**Abstract:**

Abstraction is one of the [key concepts](https://stackify.com/oops-concepts-in-java/) of object-oriented programming (OOP) languages. 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. We can find it everywhere in the real world.

It’s main purpose is to hide details, and only expose a high-level mechanism for using it. Implementation details should not be shown.

An example of abstraction, is an [**abstract class**](https://medium.com/@ellehallal/interfaces-abstract-classes-and-concrete-classes-13af02ae96cf). An abstract class can contain abstract methods. This means a class extending from the abstract class, needs to implement the abstract methods. It isn’t concerned with how it is done, as long as the abstract methods are implemented.

**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.

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