# Java Method Parameters

‹ Previous Next ›

## Parameters and Arguments

Information can be passed to methods as parameter. Parameters act as variables inside the method.

Parameters are specified after the method name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.

The following example has a method that takes a `String` called **fname** as parameter. When the method is called, we pass along a first name, which is used inside the method to print the full name:

### Example

```
public class Main {

    System.out.println(fname + " Refsnes");
  }

  public static void main(String[] args) {
```

```
  }
}
// Liam Refsnes
// Jenny Refsnes
// Anja Refsnes
```

Try it Yourself »

When a **parameter** is passed to the method, it is called an **argument**. So, from the example above: `fname` is a **parameter**, while `Liam`, `Jenny` and `Anja` are **arguments**.

# Multiple Parameters

You can have as many parameters as you like:

## Example

```
public class Main {

    System.out.println(fname + " is " + age);
  }

  public static void main(String[] args) {
    myMethod("Liam", 5);
    myMethod("Jenny", 8);
    myMethod("Anja", 31);
  }
}

// Liam is 5
// Jenny is 8
// Anja is 31
```

Try it Yourself »

Note that when you are working with multiple parameters, the method call must have the same number of arguments as there are parameters, and the arguments must be passed in the same order.

# Return Values

The `void` keyword, used in the examples above, indicates that the method should not return a value. If you want the method to return a value, you can use a primitive data type (such as `int`, `char`, etc.) instead of `void`, and use the `return` keyword inside the method:

## Example

```
public class Main {
  static int myMethod(int x) {

  }

  public static void main(String[] args) {
    System.out.println(myMethod(3));
  }
}
// Outputs 8 (5 + 3)
```

Try it Yourself »

This example returns the sum of a method's **two parameters**:

## Example

```
public class Main {
  static int myMethod(int x, int y) {
    return x + y;
  }

  public static void main(String[] args) {
    System.out.println(myMethod(5, 3));
  }
}
// Outputs 8 (5 + 3)
```

Try it Yourself »

You can also store the result in a variable (recommended, as it is easier to read and maintain):

## Example

```
public class Main {
  static int myMethod(int x, int y) {
    return x + y;
  }

  public static void main(String[] args) {
    int z = myMethod(5, 3);
    System.out.println(z);
  }
}
// Outputs 8 (5 + 3)
```

Try it Yourself »

# A Method with If...Else

It is common to use `if...else` statements inside methods:

## Example

```
public class Main {

  // Create a checkAge() method with an integer variable called age
  static void checkAge(int age) {

    // If age is less than 18, print "access denied"
    if (age < 18) {
      System.out.println("Access denied - You are not old enough!");

    // If age is greater than, or equal to, 18, print "access granted"
    } else {
      System.out.println("Access granted - You are old enough!");
    }

  }

  public static void main(String[] args) {
    checkAge(20); // Call the checkAge method and pass along an age of 20
  }
```

```
}

// Outputs "Access granted - You are old enough!"
```

Try it Yourself »

## Test Yourself With Exercises

### Exercise:

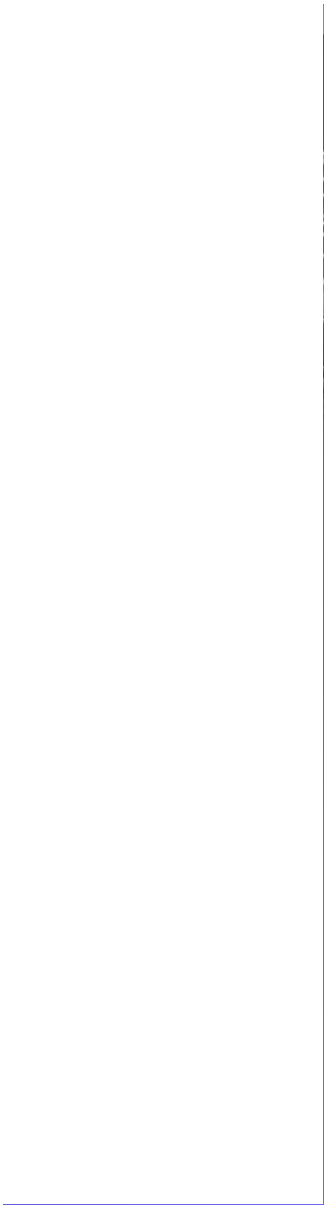
Add a `fname` parameter of type `String` to `myMethod`, and output "John Doe":

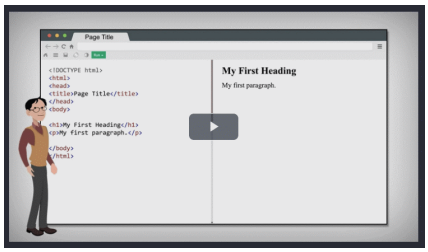
```
static void myMethod(              ) {
  System.out.println(       + " Doe");
}

public static void main(String[] args) {
  myMethod("John");
}
```

Submit Answer »

Start the Exercise





Play Game

## Top Tutorials

HTML Tutorial
CSS Tutorial
JavaScript Tutorial