FLOW OF CONTROL: LOOPS

OOP

Lecture 4

OBJECTIVES

- Design a loop
- Use `while`, `do`, and `for` in a program
- Use the `for-each` with enumerations
- Use assertion checks

JAVA LOOP STATEMENTS: OUTLINE

- The `while` statement
- The `do-while` statement
- The `for` Statement

JAVA LOOP STATEMENTS

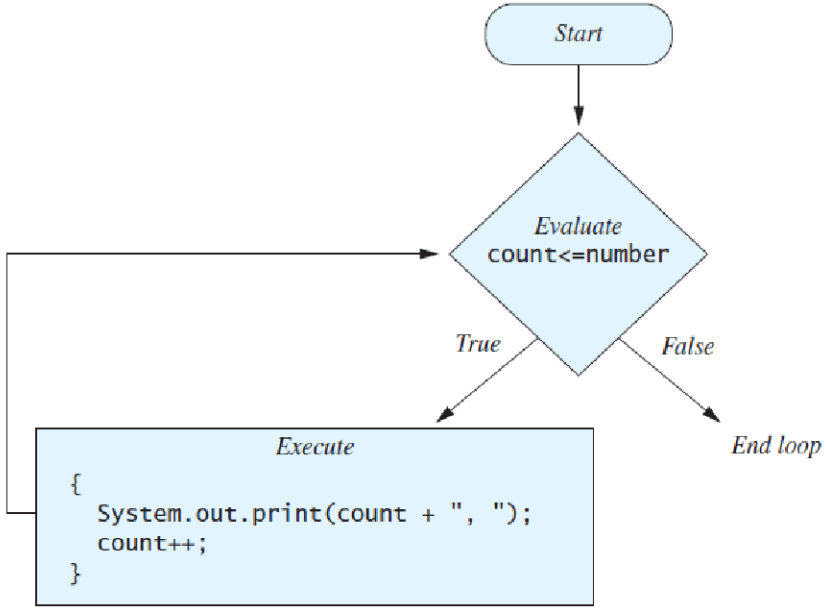
- A portion of a program that repeats a statement or a group of statements is called a *loop.*
- The statement or group of statements to be repeated is called the *body* of the loop.
- A loop could be used to compute grades for each student in a class.
- There must be a means of exiting the loop.

THE `WHILE` STATEMENT

- Also called a `while` loop.
- A `while` statement repeats while a controlling boolean expression remains true.
- The loop body typically contains an action that ultimately causes the controlling boolean expression to become false.

THE WHILE STATEMENT

```
while (count <= number)
{
   System.out.print(count + ", ");
   count++;
}
```



- Syntax

```
while (Boolean_Expression)
      Body_Statement
or
while (Boolean_Expression)
{
      First_Statement
      Second_Statement
      ...
}
```

THE DO-WHILE STATEMENT

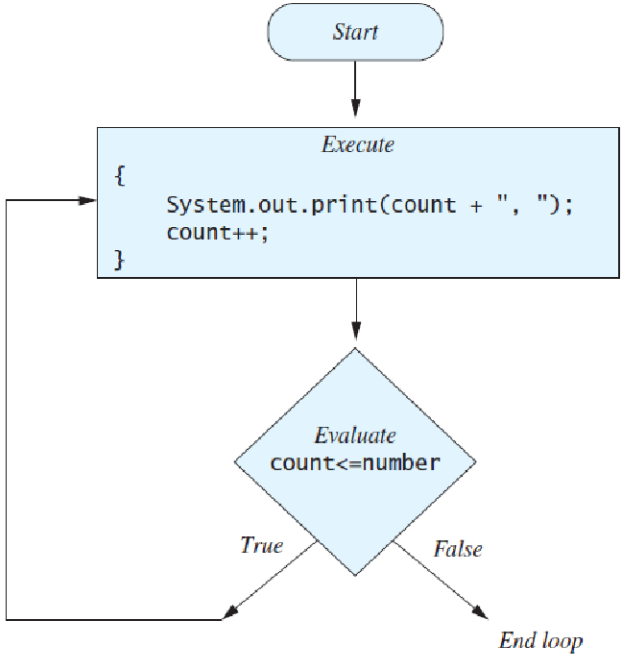
- Similar to a `while` statement, except that the loop body is executed at least once
- Syntax

```
do
    Body_Statement
while (Boolean_Expression);
```

- Don't forget the semicolon!

THE DO-WHILE STATEMENT

```
do
{
   System.out.print(count + ", ");
   count++;
} while (count <= number);
```



THE `DO-WHILE` STATEMENT

- First, the loop body is executed.
- Then the boolean expression is checked.
  - As long as it is true, the loop is executed again.
  - If it is false, the loop is exited.
- Equivalent `while` statement

```
Statement(s)_S1
while (Boolean_Condition)
    Statement(s)_S1
```

INFINITE LOOPS

- A loop which repeats without ever ending is called an *infinite loop.*
- If the controlling boolean expression never becomes false, a `while` loop or a `do-while` loop will repeat without ending.
- Example :
  - while (true) {
    }
  - For(;;) {
    }

THE FOR STATEMENT

- A `for` statement executes the body of a loop a fixed number of times.
- Example

```
for (count = 1; count < 3; count++)
    System.out.println(count);
```

THE FOR STATEMENT

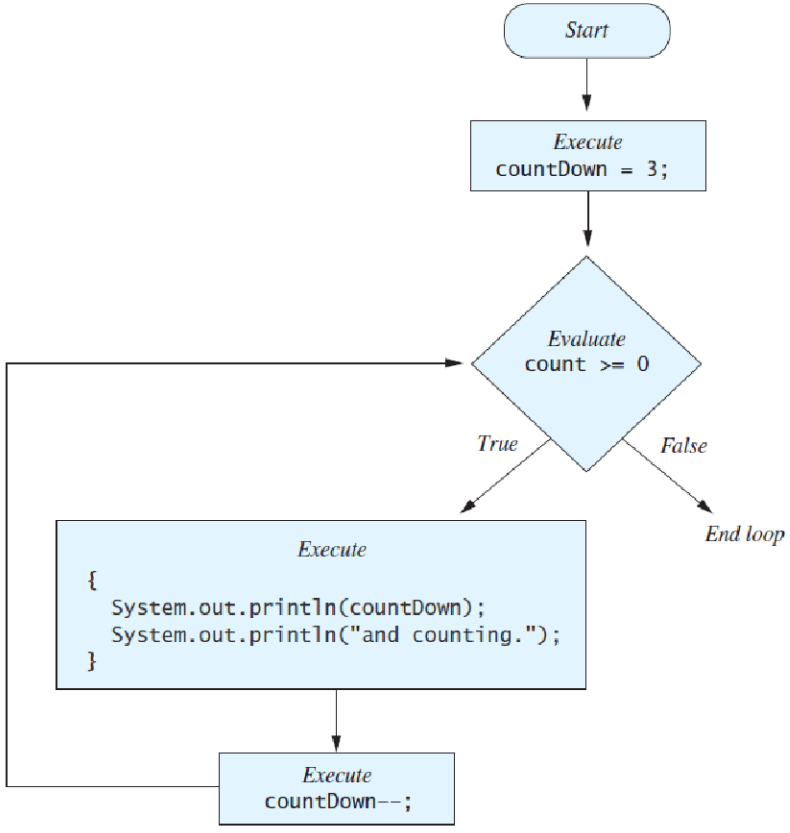
- Syntax

```
for (Initialization, Condition, Update)
    Body_Statement
```

- `Body_Statement` can be either a simple statement or a compound statement in `{}`.
- Corresponding `while` statement

```
Initialization
while (Condition)
    Body_Statement_Including_Update
```

THE FOR STATEMENT




```
for (countDown = 3; countDown >= 0; countDown--)
{
    System.out.println(countDown);
    System.out.println("and counting.");
}
```

THE FOR STATEMENT

- Possible to declare variables within a `for` statement

```
int sum = 0;
for (int n = 1 ; n <= 10 ; n++)
    sum = sum + n * n;
```

- Note that variable `n` is local to the loop

THE FOR STATEMENT

- A comma separates multiple initializations
- Example

```
for (n = 1, product = 1; n <= 10; n++)
    product = product * n;
```

- Only one boolean expression is allowed, but it can consist of &&s, ||s, and !s.
- Multiple update actions are allowed, too.

```
for (n = 1, product = 1; n <= 10;
        product = product * n, n++);
```

PROGRAMMING WITH LOOPS: OUTLINE

- The Loop Body
- Initializing Statements
- Controlling Loop Iterations
- `break` and `continue` statements
- Loop Bugs
- Tracing Variables
- Assertion checks

THE LOOP BODY

- To design the loop body, write out the actions the code must accomplish.
- Then look for a repeated pattern.
  - The pattern need not start with the first action.
  - The repeated pattern will form the body of the loop.
  - Some actions may need to be done after the pattern stops repeating.

INITIALIZING STATEMENTS

- Some variables need to have a value before the loop begins.
  - Sometimes this is determined by what is supposed to happen after one loop iteration.
  - Often variables have an initial value of zero or one, but not always.
- Other variables get values only while the loop is iterating.

CONTROLLING NUMBER OF LOOP ITERATIONS

- If the number of iterations is known before the loop starts, the loop is called a *count-controlled loop.*
  - Use a `for` loop.
- Asking the user before each iteration if it is time to end the loop is called the *ask-before-iterating technique.*
  - Appropriate for a small number of iterations
  - Use a `while` loop or a `do-while` loop.

Controlling Number of Loop Iterations

- For large input lists, a *sentinel value* can be used to signal the end of the list.
  - The sentinel value must be different from all the other possible inputs.
  - A negative number following a long list of nonnegative exam scores could be suitable.

```
90
0
10
-1
```

Controlling Number of Loop Iterations

- Example - reading a list of scores followed by a sentinel value

```
int next = keyboard.nextInt();
while (next  >= 0)
{
     Process_The_Score
     next = keyboard.nextInt();
}
```

SUMMARY

- A loop is a programming construct that repeats an action
- Java has the `while`, the `do-while`, and the `for` statements
- The `while` and `do-while` repeat the loop while a condition is true
- The logic of a `for` statement is identical to the while