Method Overriding

In this lesson, you'll learn about what method overriding is and how to achieve it in C#.

WE'LL COVER THE FOLLOWING

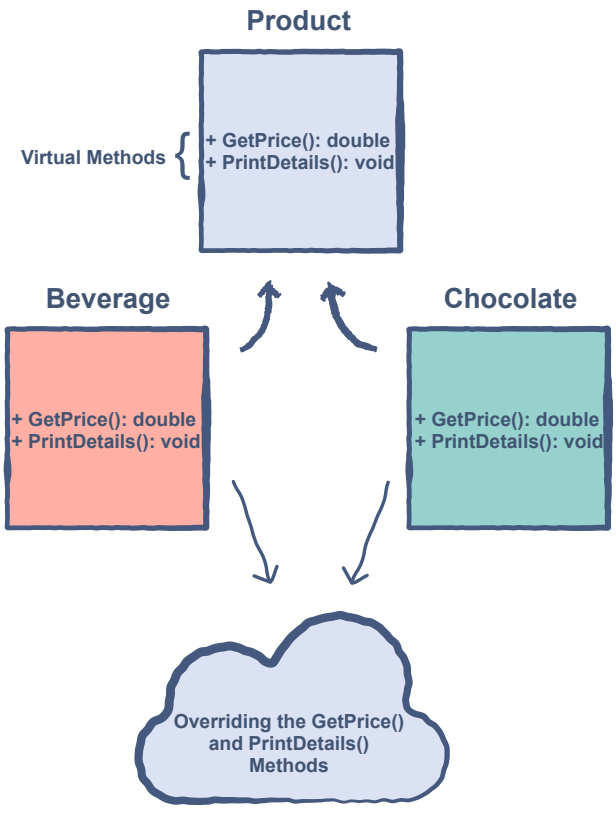
A Brief Introduction

> *Method overriding* is the process of redefining a parent class' method in a subclass.

In other words, if a subclass provides the specific implementation of a method that has been declared by one of its parent classes, it is known as **method overriding.**

In the previous example, the `Beverage` and `Chocolate` classes were overriding the `GetPrice()` and `PrintDetails()` methods from the `Product` class.

Overriding is done so that a child class can give its own implementation to a method which is already provided by the parent class.



In this case:

- The methods in the parent class are called **overridden methods** and have to be declared `virtual`.
- The methods in the child classes are called **overriding methods** and have to include the keyword `override` in their signatures.

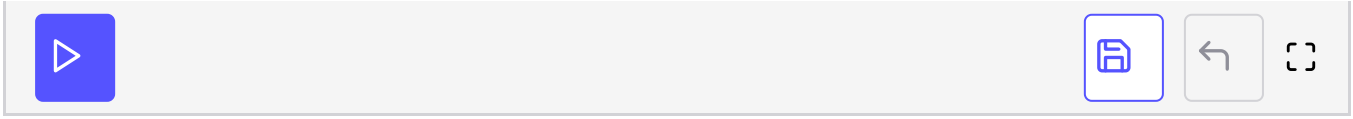
We have already seen the implementation of the `GetPrice()` and `PrintDetails()` methods in the previous lesson, which depicts the concept of overriding. The *highlighted* portions show where method overriding is happening.

Let's have a look!

```
class Product
{
    private string _name;
    private double _purchasePrice;

    // Parameterized Constructor
    public Product(string name, double purchasePrice)
    {
        this._name = name;
        this._purchasePrice = purchasePrice;
    }

    // Getters
    public string GetName()
    {
        return this._name;
    }

    public double GetPurchasePrice()
    {
        return this._purchasePrice;
    }
    // Method to calculate selling price
    public virtual  double GetPrice()
    {
        return 0;
    }
    // Method to print details
```

```
    public virtual void PrintDetails()
    {
        Console.WriteLine("Selected Product's Name: " + this.GetName());

    }

}


class Beverage : Product
{


    private double _refCost;
    private double _profit;

    // Parameterized Constructor
    public Beverage(string name, double price)
        : base(name,price)
    {
        this._refCost = GetPurchasePrice() * 0.10; // 10% of purchase price
        this._profit = GetPurchasePrice() * 0.15; // 15% of purchase price
    }

     // public method to get selling price
    public override double GetPrice()
    {   //calculating selling price, Math.ceiling is just an ibuilt method to round off the p
        return (GetPurchasePrice() + (int)Math.Round(this._refCost) + (int)Math.Round(this._p
    }

    public override void PrintDetails()
    {
        base.PrintDetails();
        Console.WriteLine("Selling price: {0}", this.GetPrice());
    }

}


class Chocolate : Product
{

    private double _profit;

    // Parameterized Constructor
    public Chocolate(string name, double price)
        : base(name, price)
    {
        this._profit = base.GetPurchasePrice() * 0.20; // 20% of purchase price
    }



    // public method to get selling price
    public override double GetPrice()
    {   //calculating selling price, Math.ceiling is just an ibuilt method to round off the p
        return (base.GetPurchasePrice() + (int)Math.Round(this._profit));
    }


    public override void PrintDetails()
    {
        base.PrintDetails();
```

```
        Console.WriteLine("Selling price: {0}", this.GetPrice());
    }

}


class Demo
{

    public static void Main(string[] args)
    {
        // Placing the products in an array
        Product[] products = new Product[4];
        products [0] = new Beverage("Cola", 9);
        products [1] = new Chocolate("Crunch", 15);
        products [2] = new Chocolate("Kit-kat",20);
        products [3] = new Beverage("Fanta",8);

        // name and price of respective  product is displayed
        foreach(Product product in products)
        product.PrintDetails();

    }
}
```



Advantages of the Method Overriding

Method overriding is very useful in OOP. Some of its advantages are stated below:

- The derived classes can give their own specific implementations to inherited methods without modifying the parent class methods.
- The overriding method, in its definition, can extend or add on to the virtual method's definition by calling it in the subclass e.g. the `PrintDetails()` method in the above code.

Things to Keep in Mind

Here are some details we should take care of when thinking about overriding a method:

- Method overriding needs inheritance and there should be at least one derived class.
- Derived class(es) must have the same declaration, i.e., the same name,

parameters, and return type of the overriding methods as those of the virtual methods in the base class.

- The method in the derived class(es) should have a different implementation from each other.
- The method being overridden should be declared `virtual` in the base class and the overriding method should be declared with the keyword `override`.
- The base class or method must not be declared as `sealed`.

Now that we are familiar with the concept of method overriding, let's understand the difference between method overloading and method overriding in the next lesson.