2IP65
Lecture 3

Arrays
for loop
for-each loop
do-while loop
ArrayList

Remarks

- i++ and i+=1 are shorthand for i=i+1

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

- 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++;
}
double gemiddelde = som/3;
```

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();
}

students[0] = 7; students[1] = 5; students[2] = 6;
// 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, gives nasty errors

```
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;
- *creation*
  a = new ArrayList<String>();
- *appending*
  a.add("aap");
- *insertion (elements after i are shifted)*
  a.add(i, "jet");
- *change*
  a.set(0, "AAP");
- *inspection*
  s = a.get(0) + a.get(1);
- *deletion*
  a.remove(i);

comparison to arrays

String[] a;

a = new String[...];

–

–

a[0] = “AAP”;

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

–

ArrayList: more

- *visit all elements*

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

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

- *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

```
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( ));

averagelength = totallength / words.size;

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