

Loops



Overview

- Like humans, programs should be able to repeat some actions while a condition is true
- In this module we will learn how to express repetitions in a program
- The program will repeat the same action while a condition is true and stops when it is false
- Example:
 - while *hungry is true* eat; when *hungry is false* stop eating

Learning Objectives

- To be able to write a while loop
- To be able to write a for loop
- To be familiar with the three parts of a loop
- To be able to trace a loop
- To be comfortable with nested loops
- To use a loop to solve problems involving String manipulation

Iteration

- Repetition of a program block while a condition is true
- Iterations allow us to control the flow of a program
- Two options:
 - `while` loop
 - `for` loop

while loop

- Executes the body of the loop as long as (or while) a Boolean expression is **true**

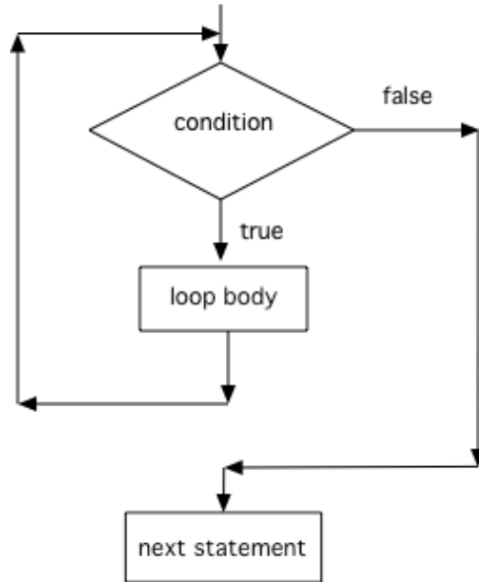


Figure 3: Control Flow in a while Loop

```
// while statements are repeated while the condition is true  
while (condition)  
{  
    statements;  
}
```

The simplest while loop

```
while (true) {  
    // start of the loop  
    statements;  
    statements;  
    statements;  
    statements;  
    // end of the loop  
}  
// code here won't get run!
```

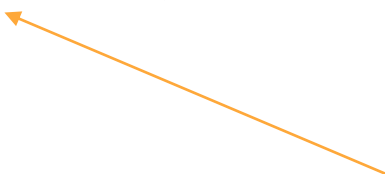
Basic Recipe for Animation with PennDraw

- Setup
 - `PennDraw.setCanvasSize`, `PennDraw.enableAnimation`, variable declarations
- `while(true)` loop
 - Clear the screen, then draw the next frame
 - Update the state
 - `PennDraw.advance()`

Check out **BubbleOut.java** for an example!

Basic Recipe for Animation with PennDraw

- Setup
 - `PennDraw.setCanvasSize`, `PennDraw.enableAnimation`, variable declarations
- `while(true)` loop
 - Clear the screen, then draw the next frame
 - Update the state
 - `PennDraw.advance()`



Tells PennDraw how many times per second to redraw the screen.

Check out **BubbleOut.java** for an example!

Basic Recipe for Animation with PennDraw

- Setup
 - `PennDraw.setCanvasSize`, `PennDraw.enableAnimation`, variable declarations
- `while(true)` loop
 - Clear the screen, then draw the next frame
 - Update the state
 - `PennDraw.advance()`

Tells PennDraw to draw all shapes and marks you've asked for since the last time you called this

Check out **BubbleOut.java** for an example!

Loop control variable

- The loop condition involves a **loop control variable**
- The **loop control variable** controls when the loop stops
- The loop condition tests that the value of the loop control variable matches a specific condition (>, <, >=, <=, ==, !=)

Three steps of a while loop

1. Initialize the loop variable (before the while loop)
2. Test the loop variable (in the loop header)
3. Change the loop variable (in the while loop body at the end)

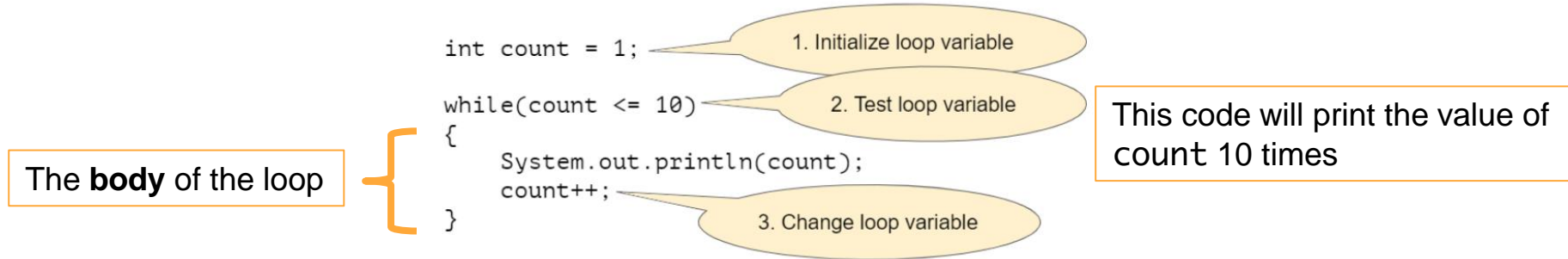
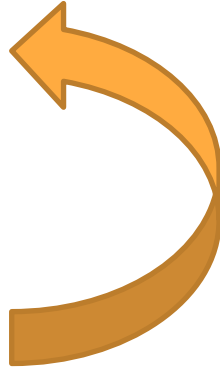


Figure 4: Three Steps of Writing a Loop

Tracing a while loop

- Evaluate a Boolean expression
- If **true**:
 - Execute the body of the loop
 - Repeat
- If **false**, exit the loop



Tracing a while loop

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

count	Count <= 5	Output
1	true	1
2	true	2
3	true	3
4	true	4
5	true	5
6	false	

Infinite loop

- A loop that never stops
- Happens when the condition is always true:
 - We forget to change the loop control variable
 - We do not change the loop control variable correctly

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

```
int count = 1;
while (count <= 5) {
    System.out.println(count);
    count--;
}
```

Off-by-one error

- The loop run one too many time
- The loop run one too few times
- Happens when:
 - We use the incorrect relational operator in the test loop step

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

The loop will execute 4 times instead of 5. Because we used < instead of <=

Aside: Scope

- Scope is the part of the program where a variable exists
 - A variable's scope is within the pair of curly braces { } it is defined in
 - Variables declared in a loop or if/else if/else only exist inside that structure.
 - A variable declared in a loop is “reset” on each iteration of the loop
 - Variables declared in separate scopes can have the same name.

Scope Example

- A variable declared in a loop is “reset” on each iteration of the loop

```
int x = 0;
while (x < 10) {
    int y = 0;
    System.out.println(y); // always prints 0
    y++;
    x++;
} // y stops existing on each iteration
System.out.println(x); // prints 10.
// trying to print y here would get a compiler error
```

Practice

- Complete all the learning activities in unit [4.1](#).

for loop

- Used when you know how many times you want the loop to execute

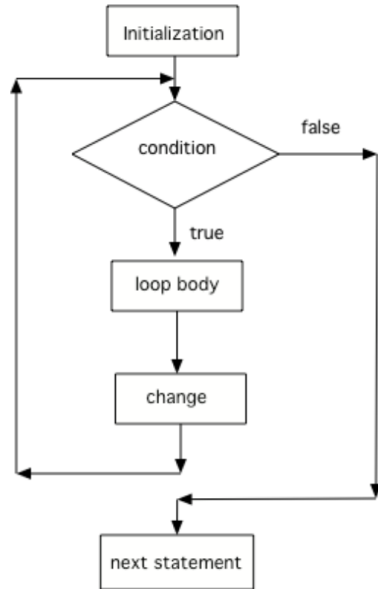


Figure 2: Control flow in a for loop

```
for (initialize; test condition; change)
{
    loop body
}
```

for vs while loop

```
for (int x = 3; x > 0; x--)  
{  
    System.out.println(x);  
}  
  
int x = 3;  
while (x > 0)  
{  
    System.out.println(x);  
    x = x - 1;  
}
```

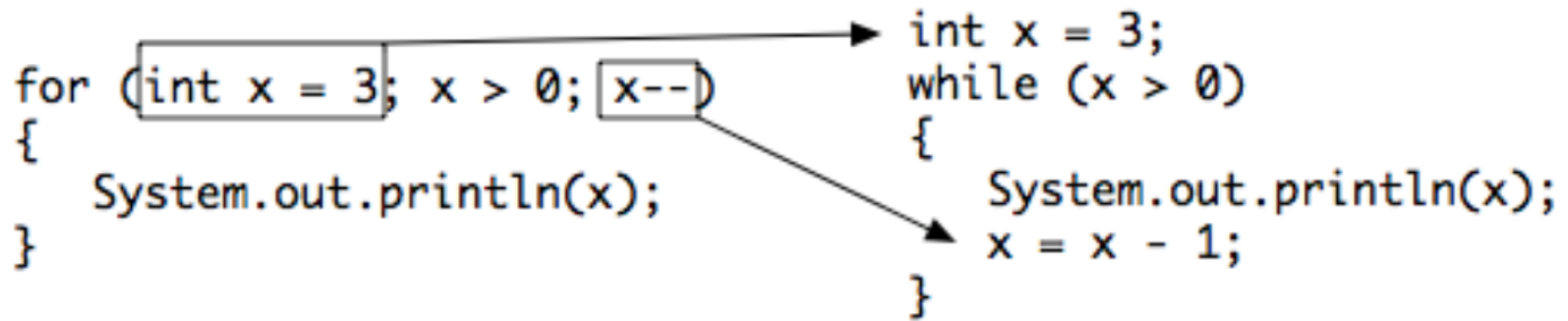


Figure 3: Showing how a for loop maps to a while loop

for loop patterns

```
// These loops both run 10 times  
// If you start at 0, use <  
for(int i = 0; i < 10; i++)  
{  
    System.out.println(i);  
}  
// If you start at 1, use <=  
for(int i = 1; i <= 10; i++)  
{  
    System.out.println(i);  
}
```

Practice

- Complete all the learning activities in unit [4.2](#).

Loops and Strings

- Loops are often used for **String Traversals** or **String Processing**
- **Traversing** a string involves going through a string character by character
- Characters are located based on their position(or index) in the string
- The **first character** in a Java String is at **index 0** and the **last character** is at **length() - 1**



Figure 1: A string with the position (index) shown above each character

Loops and Strings

```
String s = "welcome";  
int count = 0;  
while (count < s.length()) {  
    System.out.println(s.charAt(count));  
    count++;  
}
```

```
String s = "welcome";  
for (int i = 0; i < s.length(); i++) {  
    System.out.println(s.charAt(i));  
}
```

Both programs will print the characters in s one at a time

Practice

- Complete all the learning activities in unit [4.3](#).

Nested loops

- When a loop is located inside another one
- In each iteration of the outer loop, the inner loop will be re-started.
- The inner loop must finish all of its iterations before the outer loop can continue to its next iteration

```
Outer Loop [ for (int row = 1; row <= 3; row++)  
            {  
Inner Loop [   for (int col = 1; col <= 5; col++)  
              {  
                System.out.print("*");  
              }  
              System.out.println();  
            }  
          }
```

Figure 1: Nested Loops

Tracing nested loops

```
Outer Loop {  
  Inner Loop {  
    for (int row = 1; row <= 3; row++)  
    {  
      for (int col = 1; col <= 5; col++)  
      {  
        System.out.print("*");  
      }  
      System.out.println();  
    }  
  }  
}
```

Figure 1: Nested Loops

row	row <= 3	Output
1	true	
2	true	
3	true	
4	false	

col	col <= 5	Output
1	true	*
2	true	**
3	true	***
4	true	****
5	true	*****
6	false	

col	col <= 5	Output
1	true	***** *
2	true	***** **
3	true	***** ***
4	true	***** ****
5	true	***** *****
6	false	

Practice

- Complete all the learning activities in units [4.4](#), [4.5](#), [4.6](#), [4.7](#), [4.8](#), [4.9](#), [4.10](#), [4.11](#), [4.12](#), [4.13](#).