Flow control in a program

v The order of execution of a program's instructions is usually linear
- one statement after another, in sequence
v Some instructions allow you to change this order, deciding:
- whether or not to execute a particular declaration
- run a statement repeatedly, repeatedly
v These decisions are based on Boolean expressions (or conditions)
- which are assessed as true or false

Boolean expressions

v Boolean expressions return true or false.
v Boolean expressions use operators
relational, equality, and logical (AND, OR, NOT)
== equal to // Attention !! x == y is different from x = y! = not equal to
<less than
> greater than
<= less than or equal to
> = greater than or equal to! NOT
&& AND
|| OR
v Examples
x> = 10
(y <z) && (z> t)

Tables of truth

v Boolean algebra is based on truth tables.
vConsidering A and B, eg: ((y <z) && (z> t))
- Both must be true for the expression A && B to be
true.
- A true being is enough for the expression A || B be true.

a	!a
true	false
false	true

a	b	a && b	a || b
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

Ternary operator

v The ternary operator (? :) is also known as the conditional operator.
result = testCondition? valueIfTrue: valueIfFalse
- Evaluates an expression (1st operand) and, if true, the result is equal to the 2nd operand, otherwise the result is equal to the 3rd operand.

```
char code = 'F';
boolean capitalLetter = (code> = 'A') && (code <= 'Z'); System.out.println (capitalLetter? "Yes": "no");
minVal = (a <b)? a: b;
```

Conditional statements

v In Java there are two types of decision / selection statements:
- if
- switch

v The if statement has the following format:

```
if (Boolean expression)
// do_this;
else // optional
// do_this;
```

If decision statement

v We can chain several if statements:

```
if (condition1)
    block1;
else if (condition2)
    block2;
else block3;
```

If a block includes more than one instruction, the block must be delimited by {..}.

Switch selection statement

v The switch statement executes one of several paths (case), depending on the result of an expression

```
switch (expression) {
   case valor1:
block1;
      break;
   case valor2:
block2;
break;
// ...
default:
Final block;
}
```

The result of the expression is searched in the list of alternatives existing in each case, in the order in which they are specified.
If the search is successful, the corresponding code block is executed. If there is a break statement, the execution of the switch ends. Otherwise, all the following options will be executed until break appears or the end of the switch is reached.
If the search is not successful and the default exists, the corresponding code block (Final block) is executed.

Cycles

v Sometimes there is a need to execute instructions repeatedly.
- A set of instructions that are executed repeatedly is called a cycle.
v A cycle can be of the conditional type (while and do ... while) or of the counter type (for).
- We normally use conditional cycles when the number of iterations is unknown and counter-type cycles when we know the number of iterations in advance.

While cycle

v The while cycle runs as long as the cycle condition is true.
- The condition is tested before each iteration of the cycle.

```
while (condition) execute_block;
```
- Example:
```
Scanner sc = new Scanner (System.in); int grade = -1;
while ((grade> 20) || (grade <0)) {
System.out.println ("Insert the student's grade.");
nota = sc.nextInt ();
}
sc.close ();
```

Do-While cycle

v The do ... while cycle runs a first time and then check if it is necessary to repeat.
- The condition is tested at the end of each iteration of the cycle.

of
execute_block;
while (condition);
- Example:
Scanner sc = new Scanner (System.in); int note;
of {
System.out.println ("Insert the student's grade.");
nota = sc.nextInt ();
} while ((grade> 20) || (grade <0));
sc.close ();

---

for cycle

v The for cycle is more general because it supports all situations of repeated execution.

for (initialization; condition; update)
bloco_a_executar;

1. Before the 1st test, do the initialization (only once)
2. Then perform the condition test.
If true, execute the block, if false ends
3. At the end of each term, execute the update part and resume in point 2 above.

For loop (foreach syntax)

v The for cycle, when used with vectors, can have a more succinct form (foreach).
Scanner sc = new Scanner (System.in); double [] a = new double [5];
for (int i = 0; i <a.length; i ++)
a [i] = sc.nextDouble (); for (double el: a)
System.out.println (el); sc.close ();

Break and continue instructions

v We can end the execution of a block of instructions with two special instructions:
- break and continue.
v The break statement allows for the immediate exit of the code block being executed.
- It is normally used in switch but can also be used in cycles.
v The continue statement allows you to end the execution of the current iteration, forcing the passage to the next iteration (i.e. it does not end the cycle).
