Composition

In this lesson, you'll learn how to achieve composition in C#.

`Composition` is the practice of accessing other class objects in your class. In such a scenario, the class which creates the object of the other class is known as the *owner* and is responsible for the lifetime of that *owner* object.

Composition relationships are **Part-of** relationships where the part must be a constituent of the whole object and cannot exist independently of the whole. We can achieve composition by adding the classes together like parts in another class to make a complex unit.

So, what makes the composition unique?

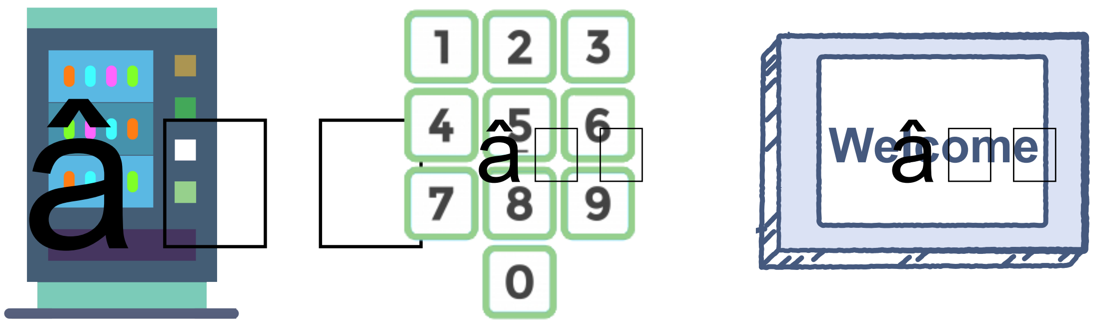
> In **composition**, the lifetime of the owned object depends on the lifetime of the owner.

Example

For the sake of simplicity, let's assume that a `VendingMachine` is composed of a *Display* and *KeyPad*. In this case, a `VendingMachine` owns these objects so it is an *Owner* class and `Display` and `KeyPad` classes are *Owned* classes.

Implementation

Let's look at the implementation of the `VendingMachine` class for better understanding:



Keypad and Display cannot exist independently
if the vending machine ceases to exist

Composition

```
class Display { // Display class

  public Display(){} // Parameter-less constuctor
  public void WelcomeMessage { // Method to welcome a customer
    Console.WriteLine("Welcome to the Vending Machine")
  }
  public void DisplayMethod() // Method to show the machine menu
  {
            Console.Write(@"
.-----.---------------------.
|Press|       Action        |
|-----|---------------------|
|  1  |     Feed Money      |
|  2  |     Select a Product |
|  3  |     Get Change      |
|  4  |     Quit            |
'-----'---------------------'");

}

class KeyPad { // KeyPad class

  public KeyPad(){} // Parameter-less constructor
  public int readKey() // Mehtod to read the user input
  {
      string userInput;
      userInput = Console.ReadLine();
      /* Convert to integer type as the keypad should only have integer inputs */
      int value;
      if (int.TryParse(userInput, out value) && value > 0)
      {
          return value;
      }
      else return -1;
  }

}


class VendingMachine { // Owner class
  // Fields of VendingMachine class
  private Display _machineDisplay;
  private KeyPad _machineKeyPad;
  // Constructor of VendingMachine class

  public VendingMachine(){
    // Creating Owned Objects
    this._machineDisplay = new Display();
    this._machineKeyPad = new KeyPad();
  }
  /* The above Display and KeyPad objects can be used here*/
}
```

> **Note:** We'll see a running example of all this in the Project lesson.

We have created a `VendingMachine` class which contains the objects of `Display` and `KeyPad` classes. The `VendingMachine` class is responsible for the lifetime of the owned objects, i.e., when the vending machine ceases to exist, so does the keypad and display. This makes sense because if the existence of a vending machine ceases, there would be no purpose for a standalone component.

In the next lesson, you can take a short quiz to reinforce your concepts of association, aggregation, and composition.