Introductory Programming
Imperative Programming III, sections 3.6-3.8, 3.0, 3.9

Anne Haxthausen[a]

IMM, DTU

1. Loops (**while**, **do**, **for**) (sections 3.6 – 3.8)
2. Overview of Java statements (learnt so far)
3. Program development (sections 3.0, 3.9)

a. Parts of this material are inspired by/originate from a course at ITU developed by Niels Hallenberg and Peter Sestoft on the basis of a course at KVL developed by Morten Larsen and Peter Sestoft.

Control flow of programs

control flow = order in which statements are executed

- Unless otherwise specified, a program is executed *linearly*:
  - Java executes the statements of the `main` method one by one
- The execution order can be controlled by two special kinds of statements:
  - *conditionals* (valg-sætninger) give the choice between two or more statements: **if**, **if-else** and **switch**.
  - loops (løkker/gentagelses-sætninger) repeat a statement: **while**, **do** and **for**.

Loops

Loops are used for repeating a statement.

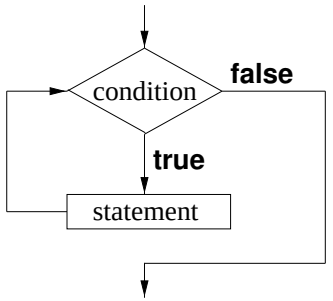
kind of loop	useful for repetitions of the form
while	as long as ... do ...
do	do ... as long as ...
for	do ... a fixed number of times

While loops: generally

General format:

while (*condition*)
  statement

Effect:



While loops: example

```
final int LIMIT = 7;
int count = 2;

while (count <= LIMIT)
{
  System.out.println(count);
  count = count + 1;
}
```

gives the following output:

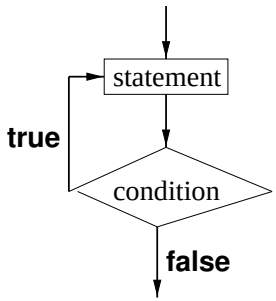
```
2
3
4
5
6
7
```


Do loops: generally

General format:

do *statement*
while (*condition*) ;

Effect:



Do loops: example

do loops can e.g. be used for repeating input until the user has given a correct input.

Example:

```
//input a 'y','Y', 'n' or 'N'

  char c;

  do
  {
     System.out.print("Enter (y/n)? ");
     c = Character.toUpperCase(Keyboard.readChar());
  }
  while (! (c=='Y' || c=='N'));
```

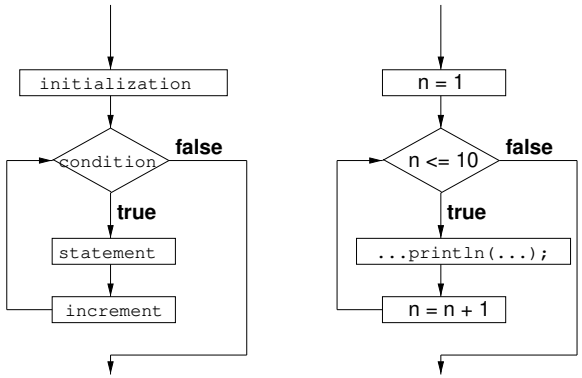

For loops: example

```
for (int n=1; n<=10; n = n + 1)
  System.out.println(n + " squared is " + (n * n));
```

gives the following output:

```
1 squared is 1
2 squared is 4
3 squared is 9
4 squared is 16
5 squared is 25
6 squared is 36
7 squared is 49
8 squared is 64
9 squared is 81
10 squared is 100
```

For loops: generally + example



For loops: generally

General format:

for (*initialization* **;** *condition* **;** *increment*)
statement

An *initialization* is usually a declaration of a variable containing an initial value, e.g. `int n=1`
A *condition* is an expression of type **boolean** (e.g. `n<=10`); can be omitted – is then a shorthand for **true**
An *increment* is usually an assignment (e.g. `n=n+1`) or a list of assignments.

Equivalent to:

initialization;
while (*condition*)
{
statement;
increment ;
}

Nested loops

A loop inside the body of another loop is said to be *nested*. **Example:**

```
final int MAX_ROWS = 7;
for (int row = 1; row <= MAX_ROWS; row++)
{
   for (int star = 1; star <= row; star++)
      System.out.print ("*");
   System.out.println();
}
```

gives the following output:

```
*
**
***
****
*****
******
*******
```


Warning!

What is wrong in the following piece of program?

```
int i = 1;

while (i < 7) { i = i - 1; }
```

Overview of Java statements

Kind of statement	Example
assignment	`x = x + 1;`
if	`if (x != 0) y = 1/y;`
if-else	`if (x != 0) y = 1/x; else y = 0;`

switch	`switch (x) {`
	`  case 0: y = 0; break;`
	`  case 1: y = 117; break;`
	`  default: y = 1/x; break`
	`}`

invocation of method	`System.out.println("Hello");`
block	`{ x = 1; y = 1/x; System.out.print(y); }`
for loop	`for (int i=1; i<100; i=i+1) sum = sum + i;`
while loop	`while (sum < 100) sum = sum + 1;`
do-while loop	`do { sum = sum + 1;} while (sum < 100);`

Statements can only be written inside a method (e.g. in `main`), not in a class.

Program development

The development of a program usually goes through the following four phases:

Analysis: decide the requirements to the program – WHAT should the program accomplish?

Design: decide HOW the program can accomplish its requirements.

Implementation: write the program transforming the design into the chosen programming language

Test: run the program with many different inputs and check that the output is as expected.

You can learn more about this in course *02260 Software Engineering*.

Program development: example (exercise 8)
Requirements from the client

Write a program that (1) reads a password and (2) checks that the length is greater than 4 and less than 9.

Program Development: example
Analysis

- Should the program be interactive (dialog on the screen)?
- What should the program answer if the password is legal?
- What should the program answer if the password is illegal? Should it for instance just answer “Password is illegal”, or should it state in which way it is illegal?

After conferring with the client, we find out that the answers should be:

- Yes.
- ”The length of the password is legal”.
- ”The length of the password is illegal”.

Program Development: example
Design 1

The algorithm can be sketched by pseudocode:

prompt for and read a password;

check password and print result;

Pseudocode is a mixture of code statements and phrases of natural language.

Program Development: example
More detailed design

prompt for and read a password;

if (password is ok) then print "The length of the password is legal"
otherwise print "The length of the password is illegal"

Program Development: example
Implementation

```
import cs1.Keyboard;

public class Password {

public static void main(String[] args) {
    //prompt for and read a password
    System.out.print("Write a password: ");
    String password = Keyboard.readString();

    //Check password and print the result
    int plength=password.length();
    if (plength>4 && plength<9)
      System.out.println("The length of the password is legal");
    else
      System.out.println("The length of the password is illegal");
}}
```


Program Development: example
Test (Danish: afprøvning)

1. Make a *test suite*, i.e. a collection of examples of input and expected output.
2. Run the program with the chosen input and check that the output is as expected. If not, you must re-do the development.

Two kinds of testing:

	internal	external
Context	program text	problem
Kinds of errors	logic	over-seen cases
	wrong initializations of variables	over-seen requirements

Internal test

Kind of statement	cases that should be tested
`if`	Condition false and true
`switch`	each branch
`for`	zero, one or more executions
`while`	zero, one or more executions
`do-while`	one or more executions

Program Development: example

Test: example of a test suite

Test 1

```
Write a password: pass22
The length of the password is legal
```

Test 2

```
Write a  password: yes
The length of the password is illegal
```

Test 3

```
Write a password: course02199
The length of the password is illegal
```