

Abstraction: <https://stackify.com/oop-concept-abstraction/>

Its main goal is to handle complexity by hiding unnecessary details from the user. That enables the user to implement more complex logic on top of the provided abstraction without understanding or even thinking about all the hidden complexity.

That’s a very generic concept that’s not limited to object-oriented programming. You can find it everywhere in the real world.

### **Abstraction in the real world**

I’m a coffee addict. So, when I wake up in the morning, I go into my kitchen, switch on the coffee machine and make coffee. Sounds familiar?

Making coffee with a coffee machine is a good example of abstraction.

You need to know how to use your coffee machine to make coffee. You need to provide water and coffee beans, switch it on and select the kind of coffee you want to get.

The thing you don’t need to know is how the coffee machine is working internally to brew a fresh cup of delicious coffee. You don’t need to know the ideal temperature of the water or the amount of ground coffee you need to use.

Someone else worried about that and created a coffee machine that now acts as an abstraction and hides all these details. You just interact with a simple interface that doesn’t require any knowledge about the internal implementation.

You can use the same concept in object-oriented programming languages like Java.

## **Abstraction in OOP**

Objects in an OOP language provide an abstraction that hides the internal implementation details. Similar to the coffee machine in your kitchen, you just need to know which methods of the object are available to call and which input parameters are needed to trigger a specific operation. But you don’t need to understand how this method is implemented and which kinds of actions it has to perform to create the expected result.

Let’s implement the coffee machine example in Java. You do the same in any other object-oriented programming language. The syntax might be a little bit different, but the general concept is the same.

## **Summary**

Abstraction is a general concept which you can find in the real world as well as in OOP languages. Any objects in the real world, like your coffee machine, or classes in your current software project, that hide internal details provide an abstraction.

These abstractions make it a lot easier to handle complexity by splitting them into smaller parts. In the best case, you can use them without understanding how they provide the functionality. And that not only helps you to split the complexity of your next software project into manageable parts, it also enables you every morning to brew a fresh cup of amazing coffee while you’re still half asleep.

**What is Encapsulation** <https://www.c-sharpcorner.com/blogs/abstraction-and-encapsulation1>

Encapsulation is one of the fundamental [concepts in object-oriented programming](https://stackify.com/oops-concepts-in-java/)(OOP). It describes the idea of bundling data and methods that work on that data within one unit, e.g., a class in Java.

This concept is also often used to hide the internal representation, or state, of an object from the outside. This is called[information hiding](https://en.wikipedia.org/wiki/Encapsulation_(computer_programming)#An_information-hiding_mechanism). The general idea of this mechanism is simple. If you have an attribute that is not visible from the outside of an object, and bundle it with methods that provide read or write access to it, then you can hide specific information and control access to the internal state of the object.

If you’re familiar with any object-oriented programming language, you probably know that these methods as getter and setter methods. As the names indicate, a getter method retrieves an attribute, and a setter method changes it. Depending on the methods that you implement, you can decide if an attribute can be read and changed, or if it’s read-only, or if it is not visible at all. As I will show you later, you can also use the setter method to implement additional validation rules to ensure that your object always has a valid state.

Let’s take a look at an example that shows the concept of encapsulation and how you can use it to implement information hiding and apply additional validation before changing the values of your object attributes.

Encapsulation in Java

If you’ve read my previous post about [abstraction](https://stackify.com/oop-concept-abstraction/), you already saw several examples for encapsulation. It’s such a basic concept that most Java developers use it without thinking about it. It’s simply how you design a Java class. You bundle a set of attributes that store the current state of the object with a set of methods using these attributes.

## Summary

Encapsulation is one of the [core concepts in object-oriented programming](https://stackify.com/oops-concepts-in-java/). It describes the bundling of data and methods operating on this data into one unit.

It is often used to implement an information-hiding mechanism. This mechanism reduces the accessibility of attributes to the current class and uses public getter and setter methods to control and restrict external access to these attributes. These methods not only allow you to define which attributes can be read or updated, but it also enables you to validate the new value before changing the attribute.

## **Difference between Abstraction and Encapsulation**

Abstraction is a process. It is the act of identifying the relevant qualities and behaviors an object should possess. Encapsulation is the mechanism by which the abstraction is implemented.

|  |  |
| --- | --- |
| Abstraction | Encapsulation |
| Abstraction solves the problem in the design level. | Encapsulation solves the problem in the implementation level. |
| Abstraction is used for hiding the unwanted data and giving onlyrelevant data. | Encapsulation is hiding the code and data into a single unit toprotect the data from outer world. |
| Abstraction is set focus on the object instead of how it does it. | Encapsulation means hiding the internal details or mechanics of howan object does something. |
| Abstraction is outer layout in terms of design.  For Example: - Outer Look of a iPhone, like it has a display screen. | Encapsulation is inner layout in terms of implementation. For Example: - Inner Implementation detail of a iPhone, how DisplayScreen are connect with each other using circuits |