Methods

… a collection of statements that are grouped together to perform an operation

```
modifiers type name(parameters) {
    statements;
}
```


Methods that do not return a value

```
modifiers void name(parameters) {
    statements;
}
```

For example:

```
public static void printSum(int x, int y) {
    int result = x + y;
    System.out.println(result);
}
```

Methods that do **not** return a value

```
modifiers void name(parameters) {
    statements;
}
```

Return type is void

For example:

```
public static void printSum(int x, int y) {
    int result = x + y;
    System.out.println(result);
}
```


Calling a method

An example:

2 parameters

```
public static void sum(int x, int y) {
    int result = x + y;
    System.out.println(result);
}
```

The arguments in the method call must evaluate to the same type as the method's parameters

```
…
int a = 5
printSum(3, a + 2);
…
```

2 arguments

Calling a method

An example:

```
public static void printTrue() {
    System.out.println("true");
}

…
if (a == 5)
    printTrue();
…
```

no parameters

A void method does not return anything. So a call to such a method cannot be used as part of an expression

no arguments

Methods that return a value

```
modifiers type name(parameters) {
    statements;
    return expression;
}
```

For example:

```
public static int sum(int x, int y) {
    int result = x + y;
    return result;
}
```

Methods that do **not** return a value

```
modifiers void name (parameters) {
    statements;
    return;
}
```

For example:

```
public static void sum(int x, int y) {
    System.out.println(x + y);
    return;
}
```


Methods that return a value

```
modifiers type name(parameters) {
    statements;
    return expression;
}
```

The return statement's expression must evaluate to a value of the specified type

For example:

```
public static int sum(int x, int y) {
    int result = x + y;
    return result;
}
```

Calling a method

An example:

2 parameters

```
public static int sum(int x, int y) {
    int result = x + y;
    return result;
}

…
int a = 5
int answer = sum(3, a + 2);
…
```

2 arguments

The arguments in the method call must evaluate to the same type as the method's parameters

Calling a method

An example:

no parameters

```
public static int getOne() {
    return 1;
}

…
int a = 5
int answer = a + getOne();
…
```

no arguments

If the method has no parameters then the method call must have no arguments

Some additional terminology

- **parameters – "**parameters" are *a part of the definition of a method*. They indicate the type of data that is expected when the method is called. Java parameters are variables defined in the method header that have a type and name.

- **arguments – "**arguments" are *the values that are provided to a method when it is called*. They must have the same type as the parameters that receive the data.

Some additional terminology

- **functions –** Methods that return a value are often called "functions".

- **void methods –** Methods that do **not** return a value.

- **non-void methods –** Methods that have a return type that is ***not*** void. In other words, non-void methods are functions.