Class Member Methods

In this lesson, an explanation of how methods can be made within classes is provided.

We'll cover the following

Defining member methods

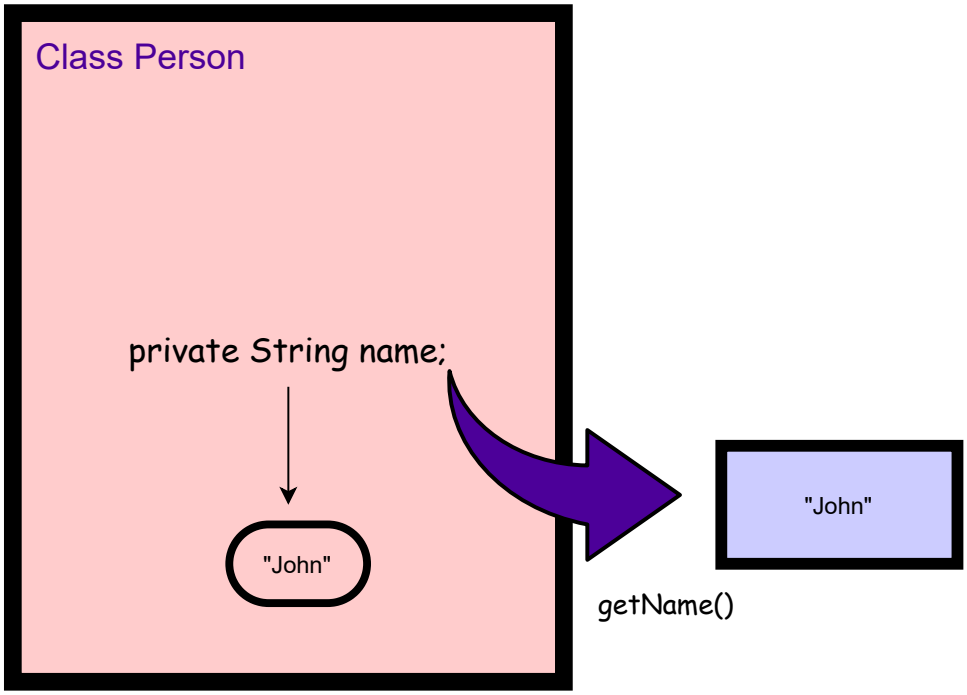
Member methods are *declared* where the class is defined. The methods are a **member** of the class within which they are written. These methods are also used to access the private parts of the class and help us perform various actions on an object through them.

Setters and Getters

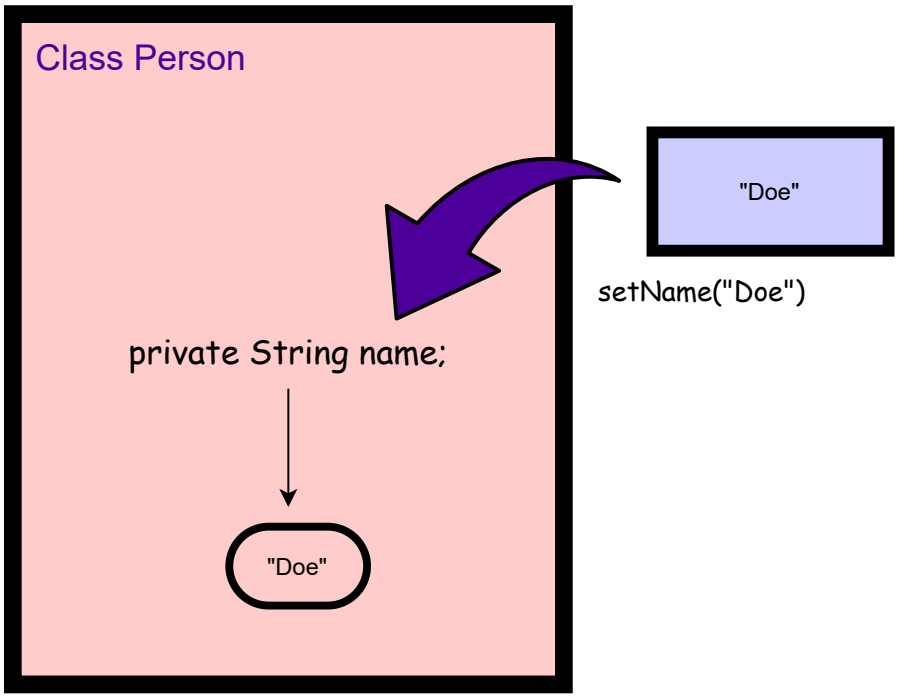
As we discussed, in the previous lesson, `private` *member* variables cannot be accessed directly outside the class.

In order to *access* or *change* their values, we need to *define* `public` *member* methods. These *methods* can be used to *set* the values of the `private` variables as well as to *get* their values since being `private` these members cannot be accessed directly.

Take a look at the example below to understand this better.



Getter Method



Setter Method

```
class Pet {

    private static int newID;
    private int petID;
    private String petType;
    private String petName;
    private int petAge;

    public Pet(String type, String name, int age) {
        petType = type;
        petID = newID;
        petName = name;
        petAge = age;

        newID = newID + 1;
    }

    public String getName() {
        return petType;
    }

    public static int getNewID() {
        return newID;
    }

    public void setName(String new_name) {
        petName = new_name;
    }

    public void printPetDetails() {
        System.out.println("Pet ID: " + petID);
        System.out.println("Pet Type: " + petType);
        System.out.println("Pet Name: " + petName);
        System.out.println("Pet Age: " + petAge);
    }

}

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

        Pet myDog = new Pet("dog", "Ruffy", 45);
        myDog.printPetDetails();
        myDog.setName("Scooby");
        myDog.printPetDetails();

        Pet newcat = new Pet("cat", "Princess", 2);
        newcat.printPetDetails();
    }
}
```

Example explanation

- First we make a class named `Pet` and declare the `public` and `private` *members*.
- `Public` members include:
    - *Methods*: `getName`, `setName` and `printPetDetails`
- `Private` members include:
    - *Variables*: `petID`, `petType`, `petName` and `petAge`

Let's take a look at all the *methods* one by one.

getName()

- It simply returns the **petName** variable, hence allowing the program to access a *private* variable. This is an example of a **getter** method.

getNewID()

- It simply returns the **newID** variable, hence allowing the program to access a *private* variable. This is also an example of a **getter** method.

setName(String new_name)

- Since the `petName` variable is **private** it cannot be changed directly by the `main()` method outside of the Pet class as we saw in the last lesson.
- Hence, we have a **setter** method which takes a parameter i.e. the new value of the name of the pet and *sets* the **current** `petName` to the `new_name`.

printPetDetails():

- Displays the various properties within the Pet class i.e the `name`, `age`, `type` and the `ID`.

main(String[] args)

- Declares **two** objects `myDog` and `newCat` for class `Pet`.
- Calls a `constructor` method to create these objects.
- *Calls* the `printPetDetails` method to show the values for each variable within object `myDog`.
- *Calls* `set` function to *update* the value for `petName` for `myDog`.

- *Calls* the `get` method to show the change for `petName`.

Now that we know how to write member methods, we move on to inheritance in Java in the next lesson.