Looping and Arrays

Jim Wilson
MOBILE SOLUTIONS DEVELOPER & ARCHITECT
@hedgehogjim jwhh.com

Overview

While loop

Do-while loop

For loop

Arrays

For-each loop

Loops

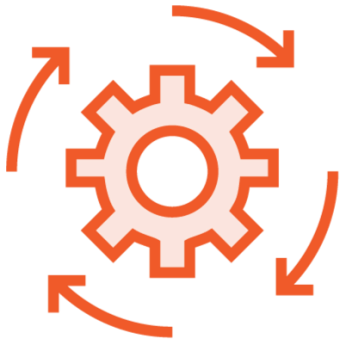
Repeatedly execute a statement as long the provided condition is true

While loop

Basic looping

Do-while loop

Looping with deferred condition check



For loop

Looping with simplified notation for common use case

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);
```

someValue

4

factorial

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);
  System.out.print(“ * 2 = “);
  iVal *= 2;
  System.out.println(iVal);
  while(iVal < 25);
```

```
5
10
20
```

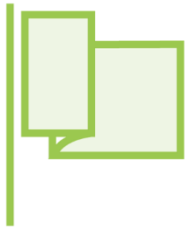
Do-while Loop

Main.java

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

```
80
```

For Loop



Condition checked at loop start

Loop body may never run

Simplified notation for loop control values

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

For Loop

WhileLoop.java

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

ForLoop.java

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

```
float[] theVals

theVals[0] = 10.0f;

theVals[1] = 20.0f;

theVals[2] = 15.0f;
```

Arrays

Provide an ordered collection of elements

- Each element accessed via an index
- Index range from 0 to number-of-elements minus 1
- Number of elements can be via array's length value

Arrays

```
float[] theVals = new float[3];
theVals[0] = 10.0f;
theVals[1] = 20.0f;
theVals[2] = 15.0f;
float sum = 0.0f;
for(int index = 0; index < theVals.length; index++)
    sum += theVals[index];
System.out.println(sum);
```

Arrays

```
float[] theVals = new float[3];
theVals[0] = 10.0f;
theVals[1] = 20.0f;
theVals[2] = 15.0f;
float sum = 0.0f;
for(int index = 0; index < theVals.length; index++)
    sum += theVals[index];
System.out.println(sum);
```

Arrays

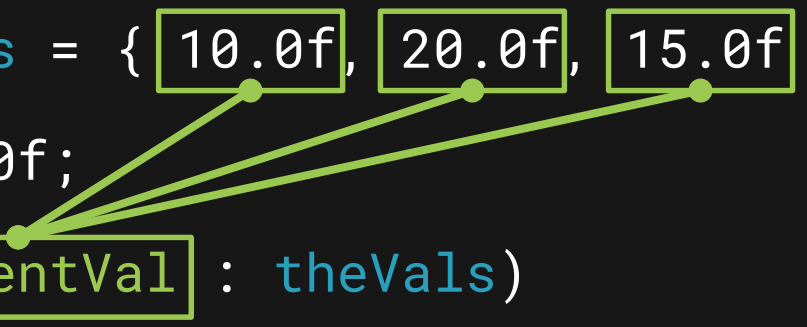
```
float[] theVals = new float[3];
theVals[0] = 10.0f;
theVals[1] = 20.0f;
theVals[2] = 15.0f;
float sum = 0.0f;
for(int index = 0; index < theVals.length; index++)
    sum += theVals[index];
System.out.println(sum); // displays 45
```

Arrays

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

```
float sum = 0.0f;
for(int index = 0; index < theVals.length; index++)
    sum += theVals[index];
System.out.println(sum); // displays 45
```

```
float[] theVals = { 10.0f, 20.0f, 15.0f };
float sum = 0.0f;
for(float currentVal : theVals)
    sum += currentVal;
System.out.println(sum); // displays 45
```



For-each Loop

Executes a statement once for each array member

Handles getting collection length

Handles accessing each value

```
for (loop-variable : array)
  statement ;
```

Summary

While loop

- Checks loop control condition at start
- Loop body may never run

Do-while loop

- Checks loop control condition at end
- Loop body always runs at least once

For loop

- Similar to a while loop
- Simplified notation for loop initialization and control

Summary

Arrays

- Ordered collection of elements
- Elements accessed via index
- Index is zero-based

For-each loop

- Simplifies working with array members
- Handles details of running the loop body once for each array member