What Is Polymorphism?

In this lesson, the concept of polymorphism will be explained which is an important part of OOP.

WE'LL COVER THE FOLLOWING

Definition

> The word **Polymorphism** is a combination of two Greek words, **Poly**, which means *many*, and **Morph** which means *forms*.

In OOP, *polymorphism* refers to the same object exhibiting different forms and behaviors.

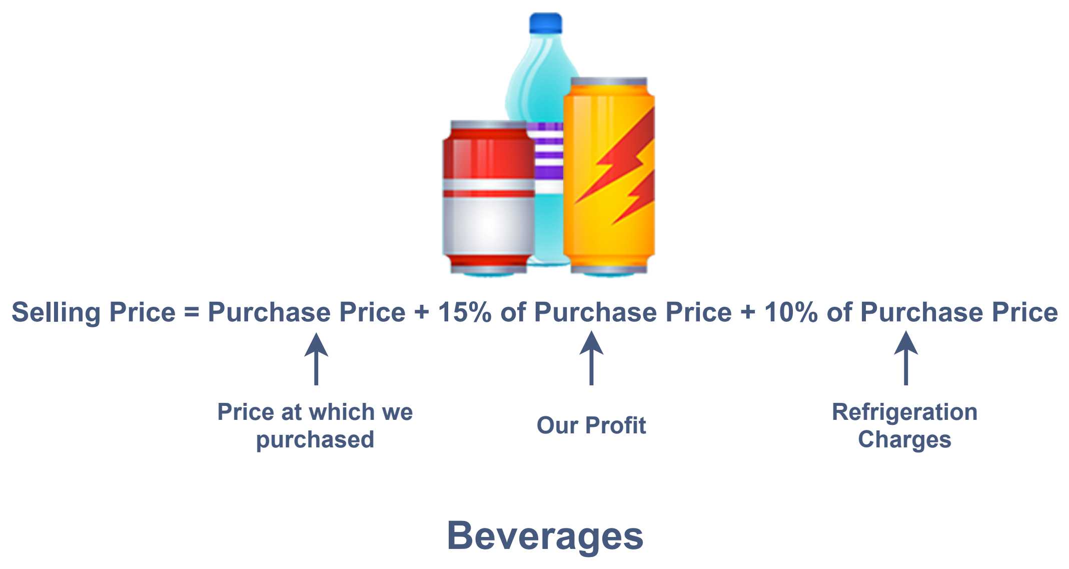
Let's have a look at the example of a vending machine. We have several products in it. The customer can buy any of these products. All of these products have different properties. This is an example of **Polymorphism**.

A Brief Introduction

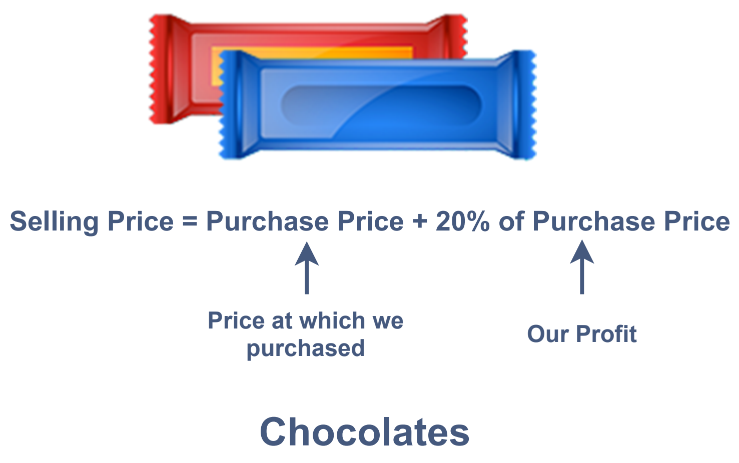
Assume there is a base class named `Product` from which the child classes `Beverage`, `Chocolate`, and `Snack` are derived.

`Beverage`, `Chocolate`, and `Snack` are derived.

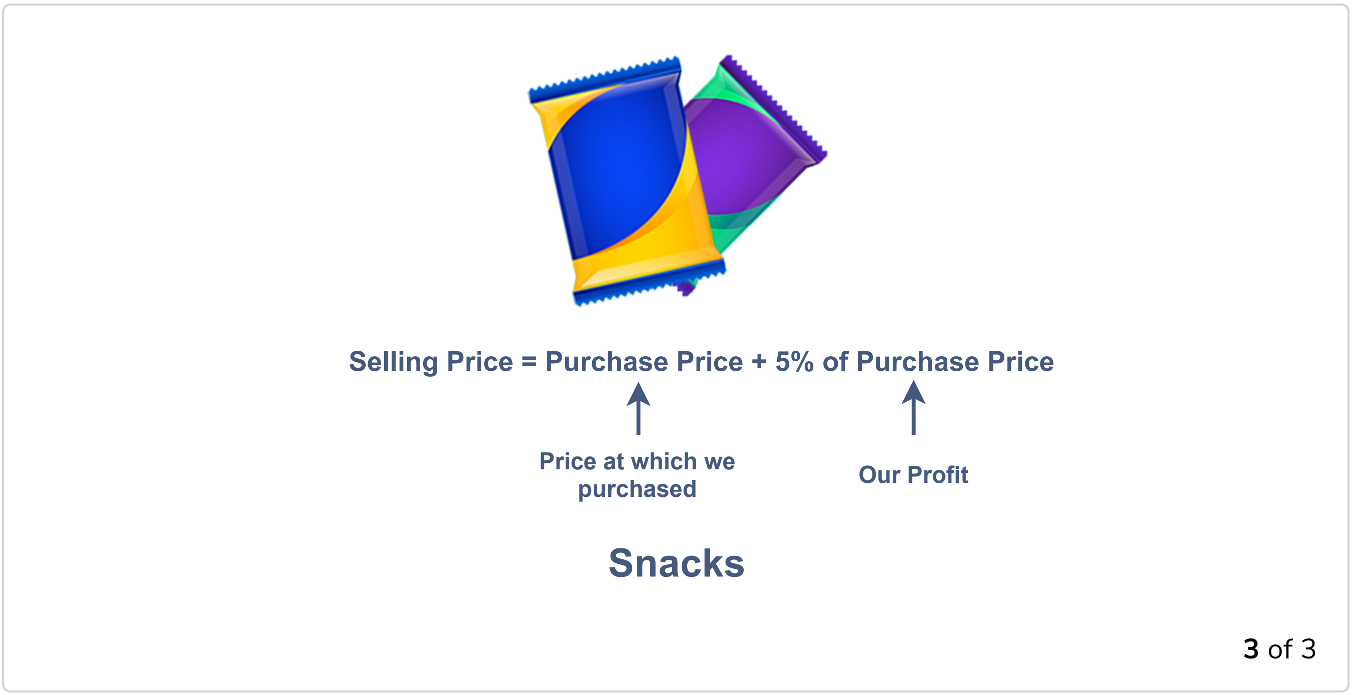
In our vending machine, these products are placed at locations marked by numbers. A customer comes to our machine and inputs the location of a specific product using the keypad. Now in return, the machine’s screen has to let the customer know about the selected product’s price. We, the owners of the machine, have decided on specific criteria to calculate the selling price for each category of the products as follows:


Selling Price = Purchase Price + 15% of Purchase Price + 10% of Purchase Price
Price at which we purchased
Our Profit
Refrigeration Charges
Beverages

1 of 3


Selling Price = Purchase Price + 20% of Purchase Price
Price at which we purchased
Our Profit
Chocolates

2 of 3


Selling Price = Purchase Price + 5% of Purchase Price
Price at which we purchased
Our Profit
Snacks
3 of 3

Of course, the selling price for each product is calculated differently. So, you can't have a single implementation. You could throw in separate methods in each class called, for instance, *GetChocolatePrice()*, *GetBeveragePrice()*, etc., but this makes it harder to remember each method's name. Wouldn't it be nice if all specific price calculation methods could be called *GetPrice()*? That's where polymorphism comes into play!

Make Things Simpler with Polymorphism

You only have to remember one method name. When you call that method on an object, the implementation, specific to that object is invoked. This can be achieved in object-oriented programming using polymorphism. The base class declares a method *GetPrice()* and may or may not provide an implementation. If it doesn't provide an implementation, all the derived classes have a common method name to work with. If it does provide an implementation, the derived classes can share that functionality and extend it, if needed.

For example, when the `GetPrice()` method is called on an object of the `Beverage` class, the method will respond by displaying the selling price of the beverage. On the other hand, when the same method is called on an object of the `Chocolate` class, the chocolate's price will be calculated and displayed on the screen, so on and so forth.

What Does Polymorphism Achieve?

In effect, polymorphism cuts down work for developers. When the time comes to create more specific derived classes with unique attributes and behaviors, the developers can alter the code in the particular portions where the responses differ. All other pieces of the code can be left untouched.

So far, we've learned what polymorphism is. In the next lesson, we will learn how to implement polymorphism using inheritance in OOP.