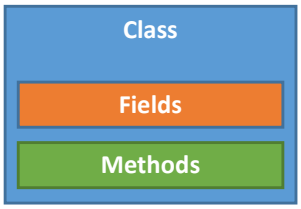
Getters and Setters

Members

A **member** of a class is any **method** or **field** variable defined inside the scope of that class.



A class can only contain fields and methods

Data Encapsulation

A **public** field variable is externally accessible.

```
public class Employee
{
    public String firstName = "John";
    public String lastName = "Doe";
    public String title = "Mr";
}
```

This means that you can not only **read** it, but **modify** it as well.

```
Employee emp1 = new Employee();
String fname = emp1.firstName;
emp1.firstName="Joe";
```

But how can we make it **read-only**?

Data Encapsulation

We can make the field **private** and access it through a **method** which gives us **read only** access instead.

Accessible inside Employee only

Declare and assign

Provides read access to the field variables

```
public class Employee {
    private String firstName = "John";
    private String lastName = "Doe";
    private String title = "Mr";

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String getTitle() {
        return title;
    }
}
```

Getters should always start with get.....

Data Encapsulation

You can expose several field variables at once through a method.

Your methods depend on how you want your class to behave externally.

```
public class Employee {
    private String firstName = "John";
    private String lastName = "Doe";
    private String title = "Mr";

    public String getEntireNameAndTitle() {
        return title + " " + firstName + " " + lastName;
    }
}
```

Concatenate all fields into one string

Data Encapsulation

There are some problems with the current solution.

For example, we can't change the employee's title or name, they are always called "Mr John Doe".

We still need to be able to change the user's data, both when we create them and later.

Can you think of a way we could do this?

Data Encapsulation

Making sure the title and name can be changed.

Option 1: Make the fields **public** again.

While this WOULD work, we generally want to **encapsulate** our fields. Encapsulation lets us change our mind later, without breaking code, on how we handle and store data internally in classes.

Option 2: Use **Getters/Setters**.

Getters/Setters

Getters/Setters are normal class **member methods**, which we can use to provide a way to read, write or compute the value of private fields.

This is an answer to the fact that **data hiding** and **encapsulation** are important parts of **OOP**.

Unlike some other object oriented programming languages, such as C#, Java does not have **properties**. We can perform the same task simply using methods, however.

Second example

Example with getters and setters

For instance, a class storing data about a **TimePeriod**, measured in seconds, might use the following approach:

```
public class TimePeriod {
    private double seconds;

    public double getHours() {
        return seconds / 3600;
    }

    public void setHours(double hours) {
        seconds = hours * 3600;
    }
}
```

We can **get** the value of the field, but re-calculated as hours

These methods encapsulates the seconds field. Since they are public methods, they can be used outside of this class

We can **set** the value of the field. In this case, we're providing the time in hours, letting the **setter** calculate the number of seconds

Example with getters and setters

We could add a second getter and setter for minutes.

```
public class TimePeriod {
    private double seconds;

    public double getHours() {
        return seconds / 3600;
    }

    public void setHours(double hours) {
        seconds = hours * 3600;
    }

    public double getMinutes() {
        return seconds / 60;
    }

    public void setHours(double minutes) {
        seconds = minutes * 60;
    }
}
```


Best practices

Getters/Setters vs. public fields

To allow safe access to internal state in an object, you should use **getters/setters**.

These give you an abstraction between the internal implementation and the public contract.

Getters/Setters can be read or write only.

Exercise 14

Let's do exercise 14