Looping

Jim Wilson
Mobile solutions developer & architect

@hedgehogjim jwhh.com

Overview

- Common aspects of loops
- While loop
- Do-while loop
- For loop
- For-each loop

Loops

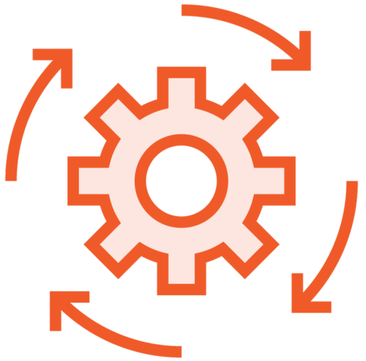
Repeatedly execute some code as long as provided condition is true

While loop

Basic looping

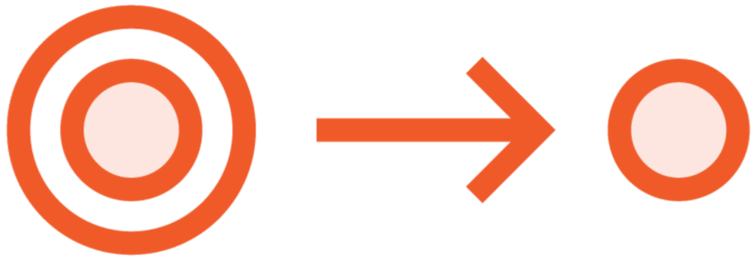
Do-while loop

Looping with deferred condition check



For loop

Looping with simplified notation for common use case



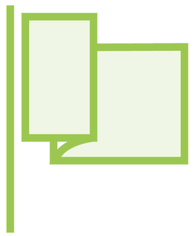
Loop body

- Code contained within a loop
- By default just a single statement
- Use block to include multiple statements

Loop iteration

- A single pass through the code contained within a loop

While Loop



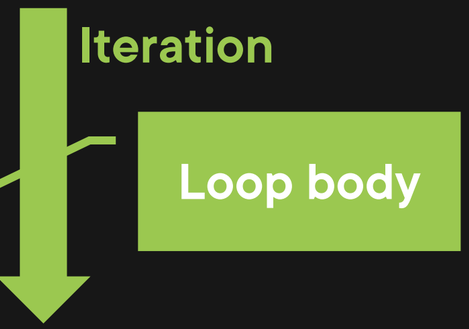
Condition checked at loop start

Loop body may never run

```
while ( condition )
  statement;
```

While Loop

```
int someValue = 4;

int factorial = 1;

while(someValue > 1)
  factorial *= someValue;

  someValue--;
}

System.out.println(factorial);
```



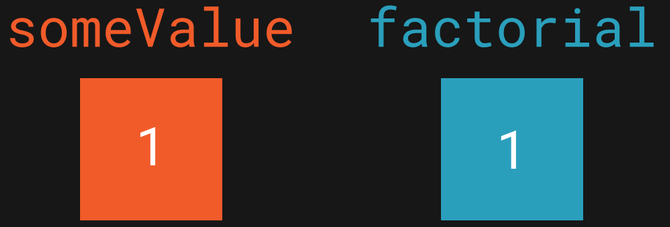
Loop body

While Loop

```
int someValue = 4;
int factorial = 1;
while(someValue > 1) {
  factorial *= someValue;
  someValue--;
}
System.out.println(factorial);
```

While Loop

```
int someValue = 1;
int factorial = 1;
while(someValue > 1) {
  factorial *= someValue;
  someValue--;
}
System.out.println(factorial);
```



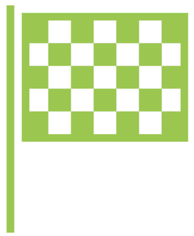
While Loop

```
int someValue = 1;
int factorial = 1;
while(someValue > 1) {
  factorial *= someValue;
  someValue--;
}
System.out.println(factorial); // displays 1
```

Boolean value, variable or expression

someValue	factorial
1	1

Do-while Loop



Condition checked at loop end

Loop body always runs at least once

```
do
  statement;
while ( condition );
```

DO-while Loop

Main.java

```
int iVal = 5;
do
  System.out.print(iVal + " * 2 = ");
  iVal *= 2;
  System.out.println(iVal);
  while(iVal < 25);
```

```
5 * 2 = 10
10 * 2 = 20
20 * 2 = 40
```

DO-while Loop

Main.java

```
int iVal = 80;
do {
  System.out.print(iVal + " * 2 = ");
  iVal *= 2;
  System.out.println(iVal);
} while(iVal < 25);
```

```
80 * 2 = 160
```

DO-while Loop

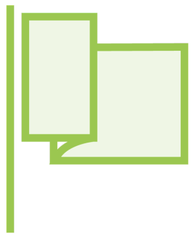
Main.java

```
int iVal = 80;
do {
   System.out.print(iVal + " * 2 = ");
   iVal *= 2;
   System.out.println(iVal);
} while(iVal < 25);
```

Boolean value, variable or expression

```
80 * 2 = 160
```

For Loop



Condition checked at loop start

```
for (initialize; condition; update)
  statement ;
```

Loop body may never run

Simplified notation for loop control values

For Loop

WhileLoop.java

```
int i = 1;
while(i < 100) {
  System.out.println(i);
  i *= 2;
}
```

ForLoop.java

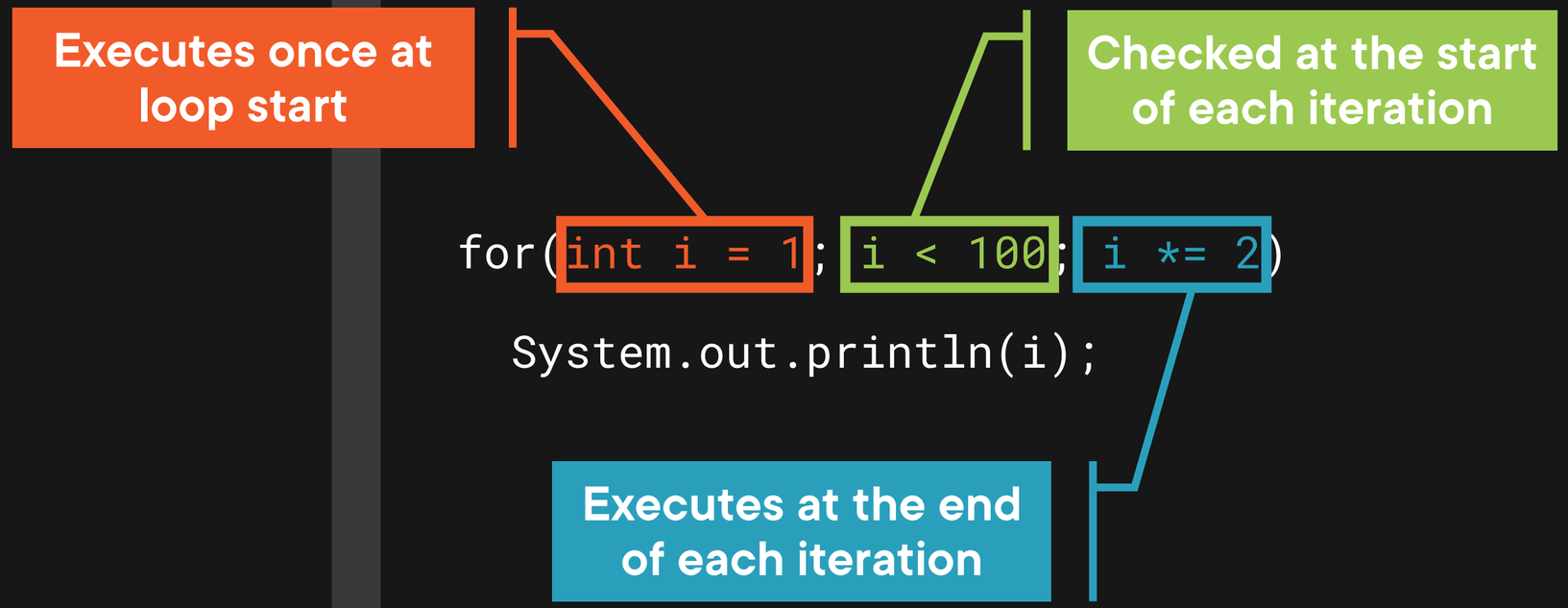
```
for(int i = 1; i < 100; i *= 2)
  System.out.println(i);
```

For Loop

WhileLoop.java

```
int i = 1;

while(i < 100) {

  System.out.println(i);

  i *= 2;

}
```

ForLoop.java



For Loop

Main.java

```
int factorial = 1;
for (int num = 3; num > 1; num--)
    factorial *= num;
    System.out.println(num + " | " + factorial);
}
System.out.println("Result:" + factorial);
```

```
3 | 3
2 | 6
Result:6
```

For Loop

Main.java

```
int factorial = 1;
for (int num = 3; num > 1; num--){
    factorial *= num;
    System.out.println(num + " | " + factorial);
}
System.out.println("Result:" + factorial);
```

```
3 | 3
2 | 6
Result:6
```

For Loop Control Variable

Main.java

```
int factorial = 1;
for (int num = 3; num > 1; num--){
    factorial *= num;
    System.out.println(num + " | " + factorial);

}

System.out.println("Result:" + factorial);
```

For Loop Control Variable

Main.java

```
int factorial = 1;

for (int num = 3; num > 1; num--){
    factorial *= num;
    System.out.println(num + " | " + factorial);


}

System.out.println(“Value of num is” + num);

System.out.println("Result:" + factorial);
```

For Loop Control Variable

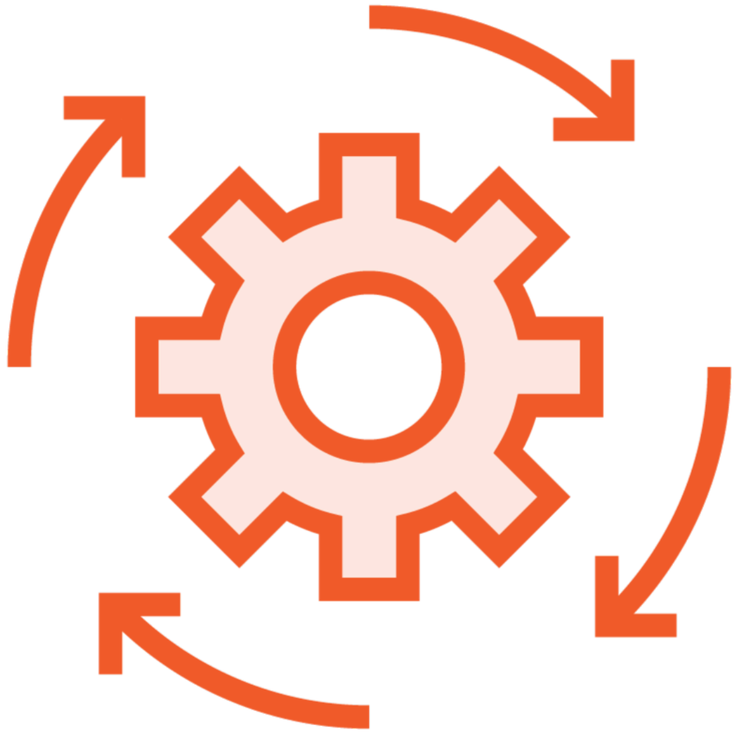
Main.java

```
int factorial = 1;
for (int num = 3; num > 1; num--){
    factorial *= num;
    System.out.println(num + " | " + factorial);
    num = num + 5;
}

System.out.println("Result:" + factorial);
```

This is legal!

```
3  |  3
7  |  21
11  |  231
```

For-each Loop



Simplifies iterating over a collection of items

Executes loop body once for each member

- Handles getting collection length
- Handles accessing each item

Types supported

- Arrays
- Any type that implements Iterable
- Most collection classes

For-each Loop

Foreach.java

```
float[] vals = { 10.0f, 20.0f, 15.0f };

float sum = 0.0f;

for(float currentVal : vals)

    sum += currentVal;

System.out.println(sum); // displays 45
```

TraditionalFor.java

```
float[] vals = { 10.0f, 20.0f, 15.0f };

float sum = 0.0f;

for(int i = 0; i < vals.length; i++)

    float currentVal = vals[i];

    sum += currentVal;

}

System.out.println(sum); // displays 45
```

For-each Loop Limitations

```
int[] left = {5, 3, 7};
int[] right = {12, 9, 8};
for(int i = 0; i < left.length; i++)
    int result = left[i] + right[i];
    System.out.println("result = " + result);
}
```

For-each Loop Limitations

```
int[] left = {5, 3, 7};
int[] right = {12, 9, 8};
for(int i = 0, j = right.length-1; i < left.length; i++, j--)
    int result = left[i] + right[j];
    System.out.println("result = " + result);
}
```

Summary

Loop control condition

- Boolean value, variable, or expression
- Loop repeatedly executes provided condition is true

Loop body

- Code contained within loop
- Multiple statements must be in block

Loop iteration

- A single pass through code within loop

Summary

While loop

- Condition checked at loop start
- Loop body may never run

Do-while loop

- Condition checked at loop end
- Loop body always runs at least once

Summary

For loop

- Condition checked at loop start
- Loop body may never run
- Simplified notation for loop control values

For loop control value

- Scope limited to within the loop
- Can be modified within the loop body

Summary

For-each loop

- Simplifies iterating over a collection of items

Types supported

- Arrays
- Any type that implements Iterable

Be aware of its limitations

- Designed for iterating sequentially through a single collection of items