Types of Inheritance

In the previous lessons, you covered the basics of inheritance. In this lesson, you will learn about the various types of inheritance in C#.

Depending whether it is a base class or a derived class, these are the **five** types of inheritance in general:

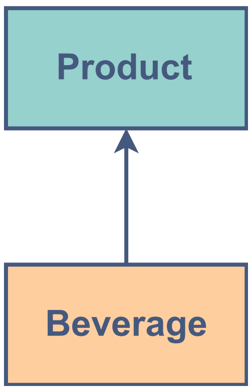
1. **Single**
2. **Multi-level**
3. **Hierarchical**
4. **Multiple**
5. **Hybrid**

Single Inheritance

In single inheritance, there is only a single class extending from another class. We can take the example of the `Product` class (base class) and the `Beverage` class (derived class).

Example

We have already implemented these classes in the previous lessons:



Classification of Products in a Vending Machine

Let’s have a look at the code:

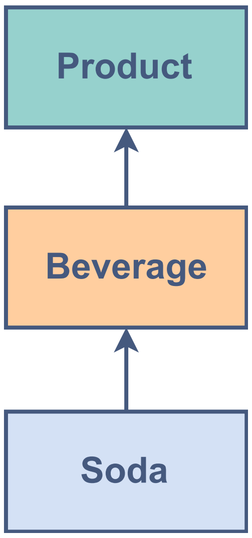
```
// Base Class Product
class Product {

  // Private Fields: Common attributes of all type of products
  private string _name;
  private double _price;
  private string _expiryDate;


  // Parameterized Constructor
  public Product(string name, double price, string expiryDate) {
    this._name = name;
    this._price = price;
    this._expiryDate = expiryDate;

  }

  // public method to print details
  public void PrintDetails() {
    Console.WriteLine("Name: " + this._name);
    Console.WriteLine("Price: " + this._price);
    Console.WriteLine("Expiry Date: " + this._expiryDate);
  }

}

// Derived Class Beverage
class Beverage : Product {

  // Private fields : Fields specific to the derived class
  private double _litres;
  private string _flavor;

  // Parameterized Constructor
  public Beverage(string name, double price, string expiryDate, double litres, string flavor)
    : base(name, price, expiryDate) //calling parent class constructor
  {
      this._litres = litres;
      this._flavor = flavor;
  }


  public void BeverageDetails() {  //details of Beverage
    PrintDetails();         //calling inherited method from parent class
    // Printing fields of this class
    Console.WriteLine("Litres: " + this._litres);
    Console.WriteLine("Flavor: " + this._flavor);
  }

}

class Demo {

  public static void Main(string[] args) {
    Beverage cola = new Beverage("Fanta", 0.5, "12/12/2019", 0.35, "Grape"); //creation of Be
    cola.BeverageDetails(); //calling method to print details
  }

}
```

Multi-level Inheritance

When a class is derived from a class that itself is derived from another class it is called multi-level inheritance. Classes can be extended to any further levels as per the requirement of the model.

Example

Let’s look at an example of multi-level inheritance.



Classification of Products in a Vending Machine

Let’s implement the three classes illustrated above:

- A `Beverage` **IS A** `Product`.
- A `Soda` **IS A** `Beverage`.

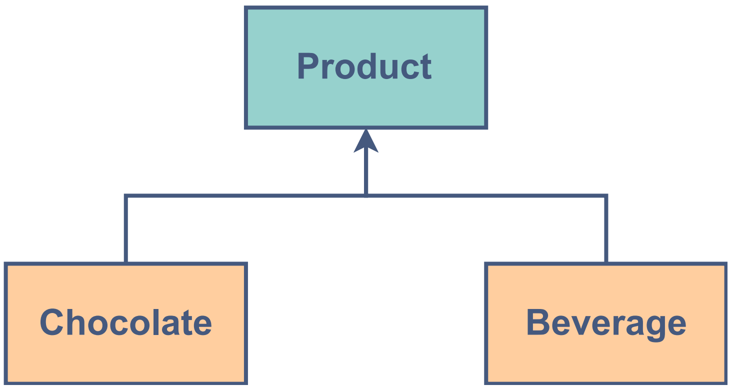
```
// Base Class Product
class Product
{

    // Private Fields: Common attributes of all type of products
    private string _name;
    private double _price;
    private string _expiryDate;


    // Parameterized Constructor
    public Product(string name, double price, string expiryDate)
    {
        this._name = name;
        this._price = price;
        this._expiryDate = expiryDate;

    }

    //getter for name
    public string GetName()
    {
        return this._name;
    }

    // public method to print details
    public void PrintDetails()
    {
        Console.WriteLine("Name: " + this._name);
        Console.WriteLine("Price: " + this._price);
        Console.WriteLine("Expiry Date: " + this._expiryDate);
    }

}

// Derived Class Beverage
class Beverage : Product
{

    // Private fields : Fields specific to the derived class
    private double _litres;
    private string _flavor;

    // Parameterized Constructor
    public Beverage(string name, double price, string expiryDate, double litres, string flavo
        : base(name, price, expiryDate) //calling parent class constructor
    {
        this._litres = litres;
        this._flavor = flavor;
    }

    public void BeverageDetails()
    {  //details of Beverage
        PrintDetails();         //calling inherited method from parent class
                                // Printing fields of this class
        Console.WriteLine("Litres: " + this._litres);
        Console.WriteLine("Flavor: " + this._flavor);
    }


}

class Soda : Beverage
{ // Inheriting Beverage

    // Specifications of a soda
    private bool _isCaffeinated;
    private bool _isDiet;

    // Constructor
    public Soda(string name, double price, string expiryDate, double litres, string flavor, b
    : base(name,  price, expiryDate, litres, flavor)
    {
        this._isCaffeinated = isCaffeinated; // A sode having caffeine will have true
        this._isDiet = isDiet; // A soda having no sugar will have true
    }

    public void SodaDetails()
    {
        BeverageDetails();
        Console.WriteLine("Is the {0} caffeinated? {1}", GetName(), this._isCaffeinated);
        Console.WriteLine("Is the {0} diet? {1}", GetName(), this._isDiet);
    }


}

class Demo
{

    public static void Main(string[] args)
    {
        var cola = new Soda("CocaCola", 0.9, "12/12/2019", 0.35, "Cola", false, true); //crea
        cola.SodaDetails(); //calling method to print details
    }

}
```

Hierarchical Inheritance

When more than one class inherits from the same class, it is referred to as hierarchical inheritance. In hierarchical inheritance, more than one class extends, from the same base class as per the requirement of the design. The common attributes of these child classes are implemented inside the base class.

Example

- A `Beverage` **IS A** `Product`.
- A `Chocolate` **IS A** `Product`.



```
// Base Class Product
class Product
{

    // Private Fields: Common attributes of all type of products
    private string _name;
    private double _price;
    private string _expiryDate;


    // Parameterized Constructor
    public Product(string name, double price, string expiryDate)
    {
        this._name = name;
        this._price = price;
        this._expiryDate = expiryDate;

    }

    //getter for name
    public string GetName()
    {
        return this._name;
    }

    // public method to print details
    public void PrintDetails()
    {
        Console.WriteLine("Name: " + this._name);
        Console.WriteLine("Price: " + this._price);
        Console.WriteLine("Expiry Date: " + this._expiryDate);
    }

}

// Derived Class Beverage
class Beverage : Product
{

    // Private fields : Fields specific to the derived class
    private double _litres;
    private string _flavor;

    // Parameterized Constructor
    public Beverage(string name, double price, string expiryDate, double litres, string flavo
        : base(name, price, expiryDate) //calling parent class constructor
    {
        this._litres = litres;
        this._flavor = flavor;

    }

    public void BeverageDetails()
    {   // Details of Beverage
        PrintDetails();   // Calling inherited method from parent class
        // Printing fields of this class
        Console.WriteLine("Litres: {0}ml", this._litres);
        Console.WriteLine("Flavor: {0} \n", this._flavor);
    }

}

class Chocolate : Product
{ // Inheriting Product

    // Specifications of a chocolate
    private bool _isMilky;
    private double _grams;

    // Constructor
    public Chocolate(string name, double price, string expiryDate, double grams, bool isMilky
        : base(name, price, expiryDate) // Calling parent class constructor
    {
        this._grams = grams; // A chocolate having caffeine will have true
        this._isMilky = isMilky; // The weight of chocolate bar
    }

    public void ChocolateDetails()
    {
        PrintDetails();
        Console.WriteLine("Is the {0} milky? {1}", GetName(), this._isMilky);
        Console.WriteLine("The {0} bar wieghs: {1}g", GetName(), this._grams);
    }


}

class Demo
{

    public static void Main(string[] args)
    {   // Creating and using the Beverage Object
        var cola = new Beverage("RC Cola", 0.9, "12/12/2019", 350, "Cola");
        cola.BeverageDetails();

        // Creating and using the Chocolate Object
        var crunch = new Chocolate("Crunch", 2.3, "3/9/2019", 43, true);
        crunch.ChocolateDetails();

    }

}
```

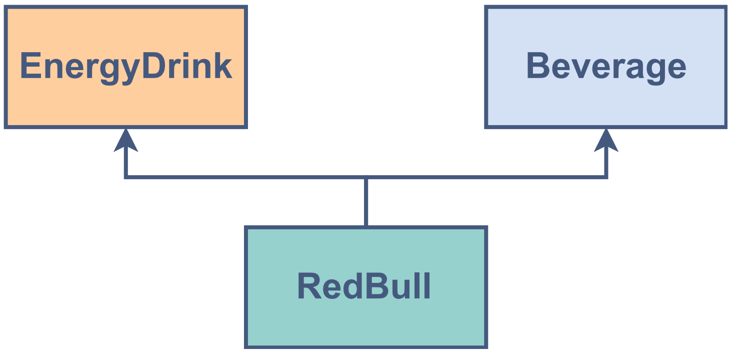
Multiple Inheritance

Multiple Inheritance

When a class is derived from more than one base class, i.e., when a class has more than one immediate parent class, this type of inheritance is called multiple inheritance.

Example

- `RedBull` **IS A** `Beverage`.
- `RedBull` **IS AN** `EnergyDrink` also.

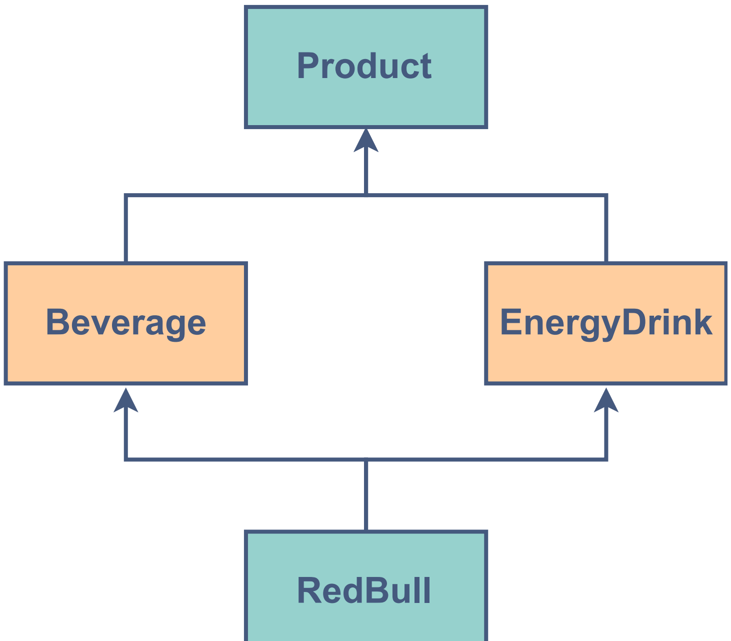


Hybrid Inheritance

A type of inheritance which is a combination of **multiple** and **multi-level** inheritance is called *hybrid inheritance*.

Example

- A `RedBull` is an `EnergyDrink`.
- A `RedBull` is a `Beverage`.
- Both `EnergyDrink` and `Beverage` are products.



> **Note:** In C#, **multiple** and **hybrid** inheritance are applicable using interfaces only.

This lesson was about different types of inheritance. In the next lesson, we will discuss the advantages of inheritance.