Implementation of Classes and Objects

Introduction

This reading introduces the fundamentals of creating classes and objects in C#, key components of object-oriented programming (OOP). You'll learn to define a class, instantiate objects, and use object properties and methods.

Understanding Classes

A class in C# serves as a blueprint that defines the structure and behavior of objects. It includes several key components:

- Properties: These are the attributes or data that the object holds, such as a person’s name or age.
- Methods: Functions that define the behaviors or actions that the object can perform.
- Constructors: Special methods used to initialize objects when they are created.

For example, a Person class could have a name and age property, a constructor that sets these values when an object is created, and a method to display the person's information.

Defining a Class in C#

To define a class in C#, you follow a specific syntax:

- Access Modifier - Specifies the visibility of the class (e.g., `public`).
- Class Keyword - The keyword class is used to declare a class.
- Class Name - An identifier for the class.

Here’s an example of a `Person` class in C#:

```
public class Person {
  public string Name { get; set; }
  public int Age { get; set; }
  public Person(string name, int age) {
    Name = name;
    Age = age;
  }
  public void DisplayInfo() {
```

```
    Console.WriteLine($"Name: {Name}, Age: {Age}");
  }
}
```

In this Example

- public is the access modifier that allows the class to be used by any other code.
- Person is the name of the class.
- Inside the class, we define properties (Name and Age), a constructor, and a method (DisplayInfo).

Creating Objects (Instantiation)

To use a class, you need to create an instance of it, which is called an object. This process is known as instantiation. An object is a specific example of a class, similar to a cookie made from a cookie cutter.

To create an object, use the `new` keyword with the class constructor:

```
Person neighbor = new Person("John Doe", 30);
```

In this Line

- `Person` is the type of the object.
- `neighbor` is the name of the object.
- `new Person("John Doe", 30);` creates a new `Person` object using the constructor.

Using Dot Notation

Once an object is instantiated, you can access and modify its properties or call its methods using dot notation. Dot notation involves using a period (.) to reference a member of the object.

Example

- Accessing properties: `neighbor.Age = 31;` changes the `Age` property of the `neighbor` object.
- Calling methods: `neighbor.DisplayInfo();` invokes the `DisplayInfo` method to display the object's information.

Conclusion

Understanding classes and objects is fundamental to developing applications in C#. They enable developers to write modular, reusable code, making managing and extending software projects easier. Mastering these concepts is a crucial step toward building robust object-oriented programs.