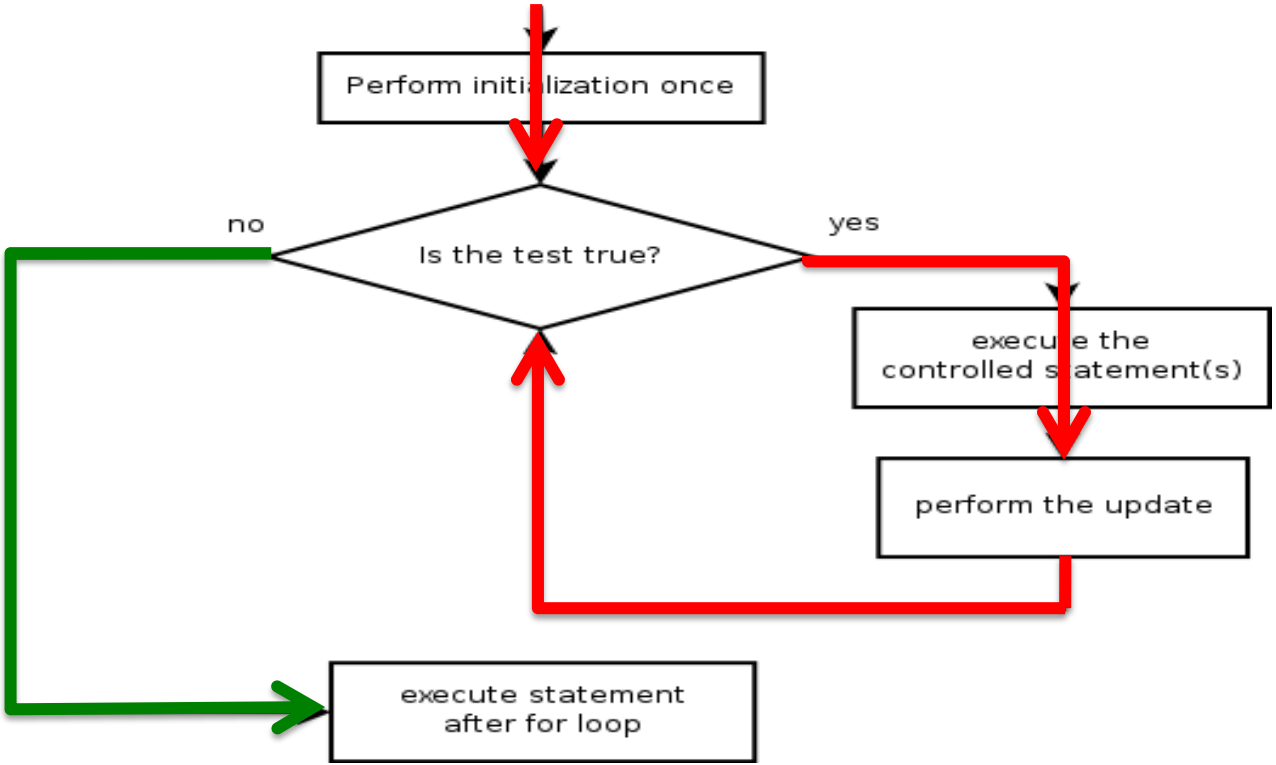
Using the Java For Loop Effectively

The Java `for` loop



Execution of the `for` loop

```
for (int i = 1; i <= 5; i = i+1) {
    out.println(i + " doubled = " + 2 * i);
}
```

Often the for-loop body uses the counter variable

General repetition

```
out.print("I ");
for (int i = 1; i <= 3; i++) {  // repeat 3 times
    out.print("love ");
}
out.println("programming!");
```

The loop's body does not have to use the counter variable

General repetition

```
out.print("I ");
for (int i = 1; i <= 3; i++) {  // repeat 3 times
    out.print("love ");
}
out.println("programming!");
```

Result:
"I love love love programming!"

Computing Series

- A series is a continuing sum of a sequence of terms.
- Series are used to compute a number of constants and functions.
- It’s useful to know how to write a program to compute one.
- Special pattern: Cumulative Sum loop

Computing Series

- A series is a continuing sum of a sequence of terms.
- Series are used to compute a number of constants and functions.
- It’s useful to know how to write a program to compute one.
- Special pattern: Cumulative Sum loop

Computing Series

- A series is a continuing sum of a sequence of terms.
- Series are used to compute a number of constants and functions.
- It’s useful to know how to write a program to compute one.
- Special pattern: Cumulative Sum loop

Computing Series

- A series is a continuing sum of a sequence of terms.
- Series are used to compute a number of constants and functions.
- It’s useful to know how to write a program to compute one.
- Special pattern: Cumulative Sum loop

Basic Series

```
double accumulator = 0;
for (int i = 1; i <= numberOfTerms; i++) {
    accumulator += Term(i)
}
```

Start by initializing an accumulator variable.

Basic Series

```
double accumulator = 0;
for (int i = 1; i <= numberOfTerms; i++) {
    accumulator += Term(i)
}
```

Then have a loop execute for the number of terms to be computed.

Basic Series

```
double accumulator = 0;
for (int i = 1; i <= numberOfTerms; i++) {
    accumulator += Term(i)
}
```

where Term(i) is a function (or expression) that computes the ith term

Basic Series

```
double accumulator = 0;
for (int i = 1; i <= numberOfTerms; i++) {
    accumulator += 1.0/i;
}
```

Here's an example of the Harmonic series:
H = 1 + 1/2 + 1/3 + 1/4 + 1/5 + ...

Expressions for counter

```
int sum = 0;
int topValue = 10000;
for (int i = 2; i <= topValue/4; i=2*i) {
     sum = sum + i;
}
```

The initial and final values for the loop counter variable and the update can be arbitrary numbers or expressions

Expressions for counter

```
int sum = 0;
int topValue = 10000;
for (int i = 2; i <= topValue/4; i=2*i) {
     sum = sum + i;
}
```

The above loop will compute the sum of the powers of two between 2 and 2500

Counting down

```
for (int i = 5; i >= 1; i--) { // or i=i-1
     out.print(i + ", ");
}
out.println("let's go!");
```

The update can use the decrement operator to make the loop count down.

- Be sure to use the right test (> or >= instead of < or <=)

Counting down

```
for (int i = 5; i >= 1; i--) { // or i=i-1
     out.print(i + ", ");
}
out.println("let’s go!");
```

Result:
"5, 4, 3, 2, 1, let’s go!"

Common Errors

```
for (int i = 10; i < 5; i++) {
    out.println("How many times am I printed?");
}
```

Some loops execute 0 times, because of the nature of their initialization and test

Common Errors

```
for (int i = 1; i <= 10; i=i++)
//note: update uses post-incr and assignment together
{
    out.println("Runaway Java program!!!");
}
```

Some loops execute endlessly (or far too many times), because the loop test never fails

- Known as an infinite loop

Common Errors

```
for (int i = 1; i <= 10; i++);
{
    out.println("How many times am I printed?");
}
```

Watch for a misplaced semicolon, which marks the end of the loop body; this results in an empty loop body.