

# Loops



Liang, Introduction to Java programming, 11<sup>th</sup> Edition, © 2017 Pearson Education, Inc.  
All rights reserved



By: Mamoun Nawahdah (Ph.D.)  
2020

## Opening Problem

Problem:

100  
times

```
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
...  
...  
...  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");
```



## Introducing **while** Loops

```
int count = 0;
while (count < 100) {
    System.out.println("Welcome to Java");
    count++;
}
```



3

## do-while Loop

```
do {
    // Loop body;
    Statement(s);
} while (loop-continuation-condition);
```



4

## for Loops

```
for ( initial-action ;
      loop-continuation-condition ;
      action-after-each-iteration ) {
    // loop body;
    Statement(s);
}
```

```
for (int i = 0 ; i < 100 ; i++) {
    System.out.println( "Welcome to Java!");
}
```



5

## 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 two **for** loops are correct:

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

```
for ( int i = 0 , j = 0 ; (i + j < 10) ; i++, j++ ) {
    // Do something
}
```



6

## 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.

```
for ( ; ; ) {
    // Do something
}
```

Equivalent

```
while (true) {
    // Do something
}
```



7

## 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*



8

## Caution

- ❖ 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 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
```



9

## 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);
    }
}
```



10

## 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);
    }
}
```



11

## Problem: Displaying Prime Numbers

Problem: Write a program that displays the first 50 prime numbers in five lines, each of which contains 10 numbers. An integer greater than 1 is *prime* if its only positive divisor is 1 or itself. For example, 2, 3, 5, and 7 are prime numbers, but 4, 6, 8, and 9 are not.

Solution: The problem can be broken into the following tasks:

- For number = 2, 3, 4, 5, 6, ..., test whether the number is prime.
- Determine whether a given number is prime.
- Count the prime numbers.
- Print each prime number, and print 10 numbers per line.



12