Java's `for` Statement

Algorithmic Pattern: The Determinate loop

- We often need to perform some action a specific number of times:
  - Generate 89 paychecks
  - Count down to 0 (take 1 second of the clock)
  - Simulate playing a game of "Let's Make a Deal" 10,000 times
- The *determinate loop* pattern repeats some action a specific number of times

Pattern: Determinate Loop

Problem: Do something exactly n times, where n is known in advance.

Algorithm

```
determine  n
repeat the following n times {
    perform these actions
}
```

Code Example:

```
int sum = 0;
int n = 4;
for(int c = 1; c <= n; c = c + 1) {
  sum = sum + c;
}
// What is sum now?
```

Determinate Loops

- This template repeats a process n times
  - replace comments with appropriate statements

```
int n = /* how often we must repeat the process */
for( int j = 1; j <= n; j = j + 1 ) {
   // the process to be repeated
}
```

- *determinate loops* must know the number of repetitions *before* they begin
  - know exactly how many employees, or students, or whatever that must be processed, for example

General Form
The Java for loop

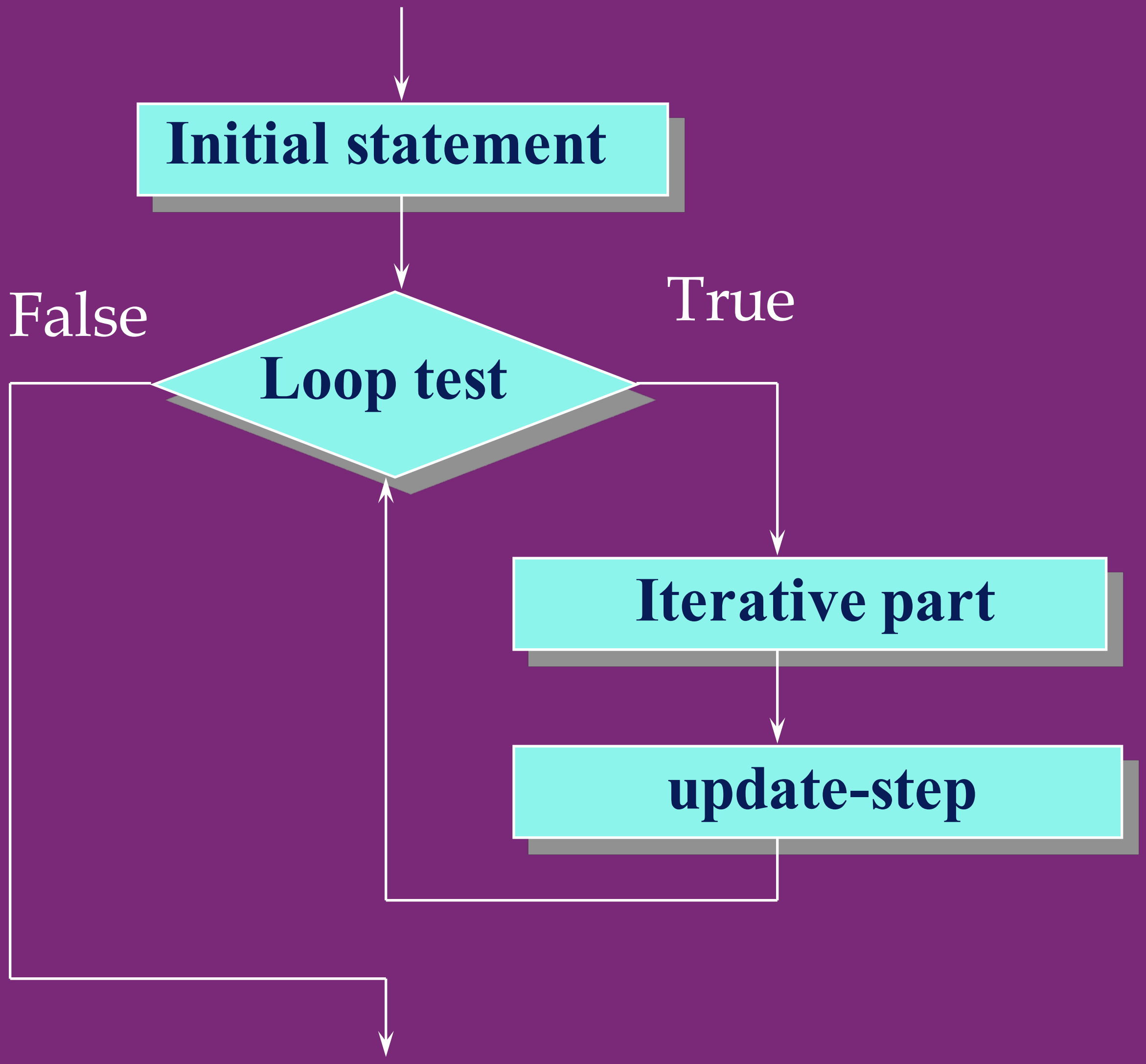
for (*initial statement* **;** *loop-test* **;** *update-step***)** **{**

repeated-part

}

– When a for loop is encountered, the *initial-statement* is executed (used here quite often as `int j = 1`). The *loop-test* evaluates. If *loop-test* is false, the for loop terminates. If *loop-test* is true, the *repeated-part* executes followed by the *update-step*.

Flow chart view of a for loop



Example for loop that produces an average

```
Scanner keyboard = new Scanner(System.in);
double sum = 0.0;
System.out.print("How many do you want to average? ");
int n = keyboard.nextInt();

// Do something n times
for (int j = 1; j <= n; j = j + 1) {
  System.out.print("Enter number: "); // <- Repeat 3
  int number = keyboard.nextInt(); // <- statements
  sum = sum + number; // <- n times
}

double average = sum / n;
System.out.print("Average = " + average);
```

Code Demo: Use the debugger to trace this code

```
int n = 5;
for (int j = 1; j <= n; j = j + 1) {
  System.out.println(j);
}

for (int k = 10; k >= 0; k = k - 2) {
  System.out.println(k);
}
```

Other Incrementing Operators

- It is common to see determinate loops of this form where n is the number of repetitions

```
for( int j = 1; j <= n; j++ ) }
    // ...
}
```

- The unary ++ and -- operators add 1 and subtract 1 from their operands, respectively.

```
int n = 0;
n++; // n is now 1 equivalent to n=n+1; or n+=1;
n++; // n is now 2
n--; // n is now 1 again
```

- The expression `count++` is equivalent to the more verbose `count = count + 1;`

Other Assignment Operators

- Java has several assignment operators in addition to `=` (`-=` and `+=`)

  `j -= 2;` is the equivalent of `j = j - 2;`

  `sum += x;` is the equivalent of `sum = sum + x;`

- What is sum when a user enters 7 and 8?

```
int sum = 0;
int x = 0;
System.out.print("Enter a number: ");
x = keyboard.nextInt(); // user enters 7
sum += x;
System.out.print("Enter a number: ");
x = keyboard.nextInt(); // user enters 8
sum += x;
```