for-loop

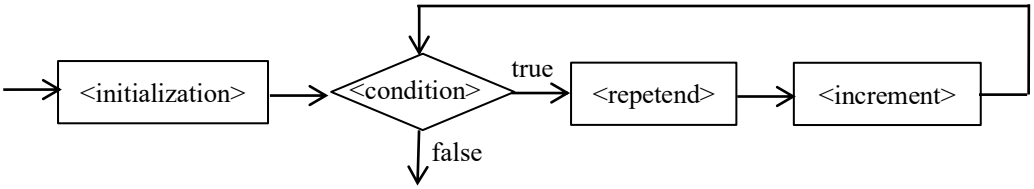
We don't describe the complete for-statement, or for-loop, as it is defined in Java, but just its most used form. The syntax of the for-loop is:

for (<initialization> ; <condition> ; <increment>) <repetend>

where

1. The <initialization> is an assignment, like k= 0. It is executed at the beginning of the for-statement.
2. The <condition> is a boolean expression.
3. The increment is an assignment, like k= k+1 or k= k-1.
4. The <repetend[1]> is a statement — either a single statement or a <block>

The following flow chart shows how the for-loop is executed.



Here are two examples.

```
(1)     for (int k= 9; k >= 0; k= k-1)          // This loop prints the values 9, 8, 7, …, 0
            System.out.println(k);

(2)     int k;                                  // This loop prints the values 0, 1, 2, …, 9
        for (k= 0; k < 10; k= k+1) {
            System.out.println(k);
        }
```

Consider these points about for-loops.

1. Each execution of the repetend is called an iteration. The first iteration is number 0, the second is number 1, and so on.
2. Because k is declared in the <initialization> in for-loop (1), its scope is only the loop. Variable k cannot be used after the loop. It you want to reference k after the for-loop, declare k before the loop, as in (2).
3. The <initialization> can be a sequence of assignments separate by commas, e.g. k= 1, i= 4, c= Color.RED
   The <initialization> can be missing —you can write for (; k < 5; k= k+1) …
4. The <condition> can be missing, in which case it is an infinite loop.
5. The increment can be a sequence of assignments, not just one.
6. Consider for-loop (1) above. Variable k is called the *counter* of this loop. It is possible to change the loop counter in the repetend, but we strongly advise against this. A loop like (1) gives the impression that all the control bookkeeping is done in the first line —initialization, testing for termination, incrementing. Changing k in the repetend is disconcerting at best, ruining what the reader is expecting. Don't do it.
7. It is possible to use the break statement in the repetend. Its execution immediately terminates execution of the for-loop. We advise against this. Changing control using a break statement makes it harder to reason about the loop. If possible, restructure to avoid using it.
8. Execution of a continue statement within the repetend terminates execution of the repetend, so that the <increment> is done next.
9. Loops are best understood (and developed) using loop invariants. See the tutorials on program correctness and loop invariants that are associated with this list of definitions and concepts.

[1] *Repetend* means *the thing to be repeated.* In the 1980's, a 13-year old who was studying Gries's book "The Science of Programming" used the term in an email. From then on, we have used that word.