Parameters and Return Types in Methods

In this lesson, an explanation of how to pass parameters to the methods is explained.

We'll cover the following

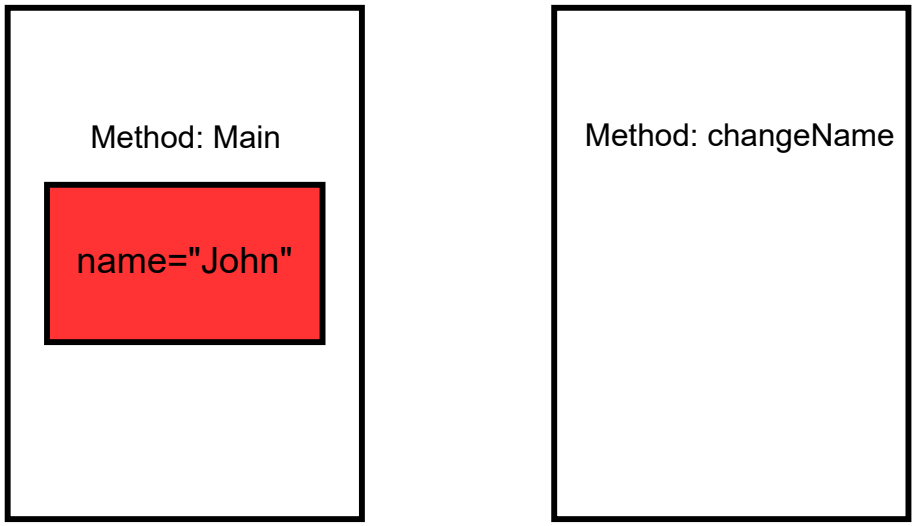
Methods take in **two** types of parameters. One is the **primitive data type** and the other is the **reference data type**. We will be looking in detail at both types of parameters and how each of them is used for a different purpose.

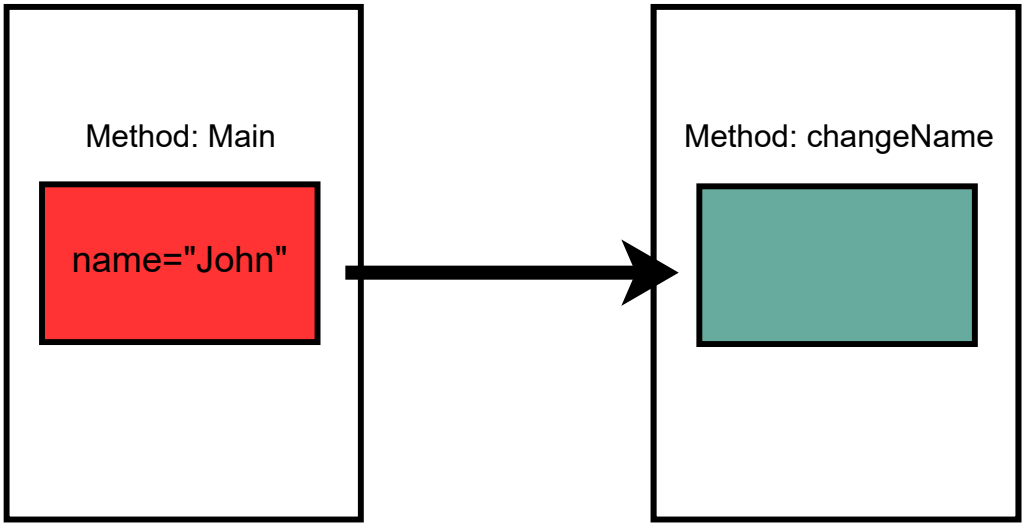
Primitive type

These types of parameters are said to be *passed by value*. The value from the calling method IS copied to a variable in the called method. What this signifies is that once a parameter is passed or given to a method as an argument, there is no relation between the object i.e. the parameter that is present outside the method body, and the modification that is happening inside the method to that parameter. This can be a bit hard to understand.

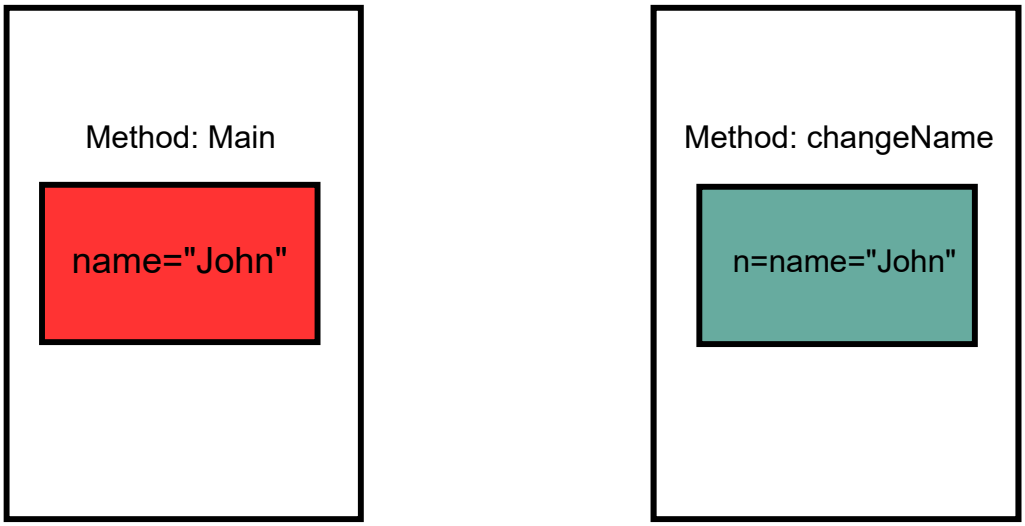
The code and diagram below explain this concept.

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

        String name = "John";
        changeName(name);
        System.out.print(name);
    }

    public static void changeName(String n) {

        n = "Doe";
        System.out.println(n);

    }
}
```

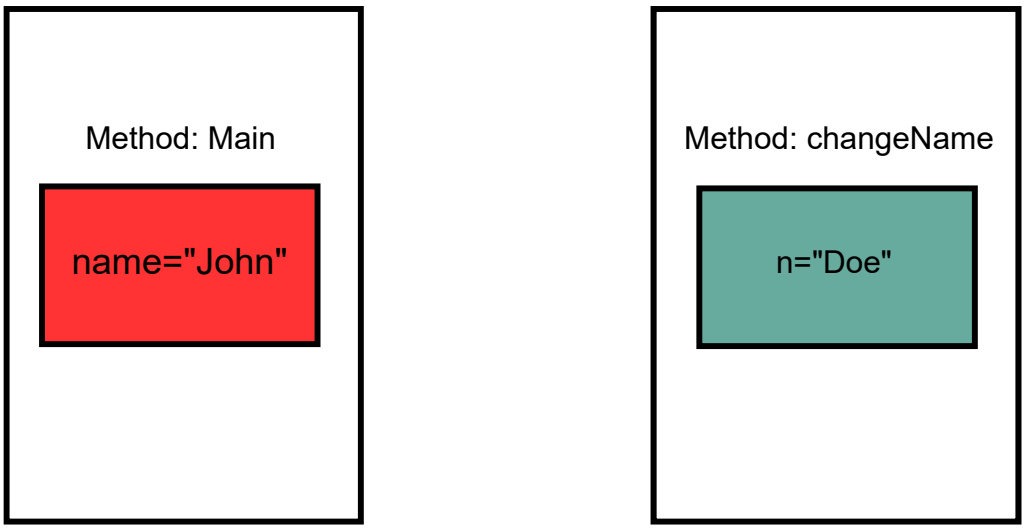




Giving 'name' as parameter to changeName



The variable within the method is n, this is a variable only relevant to the method.



n="Doe" only within the method, it has no link to the variable "name"

In the above code, we could have passed a string literal instead of a string variable in the method call.

Hence, we see that printing the variable **n** in the method gives the result as **Doe**, whereas outside the defined method, the variable value is retained as **John**.

> **Note:** The method call must match the method definition.

Reference type

In this type of parameter passing, the **reference to the object** is passed by value. This **does not** mean that the object itself is passed rather a **link** between the variable name and the object instance is passed. An **object instance** is a variation of an object within the method. This link is called an **object reference**, which simply points to the address where the object instance is kept in memory.

One analogy that we can use with value and reference parameters is a one-way and a two-way pipe between the calling and the called method. With value types, there is a one-way of passing value between them. With reference types, changes can be propagated back to the calling method. With reference types actually it is one variable that both are accessing.

```
class car {

    public String colour;
    public car() {};
    public car(String col) {
        this.colour = col;
    }
    public String getColour() {
        return this.colour;
    }
    public static void main(String[] args) {
        car newCar = new car("Pink");
        System.out.print("Car colour is: " + newCar.getColour() + "\n");
        setColourBlue(newCar);
        System.out.print("Car colour is: " + newCar.getColour() + "\n");
    }

    public static void setColourBlue(car c) {
        c.colour = "Blue";
    }
}
```

Understanding the code

This code snippet is slightly different from what we have seen so far, so let's look at it in a little bit more detail.

Line 3: A `String colour` is created which will strore the car colour.

Lines 5-10: Helper methods are created within the class to `set` and `get` the car colour.

Lines 11-16:

- A `main` method creates a new car object and the initial colour of the car is *set* as `Pink`.
- The method that returns the car colour is called.
- Then the car colour is set to `Blue` using an external method which is explained below (line 18-20).
- The car colour is then printed again, to see if the changes within the above-mentioned method are **applicable** to the **actual object**.

Lines 18-20

- A new method is declared that takes in the `car` object as an argument.
- Then uses the **colour String** belonging to the `car class` and *modifies* its value to `Blue`.
- Note that the function **does not** return anything.

In the next lesson, we will discuss return parameters in methods.