

CS1150

Principles of Computer Science

Loops (Part II)

Yanyan Zhuang

Department of Computer Science

<http://www.cs.uccs.edu/~yzhuang>

Review

- Is this an infinite loop? Why (not)?

```
int count = 1;
while (count <= 5){
    System.out.println("The value of count is " + count);
    count++;
}
```



Review

- Is this an infinite loop? Why (not)?

```
int count = 1;
while (count <= 5); {
    System.out.println("The value of count is " + count);
    count++;
}
```

```
int count = 1;
do {
    count++;
    System.out.println("Count = " + count);
} while (count <= 5);
```



Overview

- While loop
- Do...while loop
- **For loop**



For loops

- For loops are used to execute a loop a **preset** number of times

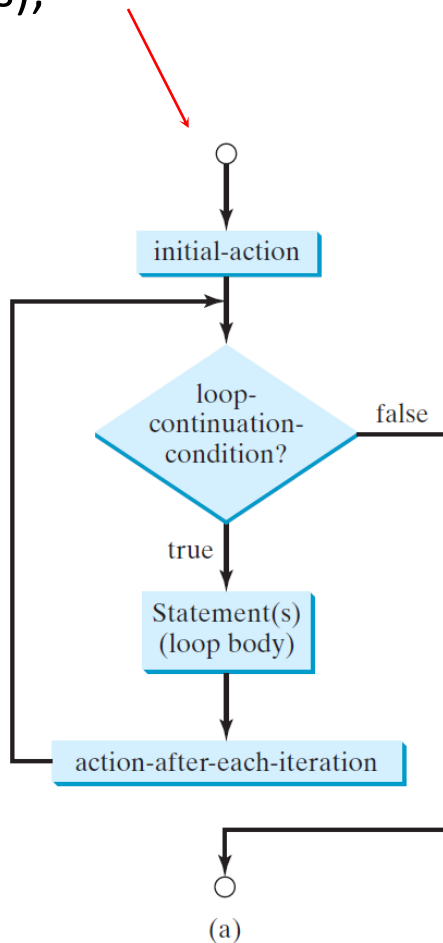
```
for (1initial-action; 2loop-condition; 4action-after-each-iteration) {  
    3Statement(s);  
}
```

1. The control variable is initialized to a start value
2. The control variable is compared to the loop condition
3. If the condition is true, then loop body is executed
4. The control variable is updated after loop body completes
5. Steps 2-4 are repeated as long as the loop condition is true

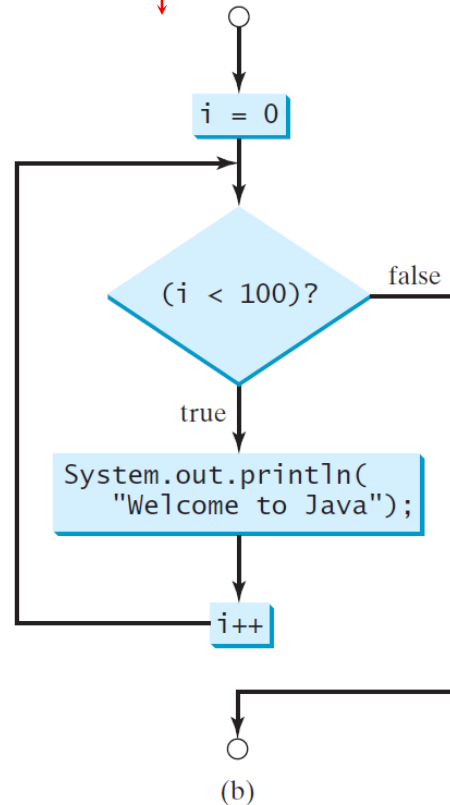


for Loops

```
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

Declare i

```
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!");  
}
```

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



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.

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

Execute adjustment statement
i now is 2

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

Rules of for loops

- The control structure of for-loop must be in parentheses
 - `for (i=0; i<= 2; i++) { statements; }`
 - The loop condition (`i <= 2`) must be a boolean expression
 - The control variable (`i`): not recommended to be changed within the for-loop body
 - Curly braces not necessary if only one statement in loop
 - Best practice is to always include curly braces
- Example: `ForPrintNTimes.java` (variable scope)



Note

- You may declare the control variable outside/within the for-loop

```
for (int j = 0; j <= 5; j++) {  
    System.out.println ("For loop iteration = " + j);  
}
```

```
int j;
```

```
for (j = 0; j <= 5; j++) {  
    System.out.println ("For loop iteration = " + j);  
}
```

- Note on variable scope (the area a variable can be referenced)
 - Declaring control variable **before the for loop** cause its **scope** to be both inside and outside for-loop
 - Declaring the control variable **in the for-loop** causes its **scope** to be only inside the for loop
 - ▶ First example above: If I tried to use the variable j outside the for-loop - error



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 is correct, but rarely used in practice.

```
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:

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

Logic Error



Caution, cont.

Similarly, the following loop is also wrong:

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

In the case of the do...while 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):

```
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.

- 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.

Fahrenheit2CelsiusForLoop.java

RandomNumbers example

- **Math.random()**
 - Returns a double value, **greater than or equal** to 0.0 and **less than** 1.0, i.e., **[0.0, 1.0)**
 - ▶ `double randomDouble;`
 - ▶ `randomDouble = Math.random(); // [0.0, 1.0)`
 - How to generate a random double in **[0.0, 10.0)**?
 - ▶ `randomDouble = Math.random() * 10;`
 - How to generate a random **integer** in **[0, 10)**?
 - ▶ `int randomInt;`
 - ▶ `randomInt = (int) (Math.random() * 10);`
 - ▶ Note: DO NOT use `(int)Math.random()*10`



RandomNumbers example

- `Math.random()`
 - How to generate a random integer between [lower, upper)?
 - ▶ Example: `int lower=100, upper=120;`
 - ▶ `randomDouble = Math.random(); // [0.0, 1.0)`
 - ▶ `randomDouble = randomDouble * (upper-lower); // expand: [0.0, 20.0)`
 - ▶ `randomDouble = lower + randomDouble; // shift: [100.0, 120.0)`
 - ▶ `randomInt = (int) randomDouble; // cast: double → int`
 - Or in one step
 - ▶ `randomInt = (int) (lower + Math.random() * (upper-lower));`
 - ▶ Let's look at `RandomNumbers.java` and `AdditionQuiz.java`



Nested Loops

- If statements can be nested

```
If (boolean expression) {  
    if(boolean expression) {  
        statement(s);  
    } // end of inner if  
} // end of outer if
```

- Loops can be nested too



Nested Loops

- While loop within a while loop

```
int outer = 1;
while (outer < 3) {
    System.out.println ("The outer loop iteration is = " + outer);
    //inner loop: It will do all iterations before outer loop does another iteration
    int inner = 1;
    while (inner <= 5) {
        System.out.println (" The inner loop iteration is = " + inner);
        inner++;
    }
    outer++;
}
```

NestedWhile.java



Nested Loops

- For loop within a for loop

```
for (int outer = 1; outer < 3; outer++) {
```

```
    System.out.println ("The outer loop iteration is = " + outer);
```

```
    // This is the inner for loop
```

```
    // It will do all iterations before outer loop does another iteration
```

```
    for (int inner = 1; inner <= 5; inner++) {
```

```
        System.out.println (" The inner loop iteration is = " + inner);
```

```
    }
```

```
}
```

NestedFor.java



Nested Loops

Problem: Write a program that uses nested for loops to print a multiplication table.

MultiplicationTable.java



Using `break` and `continue`

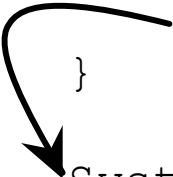
- Break in loops
 - Used "break" in switch statements to end a case
 - Can be used in a loop to **terminate** a loop
 - Breaks the entire loop
- Continue in loops
 - Used to end **current iteration** of loop
 - Program control goes to end of loop body



break

```
public class TestBreak {
    public static void main(String[] args) {
        int sum = 0;
        int number = 0;

        while (number < 20) {
            number++;
            sum += number;
            if (sum >= 100)
                break;
        }
        System.out.println("The number is " + number);
        System.out.println("The sum is " + sum);
    }
}
```




continue

```
public class TestContinue {
    public static void main(String[] args) {
        int sum = 0;
        int number = 0;

        while (number < 20) {
            number++;
            if (number == 10 || number == 11)
                continue;
            sum += number;
        }

        System.out.println("The sum is " + sum);
    }
}
```



Guessing Number Problem Revisited

Let's look at AdditionQuiz (random number + break example)

Using `break` and `continue`

- There are cases when the use of a `break` or `continue` in a loop is necessary
- Overusing or improper use can make program hard to read, maintain, and debug.
- Best practice is to never use these unless you have a specific situation



Practice! Predicting tuition

- The tuition for a university is currently \$10,000. It will increase by 7% each year
- In how many years will the tuition be doubled?
 - Which loop to use?
 - Statement(s) to be repeated?
 - Condition to check?



Summary

- While loop
- Do...while loop
- For loop
- Break and continue

