming are control and data. Data are values stored in variables. Information is data with meaning. Control determines what you do with the data.

- A program executes the statements (lines or parts of it) one after the other. The order (called control) is determined by:

  1. the order in the text

  2. control structuring

- We have seen some ways to structure control: put statements after each other, make while-loops, make if-structures

- Can we structure data too?

Simple structuring of data: arrays

- An array is a sequence of numbered variables

- Instead of

```
int student1;
int student2;
int student3;
student1 = 7;
student2 = 5;
student3 = 3;
```

- you can do:

```
int[] students;
students = new int[3];
students[0] = 7;
students[1] = 5;
students[2] = 3;
```

declaration
creation: gives 3 elements
last one is `students[2]`

Arrays

- Number between [] is called index
- In Java, all arrays start with 0. Last index is number of elements - 1
- Declaration and creation are separate; often, you don't know directly how many variables you need, e.g., asked as input from the user
- declaration and creation can be combined, though:

```
int[] students = new int[3];
```

Arrays - example

```
int[] students = new int[3];
students[0] = 7;
students[1] = 5;
students[2] = 6;

// bereken som
double som = 0.0;
int i = 0;
while (i<3) {
  som = som + students[i];
  i=i+1;
}
double average = som/3;
```

Intermezzo

- Now you can write any program
- that is, any computation or algorithm (method) that can be precisely described
- Java with arrays and loops is equivalent in power to Turing machines
- any computation that can be precisely described can be described by a Turing machine (Church-Turing thesis)
- since "any computation..." can not be formally defined, this is a conjecture

for-loop

- when you know the number of iterations (number of times you execute the loop-body) on beforehand
- when you want to do something for each member of a set (range) of values

```
for (int i = 0; i<100; i=i+1) {
  System.out.println(
     "Ik mag de accolades om loop-body's niet vergeten"
  );
}
```

for-loop - syntax

```
for ( <initialisation>; <guard>; <increment> ) {

      <statements>

}
```

- <initialisation>: starting assignment to counter
- <increment>: statement that "increases" counter (i=i+1, j=j-1, k=k+x, etc.)
- <guard>: boolean expression as in while

do-while loop

- when you have to do something at least once

```
double balance;
String choice;
// init balance
do {
  System.out.println("Type amount for deposit.");
  deposit = sc.nextDouble();
  balance = balance + deposit;
  System.out.println("New balance is " + balance + ".");

  System.out.println(
       "Do you want to make a deposit? Type y/n");
  choice = sc.next());
} while (choice.equals("y"));
```

do-while loop

- when you have to do something at least once

```
int som = 0;
int cijfer = 0;

do {
    som = som + cijfer;
    System.out.println("type cijfer, -1 om te stoppen");
    cijfer = scanner.nextInt();
} while (cijfer > 0);
```

for loop - example

```
int[] students = new int[3];
students[0] = 7;
students[1] = 5;
students[2] = 6;

// bereken som
double som = 0.0;

  int i = 0;
  while (i<3) {
    som = som + students[i];
    i++;
  }

double gemiddelde = som/3;
```

Flexibility - example

Using the number 3 is inflexible

Make the program work for any number of students

```
System.out.println("Geef aantal studenten: ");
int n = scanner.nextInt();

int[] students = new int[n];
for (int i=0; i<n; i=i+1) {
    students[i] = scanner.nextInt();
}


// bereken som
double som = 0.0;
for (int i=0; i<n; i=i+1) {
    som = som + students[i];
}
double gemiddelde = som/n;
```

beware: division does not work when som is an int

using length of arrays

- following program works for students array of any length not depend on

```
System.out.println("De volgende studenten zitten  onder het gemiddelde:");
for (int i=0; i<students.length; i=i+1) {
  if (students[i] < gemiddelde) {
    System.out.println(i);
  }
}
```

- Advise: use .length whenever possible

for-each loop

- often each element of an array (or another collection) has to be visited
- for-each loop gives exactly this without need for index variable
- for-each works for all kinds of collections, such as lists, sets, trees (see later)

Improvement - example

students.length gives you the number of elements in the array

works safer than the variable n

Note: trying to access an array element that doesn't exist, is an error

```
System.out.println("Geef aantal studenten: ");
int n = scanner.nextInt();

int[] students = new int[n];
for (int i=0; i<n; i=i+1) {
  students[i] = scanner.nextInt();
}

students[0] = 7; students[1] = 5; students[2] = 6;
// bereken som
double som = 0.0;
for (int i=0; i< students.length; i=i+1) {
  som = som + students[i];
}
double gemiddelde = som/students.length;
```

This works for any number of studensts; only the input has to be adapted

for-each loop – example

```
int[] students = new int[3];
students[0] = 7;
students[1] = 5;
students[2] = 6;

// bereken som
double som = 0.0;

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

double gemiddelde = som/3;
```

read: "for each student in students"

for-each loop — syntax

```
for ( <type> <var> : <collection> ) {
    <statements>
}
```

- <type>: member (element) type of collection
- <collection>: array or other collection with members of type <type>
- <var>: variable that will contain the members of the collection

The devious division

- som/3 doesn't work when som is an int
Why?

There are two versions of /

1. when one of the arguments is a double:
   ordinary division

2. when both arguments are int:
   integer division

   e.g., 7/2 = 3

- integer division can be handy in programming; it is confusing to have one symbol for two operations

ArrayList

ArrayList: Growing and shrinking array

- *declaration*
  ArrayList<String> a;

  comparison to arrays
  String[] a;

- *creation*
  a = new ArrayList<String>();

  a = new String[...];

- *appending*
  a.add("aap");

  –

- *insertion (elements after i are shifted)*
  a.add(i, "jet");

  –

- *change*
  a.set(0, "AAP");

  a[0] = “AAP”;

- *inspection*
  s = a.get(0) + a.get(1);

  a = a[0] + a[1];

- *deletion*
  a.remove(i);

  –

ArrayList: more

- *visit all elements*

```
for (String s : a) {
    ... s ...
}
```

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

```
for (String s : a) {
    ... s ...
}
```

- *number of elements*

  a.size( )

  a.length

- for primitive types (int, double, boolean, …)
  use **Integer**, **D**ouble, **B**oolean:

```
ArrayList<Integer> a;
 a = new ArrayList<Integer>();
```

ArrayList: example

```
import java.util.*; // for Scanner and ArrayList
...
int n;
ArrayList<String> words;
String word;
double totallength = 0;
double averagelength;

words = new ArrayList<String>( );
do {
   word = scanner.next( );
   totallength += word.length( );
   words.add(word);
} while (scanner.hasNext( ));  // doesn't work with keyboard input
// while( !scanner.hasNext("__END__") ) stops when __END__ is encountered

averagelength = totallength / words.size( );

for (String w : words) {
   if (w.length > averagelength) {
        System.out.println(w+" is above average");
   }
}
```

Hoe lang is een Chinees

- `length`
  number of members of an array

- `length()`
  number of characters in a String

- `size()`
  number of members of an ArrayList (and other Collections)