

CARRERA PROFESIONAL

# **DESARROLLO DE SISTEMAS DE INFORMACIÓN**

**HERRAMIENTAS DE  
PROGRAMACION C#**

**Tema**

**EJERCICIOS**

## EJERCICIOS RESUELTOS

Aquí tienes una serie de ejercicios sobre **polimorfismo** en C# que abordan conceptos básicos e intermedios, tales como el uso de métodos virtual, override, clases base, clases derivadas, interfaces y la sobrecarga de métodos. Cada ejercicio va acompañado de una descripción del objetivo y el desafío que propone.

### **Ejercicio 1: Sobreescritura de Métodos Virtuales (Polimorfismo en Tiempo de Ejecución)**

**Objetivo:** Crear una jerarquía de clases que sobrescriba un método de la clase base y demuestre polimorfismo en tiempo de ejecución.

#### **Descripción:**

1. Crea una clase base llamada Forma con un método virtual llamado Dibujar().
2. Crea dos clases derivadas: Circulo y Rectangulo, que sobrescriban el método Dibujar() para que impriman el nombre de la forma que están dibujando.
3. En el programa principal, crea una lista de Forma y agrega instancias de Circulo y Rectangulo.
4. Llama al método Dibujar() sobre cada elemento de la lista y muestra el resultado en la consola.

**Solución esperada:**

```
class Forma
{
    public virtual void Dibujar()
    {
        Console.WriteLine("Dibujar una forma genérica.");
    }
}

class Circulo : Forma
{
    public override void Dibujar()
    {
        Console.WriteLine("Dibujar un círculo.");
    }
}

class Rectangulo : Forma
{
    public override void Dibujar()
    {
        Console.WriteLine("Dibujar un rectángulo.");
    }
}

class Programa
{
    static void Main()
    {
        List<Forma> formas = new List<Forma>
        {
            new Circulo(),
            new Rectangulo()
        };

        foreach (var forma in formas)
        {
            forma.Dibujar();
        }
    }
}
```

**Resultado:**

Dibujar un círculo.

Dibujar un rectángulo.

**Ejercicio 2: Sobrecarga de Métodos (Polimorfismo en Tiempo de Compilación)**

**Objetivo:** Implementar métodos sobrecargados en una clase que realicen cálculos aritméticos con diferentes tipos de parámetros.

**Descripción:**

1. Crea una clase llamada Calculadora con tres métodos Sumar sobrecargados:
  - Uno que acepte dos enteros.
  - Uno que acepte tres enteros.
  - Uno que acepte dos valores de tipo double.
2. En el programa principal, invoca cada uno de los métodos Sumar con diferentes tipos de parámetros y muestra los resultados.

**Solución esperada:**

```
class Calculadora
{
    public int Sumar(int a, int b)
    {
        return a + b;
    }

    public int Sumar(int a, int b, int c)
    {
        return a + b + c;
    }

    public double Sumar(double a, double b)
    {
        return a + b;
    }
}

class Programa
{
    static void Main()
    {
        Calculadora calc = new Calculadora();
        Console.WriteLine(calc.Sumar(3, 4)); // 7
        Console.WriteLine(calc.Sumar(3, 4, 5)); // 12
        Console.WriteLine(calc.Sumar(2.5, 3.5)); // 6.0
    }
}
```

### Ejercicio 3: Polimorfismo y Herencia

**Objetivo:** Crear una jerarquía de clases donde las clases derivadas sobrescriban métodos de la clase base y demuestren el polimorfismo en una operación bancaria.

#### Descripción:

1. Crea una clase base llamada CuentaBancaria con un método virtual CalcularInteres().
2. Crea dos clases derivadas: CuentaCorriente y CuentaAhorros.
  - En CuentaCorriente, sobrescribe el método CalcularInteres() para devolver un valor fijo de interés (ejemplo: 1%).
  - En CuentaAhorros, sobrescribe el método para devolver un interés más alto (ejemplo: 3%).
3. En el programa principal, crea instancias de ambas clases y llama al método CalcularInteres().

#### Solución esperada:

```
1   using System;
2
3   4 referencias
4   class CuentaBancaria
5   {
6       4 referencias
7       public virtual double CalcularInteres()
8       {
9           return 0.0;
10      }
11
12      1 referencia
13      class CuentaCorriente : CuentaBancaria
14      {
15          3 referencias
16          public override double CalcularInteres()
17          {
18              return 1.0;
19          }
20
21          1 referencia
22          class CuentaAhorros : CuentaBancaria
23          {
24              3 referencias
25              public override double CalcularInteres()
26              {
27                  return 3.0;
28              }
29
30              0 referencias
31              class Programa
32              {
33                  0 referencias
34                  static void Main()
35                  {
36                      CuentaBancaria cuenta1 = new CuentaCorriente();
37                      CuentaBancaria cuenta2 = new CuentaAhorros();
38
39                      Console.WriteLine($"Interés en cuenta corriente: {cuenta1.CalcularInteres()}%");
40                      Console.WriteLine($"Interés en cuenta de ahorros: {cuenta2.CalcularInteres()}%");
41                  }
42              }
43          }
44      }
45  }
```

**Resultado:**

Interés en cuenta corriente: 1.0%  
Interés en cuenta de ahorros: 3.0%

