

The `for` loop

Repetition with for loops

- So far, repeating a statement is redundant:

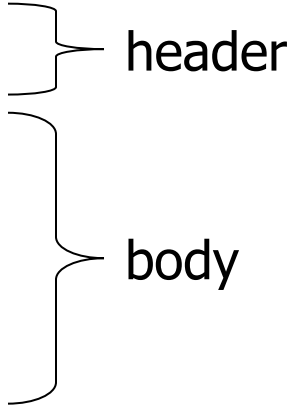
```
System.out.println("Homer says:");  
System.out.println("I am so smart");  
System.out.println("I am so smart");  
System.out.println("I am so smart");  
System.out.println("I am so smart");  
System.out.println("S-M-R-T... I mean S-M-A-R-T");
```

- Java's **for loop** statement performs a task many times.

```
System.out.println("Homer says:");  
for (int i = 1; i <= 4; i++) {    // repeat 4 times  
    System.out.println("I am so smart");  
}  
System.out.println("S-M-R-T... I mean S-M-A-R-T");
```

for loop syntax

```
for (initialization; test; update) {  
    statement;  
    statement;  
    ...  
    statement;  
}
```



header

body

- Perform **initialization** once.
- Repeat the following:
 - Check if the **test** is true. If not, stop.
 - Execute the **statements**.
 - Perform the **update**.

Initialization

```
for (int i = 1; i <= 6; i++) {  
    System.out.println("I am so smart");  
}
```

- Tells Java what variable to use in the loop
 - Performed once as the loop begins
 - The variable is called a *loop counter*
 - can use any name, not just `i`
 - can start at any value, not just `1`

Test

```
for (int i = 1; i <= 6; i++) {  
    System.out.println("I am so smart");  
}
```

- Tests the loop counter variable against a limit
 - Uses comparison operators:
 - < less than
 - <= less than or equal to
 - > greater than
 - >= greater than or equal to

Increment and decrement

shortcuts to increase or decrease a variable's value by 1

Shorthand

variable++;

variable--;

```
int x = 2;
```

```
x++;
```

```
double gpa = 2.5;
```

```
gpa--;
```

Equivalent longer version

variable = **variable** + 1;

variable = **variable** - 1;

```
// x = x + 1;
```

```
// x now stores 3
```

```
// gpa = gpa - 1;
```

```
// gpa now stores 1.5
```

Modify-and-assign

shortcuts to modify a variable's value

Shorthand

variable += **value**;
variable -= **value**;
variable *= **value**;
variable /= **value**;
variable %= **value**;

Equivalent longer version

variable = **variable** + **value**;
variable = **variable** - **value**;
variable = **variable** * **value**;
variable = **variable** / **value**;
variable = **variable** % **value**;

x += 3;

// x = x + 3;

gpa -= 0.5;

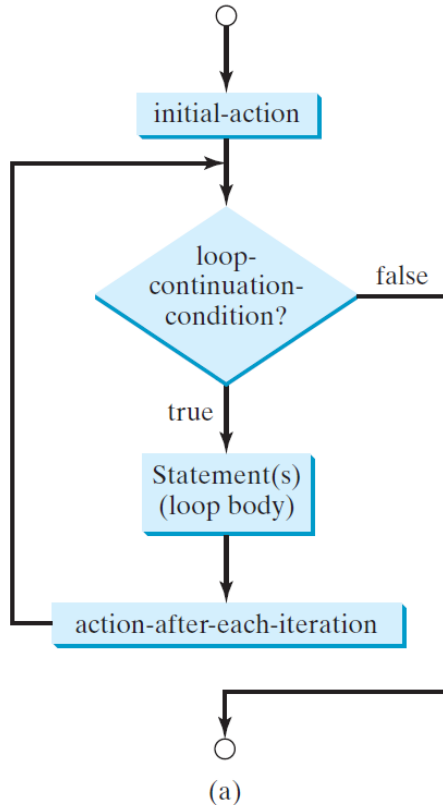
// gpa = gpa - 0.5;

number *= 2;

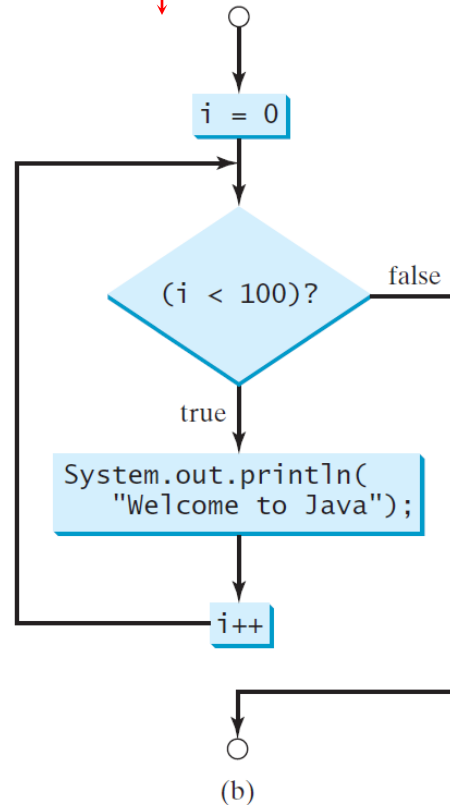
// number = number * 2;

for Loops

```
for (initial-action; loop-  
continuation-condition;  
action-after-each-iteration) {  
    // loop body;  
    Statement(s);  
}
```



```
int i;  
for (i = 0; i < 100; i++) {  
    System.out.println(  
        "Welcome to Java!");  
}
```



Trace for Loop

```
int i;
```

```
for (i = 0; i < 2; i++) {  
    System.out.println(  
        "Welcome to Java!");  
}
```

Declare i

Trace for Loop, cont.

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println(  
        "Welcome to Java!");  
}
```

Execute initializer
i is now 0

Trace for Loop, cont.

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println( "Welcome to Java!");  
}
```

(i < 2) is true
since i is 0

Trace for Loop, cont.

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```



Print Welcome to Java

System.out.println("Welcome to Java!");

Trace for Loop, cont.

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

Execute adjustment statement
i now is 1

Trace for Loop, cont.

```
int i;  
for (i = 0; i < 2; ++i) {  
    System.out.println("Welcome to Java!");  
}
```

(i < 2) is still true
since i is 1

Trace for Loop, cont.

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

Print Welcome to Java

Trace for Loop, cont.

Execute adjustment statement
i now is 2

```
int i;  
for (i = 0; i < 2; i++){  
    System.out.println("Welcome to Java!");  
}
```



Trace for Loop, cont.

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```

(i < 2) is false
since i is 2

Trace for Loop, cont.

```
int i;  
for (i = 0; i < 2; i++) {  
    System.out.println("Welcome to Java!");  
}
```



Exit the loop. Execute the next
statement after the loop

Repetition over a range

```
System.out.println("1 squared = " + 1 * 1);  
System.out.println("2 squared = " + 2 * 2);  
System.out.println("3 squared = " + 3 * 3);  
System.out.println("4 squared = " + 4 * 4);  
System.out.println("5 squared = " + 5 * 5);  
System.out.println("6 squared = " + 6 * 6);
```

– Intuition: "I want to print a line for each number from 1 to 6"

- The `for` loop does exactly that!

```
for (int i = 1; i <= 6; i++) {  
    System.out.println(i + " squared = " + (i * i));  
}
```

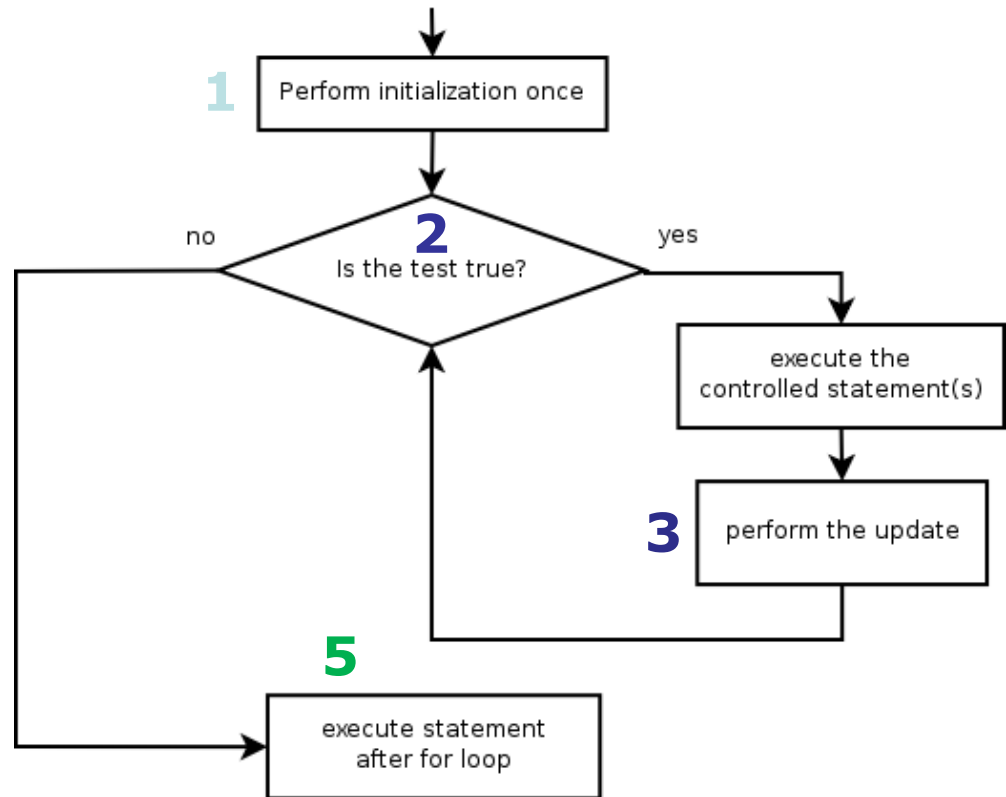
– "For each integer `i` from 1 through 6, print ..."

Loop walkthrough

```
for (int i 1 = 1; i 2 <= 4; i 3++) {  
    System.out.println(i + " squared = " + (i * i));  
}  
5 System.out.println("Whoo!");
```

Output:

```
1 squared = 1  
2 squared = 4  
3 squared = 9  
4 squared = 16  
Whoo!
```



Expressions for counter

```
int highTemp = 5;  
for (int i = -3; i <= highTemp / 2; i++) {  
    System.out.println(i * 1.8 + 32);  
}
```

– Output:

26.6
28.4
30.2
32.0
33.8
35.6

System.out.print

- Prints without moving to a new line
 - allows you to print partial messages on the same line

```
int highestTemp = 5;
for (int i = -3; i <= highestTemp / 2; i++) {
    System.out.print((i * 1.8 + 32) + "  ");
}
```

- Output:

26.6 28.4 30.2 32.0 33.8 35.6

- Concatenate " " to separate the numbers

Counting down

- The **update** can use `--` to make the loop count down.
 - The **test** must say `>` instead of `<`

```
System.out.print("T-minus ");
for (int i = 10; i >= 1; i--) {
    System.out.print(i + ", ");
}
System.out.println("blastoff!");
System.out.println("The end.");
```

– Output:

```
T-minus 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, blastoff!
The end.
```

Note

The initial-action in a for loop can be a list of zero or more comma-separated expressions. The action-after-each-iteration in a for loop can be a list of zero or more comma-separated statements. Therefore, the following two for loops are correct. They are rarely used in practice, however.

```
for (int i = 1; i < 100; System.out.println(i++));
```

```
for (int i = 0, j = 0; (i + j < 10); i++, j++) {
```

```
    // Do something
```

```
}
```


Note

If the loop-continuation-condition in a for loop is omitted, it is implicitly true. Thus the statement given below in (a), which is an infinite loop, is correct. Nevertheless, it is better to use the equivalent loop in (b) to avoid confusion:

```
for ( ; ; ) {  
    // Do something  
}
```

(a)

Equivalent

```
while (true) {  
    // Do something  
}
```


(b)

Caution

Adding a semicolon at the end of the for clause before the loop body is a common mistake, as shown below:

Logic
Error

```
for (int i=0; i<10; i++);  
{  
    System.out.println("i is " + i);  
}
```




Caution, cont.

Similarly, the following loop is also wrong:

```
int i=0;  
while (i < 10);  
{  
    System.out.println("i is " + i);  
    i++;  
}
```


Logic Error



In the case of the do loop, the following semicolon is needed to end the loop.

```
int i=0;  
do {  
    System.out.println("i is " + i);  
    i++;  
} while (i<10);
```

Correct



Which Loop to Use?

The three forms of loop statements, while, do-while, and for, are expressively equivalent; that is, you can write a loop in any of these three forms. For example, a while loop in (a) in the following figure can always be converted into the following for loop in (b):

```
while (loop-continuation-condition) {  
    // Loop body  
}
```

(a)

Equivalent

```
for ( ; loop-continuation-condition; )  
    // Loop body  
}
```

(b)

A for loop in (a) in the following figure can generally be converted into the following while loop in (b) except in certain special cases (see Review Question 3.19 for one of them):

```
for (initial-action;  
     loop-continuation-condition;  
     action-after-each-iteration) {  
    // Loop body;  
}
```

(a)

Equivalent

```
initial-action;  
while (loop-continuation-condition) {  
    // Loop body;  
    action-after-each-iteration;  
}
```

(b)

Recommendations

Use the one that is most intuitive and comfortable for you. In general, a for loop may be used if the number of repetitions is known, as, for example, when you need to print a message 100 times. A while loop may be used if the number of repetitions is not known, as in the case of reading the numbers until the input is 0. A do-while loop can be used to replace a while loop if the loop body has to be executed before testing the continuation condition.

4.4.1: Enter the for loop's output.

```
public class ForLoopOutput {  
    public static void main (String [] args) {  
        int i;  
        for (i = 0; i < 5; ++i) {  
            System.out.print(i);  
        }  
    }  
}
```

Output: 01234

4.4.1: Enter the for loop's output.

```
public class ForLoopOutput {  
    public static void main (String [] args) {  
        int i;  
        for (i = 0; i < 3; ++i) {  
            System.out.print(i * 2);  
        }  
    }  
}
```

Output: 024

4.4.1: Enter the for loop's output.

```
public class ForLoopOutput {  
    public static void main (String [] args) {  
        int i;  
        for (i = 0; i > -3; --i) {  
            System.out.print(i);  
        }  
    }  
}
```

Output: 0-1-2

4.5.1: For loops.

- Level 1:

Write a for loop that prints: 1 2 ... userNum

Ex: If the input is: 4

the output is: 1 2 3 4

4.5.1: For loops.

- Solution:

```
import java.util.Scanner;
```

```
public class ForLoops {  
    public static void main (String [] args) {  
        int userNum;  
        int i;  
  
        Scanner input = new Scanner(System.in);  
        userNum = input.nextInt();  
  
        for (i = 1; i <= userNum; ++i) {  
            System.out.print(i + " ");  
        }  
    }  
}
```

4.5.1: For loops.

- Level 2

Write code that prints: countNum ... 2 1

Print a newline after each number.

Ex: If the input is: 3

the output is:

3

2

1

4.5.1: For loops.

- Solution:

```
import java.util.Scanner;
```

```
public class ForLoops {  
    public static void main (String [] args) {  
        int countNum;  
        int i;  
  
        Scanner input = new Scanner(System.in);  
        countNum = input.nextInt();  
  
        for (i = countNum; i > 0; --i) {  
            System.out.println(i);  
        }  
    }  
}
```

4.5.1: For loops.

- Level 3

Write code that prints: Ready! firstNumber ... 2 1 Run!

Your code should contain a for loop. Print a newline after each number and after each line of text.

Ex: If the input is: 3

the output is:

Ready!

3

2

1

Run!

4.5.1: For loops.

Solution:

```
import java.util.Scanner;
public class ForLoops {
    public static void main (String [] args) {
        int firstNumber;
        int i;

        Scanner input = new Scanner(System.in);
        firstNumber = input.nextInt();

        System.out.println("Ready!");
        for (i = firstNumber; i > 0; --i) {
        System.out.println(i);
        }
        System.out.println("Run!");

    }
}
```

4.5.1: For loops.

Level 4:

Write a for loop that prints countNum ... -1 0.

- Ex: If the input is: -3

the output is: -3 -2 -1 0

4.5.1: For loops.

Solution:

```
import java.util.Scanner;

public class ForLoops {
    public static void main (String [] args) {
        int countNum;
        int i;

        Scanner input = new Scanner(System.in);
        countNum = input.nextInt();

        for (i = countNum; i <= 0; ++i) {
            System.out.print(i + " ");
        }
    }
}
```


4.5.1: For loops.

Write a for loop that prints from startNumber to finalNumber.

Ex: If the input is: -3 1

the output is: -3 -2 -1 0 1

4.5.1: For loops.

- Solution:

```
import java.util.Scanner;
```

```
public class ForLoops {  
    public static void main (String [] args) {  
        int startNumber;  
        int finalNumber;  
        int i;  
  
        Scanner input = new Scanner(System.in);  
        startNumber = input.nextInt();  
        finalNumber = input.nextInt();  
  
        for (i = startNumber; i <= finalNumber; ++i) {  
            System.out.print(i + " ");  
        }  
    }  
}
```