For-loops

Mitsu Ogihara

Department of Computer Science
University of Miami
# Table of Contents

1. For Loops as the Method for Iteration
2. Loops Inside Loops
3. Constants
What is a for-loop?

- A loop · · · a program structure where a block of the code is executed repeatedly
- The repetition continues as long as a specific condition holds
Three Types of Loops in Java

There are three types of loops in Java: for, while, and do-while.

**For-loops**: a for-loop is a structure that encompasses its loop body with three components:

- *Initialization* (a single statement)
- *Continuation condition* (a condition generating statement)
- *Update* (a single statement)
General For-loop Format

The general format for a for-loop is:

```plaintext
for (INITIALIZATION; CONTINUATION CONDITION; UPDATE) {
    loop-body
}
```
The general format for a for-loop is:

```plaintext
for (INITIALIZATION; CONTINUATION CONDITION; UPDATE) {
  loop-body
}
```

This is interpreted as:
- Execute INITIALIZATION
- As long as the CONTINUATION CONDITION holds
  - Execute the loop-body
  - Execute UPDATE
Flowchart for a For-loop

1. **Initialization**
2. **Continuation Condition**
   - **true**:
     - **Loop-Body**
     - **Update**
   - **false**:
     - **Terminate**
An Example

```java
public class ForExample {
    public static void main( String[] args ) {
        int count;
        for ( count = 1; count <= 8; count ++ ) {
            System.out.println( "The value of count is " + count );
        }
    }
}
```

INITIALIZATION: set count to 1
CONTINUATION: as long as count is less than or equal to 8
UPDATE: add 1 to count
An Example

```java
public class ForExample {
    public static void main( String[] args ) {
        int count;
        for ( count = 1; count <= 8; count ++ ) {
            System.out.println( "The value of count is " + count );
        }
    }
}
```

Loop body: print the String literal "The value of count is " followed by the value of count
Flowchart

```
count = 1

count <= 8?

System.out.println("The value of count is " + count);

count ++

END OF LOOP EXECUTION

NO

YES

END OF LOOP
```
Inserting Type Declaration into the Loop Definition

You may add the variable type to the iterator variable initialization in the for loop, for example:

```java
for (int count = 1; count <= 8; count ++) {
    System.out.println("The value of count is "+ count);
}
```
You may add the variable type to the iterator variable initialization in the for loop, for example:

```java
for (int count = 1; count <= 8; count ++) {
    System.out.println("The value of count is " + count);
}
```

Note that the scope of `count` is the entire for loop.
Loop Variants

for (count = 10; count >= 0; count --)
count goes 10, 9, 8, ..., 0

for (count = 10; count >= 0; count -= 2)
count goes 10, 8, 6, ..., 0

for (count = 0; count < 10; count += 2)
count goes 0, 2, 4, 6, 8
The for-loop iterates the variable `value` from the value 100 downward, decreasing the value by 1 each time; quits as soon as the value becomes less than 1.
Counting Down

```java
public class CountDown {
    public static void main( String[] args ) {
        for ( int value = 100; value >= 1; value -- ) {
            System.out.println( ".." + value );
        }
        System.out.println( "BOOOOOOOOOOOOOOOOOOOOOOOOM!" );
    }
}
```

The loop body simply prints out the value of the variable with two dots preceding
For Loops as the Method for Iteration

Loops Inside Loops

Constants

Counting Down

```java
public class CountDown {
    public static void main( String[] args ) {
        for ( int value = 100; value >= 1; value -- ) {
            System.out.println( ".." + value );
        }
        System.out.println( "BOOOOOOOOOOOOOOOOOOOOOOOOM!" );
    }
}
```

Upon quitting the loop, print a message
Accelerated Counting Down

```java
public class CountDownFast {
    public static void main( String[] args ) {
        for ( int value = 1000000; value >= 1; value /= 2 ) {
            System.out.println( ".." + value );
        }
        System.out.println( "BOOOOOOOOOOOOOOOOOOOOOOOOM!" );
    }
}
```

The for-loop iterates the variable `value` from one million, dividing it by two each time; the loop quits as soon as `value` becomes less than 1.
Accelerated Counting Down

```java
public class CountDownFast {
    public static void main( String[] args ) {
        for ( int value = 1000000; value >= 1; value /= 2 ) {
            System.out.println( ".." + value );
        }
        System.out.println( "BOOOOOOOOOOOOOOOOOOOOOOOOM!" );
    }
}
```

The loop body simply prints out the value of the variable with two dots preceding
Accelerated Counting Down

```java
public class CountDownFast {
    public static void main( String[] args ) {
        for ( int value = 1000000; value >= 1; value /= 2 ) {
            System.out.println( ".." + value );
        }
        System.out.println( "BOOOOOOOOOOOOOOOOOOOOOOOOM!" );
    }
}
```

Upon quitting the loop, print a message
Computing Squares

Compute the squares $i^2$ for $i = 1, \ldots, 20$
Computing Squares

Compute the squares $i^2$ for $i = 1, \ldots, 20$

```java
public class Squares {
    public static void main( String[] args ) {
        int square;
        for ( int i = 1; i <= 20; i ++ ) {
            square = i * i;
            System.out.print( "The square of " + i );
            System.out.println( " is " + square );
        }
    }
}
```

For loop for generating 1, \ldots, 20
Compute the squares $i^2$ for $i = 1, \ldots, 20$

```java
public class Squares {
    public static void main( String[] args ) {
        int square;
        for ( int i = 1; i <= 20; i ++ ) {
            square = i * i;
            System.out.print( "The square of " + i );
            System.out.println( " is " + square );
        }
    }
}
```

Compute the square in the variable `square`
Computing Squares, Alternative

Compute the squares $i^2$ for $i = 1, \ldots, 20$
Use the fact $i^2$ is equal to the sum of the first $i$ odd numbers
Compute the squares $i^2$ for $i = 1, \ldots, 20$
Use the fact $i^2$ is equal to the sum of the first $i$ odd numbers

```java
public class SquaresAlt {
    public static void main( String[] args ) {
        int square = 1;
        for ( int i = 1; i <= 20; i ++ ) {
            square += 2 * i + 1;
            System.out.print( "The square of " + i );
            System.out.println( " is " + square );
        }
    }
}
```

The initial value of `square` is 0
Computing Squares, Alternative

Compute the squares $i^2$ for $i = 1, \ldots, 20$
Use the fact $i^2$ is equal to the sum of the first $i$ odd numbers

```java
public class SquaresAlt {
    public static void main( String[] args ) {
        int square = 1;
        for ( int i = 1; i <= 20; i ++ ) {
            square += 2 * i + 1;
            System.out.print( "The square of " + i );
            System.out.println( " is " + square );
        }
    }
}
```

For loop for generating 1, \ldots, 20
Computing Squares, Alternative

Compute the squares $i^2$ for $i = 1, \ldots, 20$
Use the fact $i^2$ is equal to the sum of the first $i$ odd numbers

```java
public class SquaresAlt {
    public static void main( String[] args ) {
        int square = 1;
        for ( int i = 1; i <= 20; i ++ ) {
            square += 2 * i + 1;
            System.out.print( "The square of " + i);
            System.out.println( " is " + square );
        }
    }
}
```

Compute the square by adding $2 \times i + 1$
Summing from 1 to 100 Every Third Number

Compute the sum of numbers 1, 4, 7, ..., 100
Summing from 1 to 100 Every Third Number

Compute the sum of numbers 1, 4, 7, ..., 100

```java
public class SumTo100EveryThird {
    public static void main( String[] args ) {
        //--- initialize the sum
        long sum = 0;
        //--- iterate the value of j from 1 to 100
        for ( int count = 1; count <= 100; count += 3 ) {
            sum += count;
        }
        System.out.println( "1 + 4 + ... + 100 = " + sum );
    }
}
```

Set the total value \( \text{sum} \) to 0
Summing from 1 to 100 Every Third Number

Compute the sum of numbers 1, 4, 7, ..., 100

```java
// calculate the sum 1 + 4 + ... + 100
public class SumTo100EveryThird {
    public static void main( String[] args ) {
        //-- initialize the sum
        long sum = 0;
        //-- iterate the value of j from 1 to 100
        for ( int count = 1; count <= 100; count += 3 ) {
            sum += count;
        }
        System.out.println( "1 + 4 + ... + 100 = " + sum );
    }
}
```

Iterate the value of `count` from 1 through 100, adding 3 each time
Summing from 1 to 100 Every Third Number

Compute the sum of numbers 1, 4, 7, ..., 100

```java
// calculate the sum 1 + 4 + ... + 100
public class SumTo100EveryThird {
    public static void main( String[] args ) {
        //-- initialize the sum
        long sum = 0;
        //-- iterate the value of j from 1 to 100
        for ( int count = 1; count <= 100; count += 3 ) {
            sum += count;
        }
        System.out.println( "1 + 4 + ... + 100 = " + sum );
    }
}
```

Add the value of `count` to `sum`
Summing from 1 to 100 Every Third Number

Compute the sum of numbers 1, 4, 7, ..., 100

```java
// calculate the sum 1 + 4 + ... + 100
public class SumTo100EveryThird {
    public static void main(String[] args) {
        // initialize the sum
        long sum = 0;
        // iterate the value of j from 1 to 100
        for (int count = 1; count <= 100; count += 3) {
            sum += count;
        }
        System.out.println("1 + 4 + ... + 100 = " + sum);
    }
}
```

Print the outcome
The String literal "1 + 4 + ... + 100 = " followed by the total; the two parts are concatenated inside the println statement
Table of Contents

1. For Loops as the Method for Iteration
2. Loops Inside Loops
3. Constants
Loops May Exist in a Loop Body

A for-loop may appear in the loop body of a for-loop
A for-loop may appear in the loop body of a for-loop

```cpp
int count1, count2;
for (count1 = X; ...; ...) {
    ...
    for (count2 = Y; ... ; ...) {
        ...
    }
    ...
}
```
Loops May Exist in a Loop Body

A for-loop may appear in the loop body of a for-loop

```c
int count1, count2;
for (count1 = X; ... ; ...) {
    ...
    for (count2 = Y; ... ; ...) {
        ...
    }
    ...
}
```

Naturally we want `count1` and `count2` to be different.

The use of `count1` in place of `count2` may result in a serious logic error.
Box

Receive height and width from user and print a box of the dimensions

```java
public class DarkBox {
    //-- main method
    public static void main( String[] args ) {
        int height, width;
        // set height and width
        height = 10;
        width = 50;
        // exterior loop
        for ( int posV = 1; posV <= height; posV ++ ) {
            // interior loop
            for ( int posH = 1; posH <= width; posH ++ ) {
                System.out.print( "#" );
            }
            System.out.println();
        }
    }
}
```

Variable declarations and value assignments
Receive height and width from user and print a box of the dimensions

```java
public class DarkBox {
    //-- main method
    public static void main( String[] args ) {
        int height, width;
        // set height and width
        height = 10;
        width = 50;
        // exterior loop
        for ( int posV = 1; posV <= height; posV ++ ) {
            // interior loop
            for ( int posH = 1; posH <= width; posH ++ ) {
                System.out.print( "#" );
            }
            System.out.println();
        }
    }
}
```

The external loop iterates \texttt{posV} from 1 to \texttt{height}
Box

Receive height and width from user and print a box of the dimensions

```java
public class DarkBox {
    //-- main method
    public static void main( String[] args ) {
        int height, width;
        // set height and width
        height = 10;
        width = 50;
        // exterior loop
        for ( int posV = 1; posV <= height; posV ++ ) {
            // interior loop
            for ( int posH = 1; posH <= width; posH ++ ) {
                System.out.print( "#" );
            }
            System.out.println();
        }
    }
}
```

The internal loop iterates posH from 1 to width
Receive height and width from user and print a box of the dimensions

```java
public class DarkBox {
    //-- main method
    public static void main( String[] args ) {
        int height, width;
        // set height and width
        height = 10;
        width = 50;
        // exterior loop
        for ( int posV = 1; posV <= height; posV ++ ) {
            // interior loop
            for ( int posH = 1; posH <= width; posH ++ ) {
                System.out.print( "#" );
            }
            System.out.println();
        }
    }
}
```

Inside the internal loop print one # for each value of posH
This generates width many #'s successively
Receive height and width from user and print a box of the dimensions

```java
public class DarkBox {
    //-- main method
    public static void main( String[] args ) {
        int height, width;
        // set height and width
        height = 10;
        width = 50;
        // exterior loop
        for ( int posV = 1; posV <= height; posV ++ ) {
            // interior loop
            for ( int posH = 1; posH <= width; posH ++ ) {
                System.out.print( "#" );
            }
            System.out.println();
        }
    }
}
```

After concluding the inner loop go to the new line
Framed Box

Similar to the previous one, but surround the box with a line

- Attach a ']' before and after each line
- Attach "+———-+" at the top and the bottom, where the number of '-' is equal to the width.
Example

With dimension = 10*10

```java
% java FramedBox
+----------+
|##########|
|##########|
|##########|
|##########|
|##########|
|##########|
|##########|
|##########|
|##########|
|##########|
+----------+
```
Previously we used the following double for-loop

```java
for ( int posV = 1; posV <= height; posV ++ ) {
    for ( int posH = 1; posH <= width; posH ++ ) {
        System.out.print( "#" );
    }
    System.out.println();
}
```

We change this to

```java
for ( int posV = 1; posV <= height; posV ++ ) {
    System.out.print( "|" );
    for ( int posH = 1; posH <= width; posH ++ ) {
        System.out.print( "#" );
    }
    System.out.println( "|" );
}
```

This adds a `|` before and after each line of `#`'s
We also add the following before and after:

```java
System.out.print( "+" );
for ( int posH = 1; posH <= width; posH ++ ) {
    System.out.print( "-" );
}
System.out.println( "+" );
```

This produces a +, width many −’s, and a + and goes to the next line
Framed Box

Similar to the previous one, but surround the box with a line
Framed Box

Similar to the previous one, but surround the box with a line

```java
public class FramedBox {
    public static void main( String[] args ) {
        int height = 10, width = 50;
        //--- the top line ---/
        System.out.print( "+" );
        for ( int posH = 1; posH <= width; posH ++ ) {
            System.out.print( "-" );
        }
        System.out.println( "+" );
        //--- the middle lines ---/
        for ( int posV = 1; posV <= height; posV ++ ) {
            System.out.print( "|" );
            for ( int posH = 1; posH <= width; posH ++ ) {
                System.out.print( "#" );
            }
            System.out.println( "|" );
        }
        //--- the bottom line---/
        System.out.print( "+" );
        for ( int posH = 1; posH <= width; posH ++ ) {
            System.out.print( "-" );
        }
        System.out.println( "+" );
    }
}
```

Declaring the dimension variables and assigning values to them
Framed Box

Similar to the previous one, but surround the box with a line

```java
public class FramedBox {
    public static void main( String[] args ) {
        int height = 10, width = 50;
        //--- the top line ---//
        System.out.print( "+" );
        for ( int posH = 1; posH <= width; posH ++ ) {
            System.out.print( "-" );
        }
        System.out.println( "+" );

        //--- the middle lines ---//
        for ( int posV = 1; posV <= height; posV ++ ) {
            System.out.print( "|" );
            for ( int posH = 1; posH <= width; posH ++ ) {
                System.out.print( "#" );
            }
            System.out.println( "|" );
        }

        //--- the bottom line---//
        System.out.print( "+" );
        for ( int posH = 1; posH <= width; posH ++ ) {
            System.out.print( "-" );
        }
        System.out.println( "+" );
    }
}
```

The top part: a ’+’, dashes, and then a ’+’
Framed Box

Similar to the previous one, but surround the box with a line

```java
public class FramedBox {
    public static void main( String[] args ) {
        int height = 10, width = 50;
        //--- the top line ---//
        System.out.print( "+" );
        for ( int posH = 1; posH <= width; posH ++ ) {
            System.out.print( "-" );
        }
        System.out.println( "+" );
        //--- the middle lines ---//
        for ( int posV = 1; posV <= height; posV ++ ) {
            System.out.print( "|" );
            for ( int posH = 1; posH <= width; posH ++ ) {
                System.out.print( "#" );
            }
            System.out.println( "|" );
        }
        //--- the bottom line---//
        System.out.print( "+" );
        for ( int posH = 1; posH <= width; posH ++ ) {
            System.out.print( "-" );
        }
        System.out.println( "+" );
    }
}
```

The middle part
Framed Box

Similar to the previous one, but surround the box with a line

```java
public class FramedBox {
    public static void main( String[] args ) {
        int height = 10, width = 50;
        //--- the top line ---//
        System.out.print( "+" );
        for ( int posH = 1; posH <= width; posH ++ ) {
            System.out.print( "-" );
        }
        System.out.println( "+" );
        //--- the middle lines ---//
        for ( int posV = 1; posV <= height; posV ++ ) {
            System.out.print( "|" );
            for ( int posH = 1; posH <= width; posH ++ ) {
                System.out.print( "#" );
            }
            System.out.println( "|");
        }
        //--- the bottom line---//
        System.out.print( "+" );
        for ( int posH = 1; posH <= width; posH ++ ) {
            System.out.print( "-" );
        }
        System.out.println( "+" );
    }
}
```

The bottom part, the same as the top part
Inverted Triangle

Print the right-angle triangle of a fixed height

- The triangular area will be represented with a *
- The first line has the height-many ’s
- The second line the \((height - 1)\) many ’s
- ...
- The last line has one *

All the lines will print left-flushed
Example

```java
% java TriangleFlipped
 ******************
******************
******************
***************
**************
*************
************
***********
**********
*********
********
*******
******
*****
****
***
**
*
```
Print an inverted triangle

```java
//-- print a triangle
public class TriangleFlipped {
    public static void main( String[] args ) {
        int height = 18;
        for ( int posV = height; posV >= 1; posV -- ) {
            for ( int posH = 1; posH <= posV; posH ++ ) {
                System.out.print( "*" );
            }
            System.out.println();
        }
    }
}
```

Height specification
Flipped Triangle

Print an inverted triangle

```java
//-- print a triangle
public class TriangleFlipped {
    public static void main( String[] args ) {
        int height = 18;
        for ( int posV = height; posV >= 1; posV -- ) {
            for ( int posH = 1; posH <= posV; posH ++ ) {
                System.out.print( "*" );
            }
            System.out.println();
        }
    }
}
```

External loop is executed with the variable `posV` iterated from `height` down to 1
Print an inverted triangle

```java
/** print a triangle */
public class TriangleFlipped {
    public static void main( String[] args ) {
        int height = 18;
        for ( int posV = height; posV >= 1; posV -- ) {
            for ( int posH = 1; posH <= posV; posH ++ ) {
                System.out.print( "*" );
            }
            System.out.println();
        }
    }
}
```

The internal loop produces `posV` many `*`’s
The `println` is for moving to the next line
Consider the problem of printing in a single line '1' once, '2' twice, . . . , '9' nine times.

The code can be:

```java
for ( int number = 1; number <= 9; number ++ ) {
    for ( int count = 1; count <= number; count ++ ) {
        System.out.print( number );
    }
}
System.out.println();
```

What if we want to generate 9 lines, where the $i$-th line is truncated after output $i$ as in:

```
1
12
123
1234
12345
123456
1234567
12345678
123456789
```
Solution

```java
public class NumberSequences {
    public static void main( String[] args ) {
        for ( int line = 1; line <= 9; line ++ ) {
            for ( int number = 1; number <= line; number ++ ) {
                for ( int count = 1; count <= number; count ++ ) {
                    System.out.print( number );
                }
            }
            System.out.println();
        }
    }
}
```

The line number coincides with the maximum number to print
Solution

```java
public class NumberSequences {
    public static void main( String[] args ) {
        for ( int line = 1; line <= 9; line ++ ) {
            for ( int number = 1; number <= line; number ++ ) {
                for ( int count = 1; count <= number; count ++ ) {
                    System.out.print( number );
                }
            }
            System.out.println();
        }
    }
}
```

Individual line; uses the previous double loop with `line` substituting 9
# Table of Contents

1. For Loops as the Method for Iteration
2. Loops Inside Loops
3. Constants
A constant in a program is a value with a name that is used throughout the program with no possibility of changing value during the execution of the program.

To define a constant use

```java
public static final <type> <name> = <value>;
```

at the same level as methods, where `final` means that the variable can be assigned a value only once.

Usually they appear immediately after class declaration.

Also, we use all capitals for constants.
Framed Box with Size Constants

Use four parameters:

- **SCREEN_WIDTH** and **SCREEN_HEIGHT**: hypothetical width and height of the screen area
- **INTERIOR_WIDTH** and **INTERIOR_HEIGHT**: the previous values minus 2 - subtract character space for the frame

```java
public class FramedBoxConstant {
    //---- dimensional information
    public static final int SCREEN_WIDTH = 72;
    public static final int SCREEN_HEIGHT = 20;
    public static final int INTERIOR_WIDTH = SCREEN_WIDTH - 2;
    public static final int INTERIOR_HEIGHT = SCREEN_HEIGHT - 2;
}
```

The constants have all-capital names
The second pair are defined based upon the first two
The Remainder of the Code

```java
//---- the top and the bottom
public static void topOrBottom() {
    System.out.print( "+" );
    for ( int posH = 1; posH <= INTERIOR_WIDTH; posH ++ ) {
        System.out.print( "-" );
    }
    System.out.println( "+" );
}

//---- the middle part
public static void middle() {
    for ( int posV = 1; posV <= INTERIOR_HEIGHT; posV ++ ) {
        System.out.print( "|" );
        for ( int posH = 1; posH <= INTERIOR_WIDTH; posH ++ ) {
            System.out.print( "#" );
        }
        System.out.println( "|" );
    }
}

//---- main
public static void main( String[] args ) {
    topOrBottom();
    middle();
    topOrBottom();
}
```

Method for the top line and for the bottom line
The Remainder of the Code

```java
//---- the top and the bottom
public static void topOrBottom() {
    System.out.print( "+" );
    for ( int posH = 1; posH <= INTERIOR_WIDTH; posH ++ ) {
        System.out.print( "-" );
    }
    System.out.println( "+" );
}
//---- the middle part
public static void middle() {
    for ( int posV = 1; posV <= INTERIOR_HEIGHT; posV ++ ) {
        System.out.print( "|" );
        for ( int posH = 1; posH <= INTERIOR_WIDTH; posH ++ ) {
            System.out.print( "#" );
        }
        System.out.println( "|" );
    }
}
//---- main
public static void main( String[] args ) {
    topOrBottom();
    middle();
    topOrBottom();
}
```

Method for the middle part
The Remainder of the Code

```java
//---- the top and the bottom
public static void topOrBottom() {
    System.out.print( "+" );
    for ( int posH = 1; posH <= INTERIOR_WIDTH; posH ++ ) {
        System.out.print( "-" );
    }
    System.out.println( "+" );
}

//---- the middle part
public static void middle() {
    for ( int posV = 1; posV <= INTERIOR_HEIGHT; posV ++ ) {
        System.out.print( "|" );
        for ( int posH = 1; posH <= INTERIOR_WIDTH; posH ++ ) {
            System.out.print( "#" );
        }
        System.out.println( "|" );
    }
}

//---- main
public static void main( String[] args ) {
    topOrBottom();
    middle();
    topOrBottom();
}
```

Main uses the two methods to generate the shape
The End