Practical Implementation of Inheritance and Polymorphism

Introduction

This reading provides a concise guide to implementing inheritance and polymorphism in C# programming. It focuses on creating base and derived classes and applying method overriding and interfaces.

Inheritance in C#

- Define a base class
    - Use an access modifier (**`public`**, **`private`**, **`protected`**).
    - Use the **`class`** keyword followed by the class name.
    - Define properties to hold data and methods to define actions.

Example:

```
public class Pool
{
  public int chlorineLevel;
  public int waterLevel;
  public Pool(int chlorine, int water)
  {
    chlorineLevel = chlorine;
    waterLevel = water;
  }
  public void PoolInfo()
  {
    Console.WriteLine($"Pool: {chlorineLevel}, {waterLevel}");
  }
}
```

- Create a derived class
    - Declare like a base class but add a colon (:) followed by the base class name.
    - The derived class inherits properties and methods from the base class.

Example:

public class Spa : Pool

```
{
  public int heatLevel;
  public Spa(int chlorine, int water, int heat)
    : base(chlorine, water)
  {
    heatLevel = heat;
  }
  public void SpaInfo()
  {
    Console.WriteLine($"Spa: {chlorineLevel}, {waterLevel}, {heatLevel}");
  }
}
```

Polymorphism in C#

- Method Overriding
    - Use the virtual keyword in the base class to declare methods that can be overridden.
- In the derived class, use the override keyword to provide a specific implementation.

Example:

```
public class Instrument
{
  public virtual void Play()
  {
    Console.WriteLine("Playing an instrument");
  }
}


public class Piano : Instrument
{
  public override void Play()
  {
    Console.WriteLine("The piano is playing");
  }
}
```

- Using Interfaces

- Define an interface with method signatures that derived classes must implement.
- Implement the interface in derived classes.

Example:

public interface IPlayable

```
{
  void Play();
}
```

```
public class Guitar : IPlayable
{
  public void Play()
  {
    Console.WriteLine("The guitar is playing");
  }
}
```

Conclusion

Use inheritance to create reusable base classes and extend their functionality with derived classes. Use polymorphism through method overriding and interfaces to allow flexibility in method implementations across different classes.