

Programación orientada a objetos en C#

Vamos a crear tipos nuevos que representan una cuenta bancaria. Normalmente los desarrolladores definen cada clase en un módulo o archivo de texto diferente. De esta forma, la tarea de administración resulta más sencilla a medida que aumenta el tamaño del programa.

Esta solución contendrá la definición de una cuenta bancaria. La programación orientada a objetos organiza el código mediante la creación de tipos en forma de clases. Estas clases contienen el código que representa una entidad específica. La clase *CuentaBancaria* representa una cuenta bancaria. El código implementa operaciones específicas a través de métodos y propiedades. La cuenta bancaria admite el siguiente comportamiento:

- ✓ Tiene un número de diez dígitos que identifica la cuenta bancaria de forma única.
- ✓ Tiene una cadena que almacena el nombre o los nombres de los propietarios.
- ✓ Se puede consultar el saldo.
- ✓ Acepta depósitos.
- ✓ Acepta reintegros.
- ✓ El saldo inicial debe ser positivo.
- ✓ Los reintegros no pueden dar como resultado en un saldo negativo.

Realiza y documenta las siguientes actividades:

1. Inicia el entorno de desarrollo *Microsoft Visual Studio Community*.
2. Crea un nuevo proyecto: *Archivo + Nuevo + Proyecto...*
3. Selecciona el tipo de proyecto de consola, *Aplicación de consola (.NET Core)*, escribe el nombre *CuentaBancaria*, selecciona su ubicación *C:\VSProjects*, escribe el nombre de la solución si deseas cambiarlo, haz clic en *Crear directorio para la solución* y haz clic en *Crear*.
4. Se crea automáticamente la solución.
5. Visual Studio crea automáticamente la solución y abre el nuevo proyecto, que incluye código predeterminado de *"Hola mundo"*.
6. Elimina todo el código del módulo *Program.cs* y pega el siguiente código:

```
using System;

namespace CuentaBancaria {

    class Program {

        static void Main(string[] args) {
            var account = new BankAccount("Pedro Andrés Mancebo", 1000);

            Console.WriteLine($"La cuenta {account.Number} creada por " +
                              $"{account.Owner} con un saldo inicial de " +
                              $"{account.Balance}.");

            account.MakeWithdrawal(500, DateTime.Now, "Pago del alquiler");
            Console.WriteLine(account.Balance);
            account.MakeDeposit(100, DateTime.Now, "Ingreso nómina");
            Console.WriteLine(account.Balance);

            // Test that the initial balances must be positive.
            try {
                var invalidAccount = new BankAccount("Incorrecto", -55);
            }
        }
    }
}
```

```

        catch (ArgumentOutOfRangeException e) {
            Console.WriteLine("Error al crear una cuenta con saldo
                               negativo");
            Console.WriteLine(e.ToString());
        }

        // Test for a negative balance:
        try {
            account.MakeWithdrawal(750, DateTime.Now, "Intento de
                               descubierto");
        }
        catch (InvalidOperationException e) {
            Console.WriteLine("Error al intentar un descubierto");
            Console.WriteLine(e.ToString());
        }

        Console.WriteLine(account.GetAccountHistory());

        Console.WriteLine("\nPulsa una tecla para finalizar");
        Console.ReadLine();
    }
}

```

7. Documenta cada una de las líneas de código C#.
8. En Explorador de soluciones, selecciona el proyecto *CuentaBancaria*, haz clic con el botón derecho y selecciona *Agregar > Nueva carpeta*, asigna como nombre de la carpeta *clases*.
9. En Explorador de soluciones, selecciona la carpeta *clases* del proyecto *CuentaBancaria*, haz clic con el botón derecho y selecciona *Agregar > Clase*, asigna como nombre de la clase *Transaction.cs* y haz clic en el botón *Agregar*.
10. En el archivo *Transaction.cs*, quita todo el código de plantilla y reemplázalo por el código siguiente:

```

using System;

namespace CuentaBancaria {

    public class Transaction {
        public decimal Amount { get; }
        public DateTime Date { get; }
        public string Notes { get; }

        public Transaction(decimal amount, DateTime date, string note) {
            this.Amount = amount;
            this.Date = date;
            this.Notes = note;
        }
    }
}

```

11. Documenta cada una de las líneas de código C#.
12. En Explorador de soluciones, selecciona la carpeta *clases* del proyecto *CuentaBancaria*, haz clic con el botón derecho y selecciona *Agregar > Clase*, asigna como nombre de la clase *BankAccount.cs* y haz clic en el botón *Agregar*.
13. En el archivo *BankAccount.cs*, quita todo el código de plantilla y reemplázalo por el código siguiente:

```

using System;
using System.Collections.Generic;

namespace CuentaBancaria {

```

```

public class BankAccount {

    public string Number { get; }
    public string Owner { get; set; }
    public decimal Balance {
        get {
            decimal balance = 0;
            foreach (var item in allTransactions) {
                balance += item.Amount;
            }
            return balance;
        }
    }

    private static int accountNumberSeed = 1234567890;

    private List<Transaction> allTransactions = new List<Transaction>();

    public BankAccount(string name, decimal initialBalance) {
        this.Number = accountNumberSeed.ToString();
        accountNumberSeed++;
        this.Owner = name;
        MakeDeposit(initialBalance, DateTime.Now, "Saldo inicial");
    }

    public void MakeDeposit(decimal amount, DateTime date, string note) {
        if (amount <= 0) {
            throw new ArgumentOutOfRangeException(nameof(amount),
                "La cantidad del ingreso debe ser positiva");
        }
        var deposit = new Transaction(amount, date, note);
        allTransactions.Add(deposit);
    }

    public void MakeWithdrawal(decimal amount, DateTime date, string note){
        if (amount <= 0) {
            throw new ArgumentOutOfRangeException(nameof(amount),
                "La cantidad debe ser positiva");
        }
        if (Balance - amount < 0) {
            throw new InvalidOperationException("No existe suficiente
                saldo para el reintegro");
        }
        var withdrawal = new Transaction(-amount, date, note);
        allTransactions.Add(withdrawal);
    }

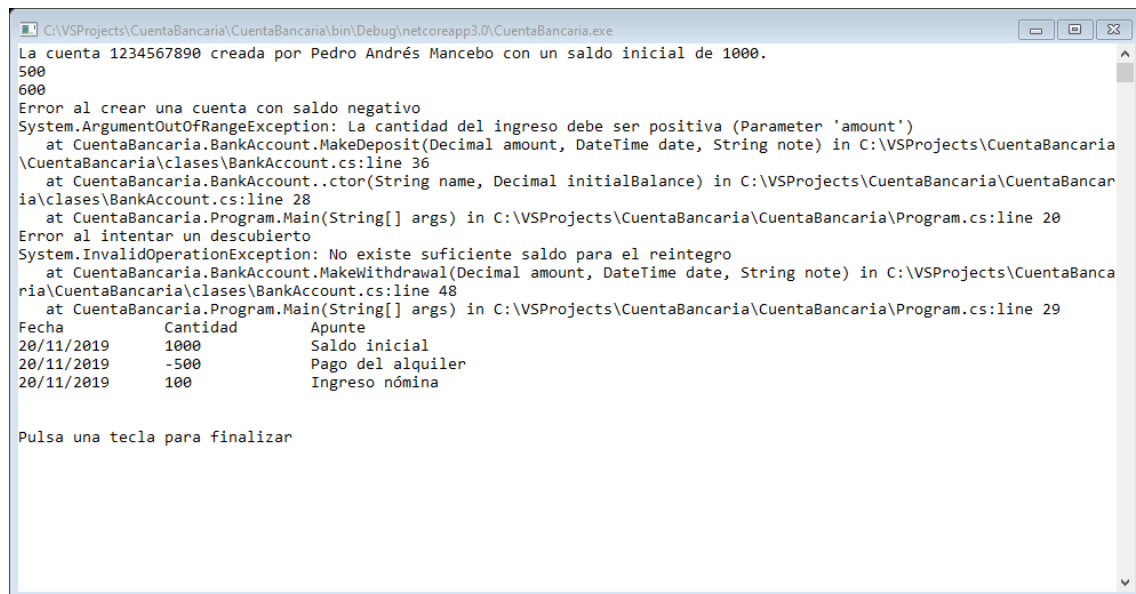
    public string GetAccountHistory() {
        var report = new System.Text.StringBuilder();

        report.AppendLine("Fecha\t\tCantidad\tApunte");
        foreach (var item in allTransactions) {
            report.AppendLine($"{item.Date.ToShortDateString()}
                \t{item.Amount}\t\t{item.Notes}");
        }
        return report.ToString();
    }
}

```

14. Documenta cada una de las líneas de código C#.

15. Compila la solución para comprobar que todo funciona correctamente: *Compilar > Compilar solución*. Observa los mensajes en la ventana de salida y el icono inferior en la barra de estado.
16. Ejecuta o inicia la depuración de la solución en la consola: *Depurar > Iniciar la depuración*.
17. Espera a que se compile y se despliegue el proyecto en la consola.



```
C:\VSProjects\CuentaBancaria\CuentaBancaria\bin\Debug\netcoreapp3.0\CuentaBancaria.exe
La cuenta 1234567890 creada por Pedro Andrés Mancebo con un saldo inicial de 1000.
500
600
Error al crear una cuenta con saldo negativo
System.ArgumentOutOfRangeException: La cantidad del ingreso debe ser positiva (Parameter 'amount')
   at CuentaBancaria.BankAccount.MakeDeposit(Decimal amount, DateTime date, String note) in C:\VSProjects\CuentaBancaria\CuentaBancaria\clases\BankAccount.cs:line 36
   at CuentaBancaria.BankAccount..ctor(String name, Decimal initialBalance) in C:\VSProjects\CuentaBancaria\CuentaBancaria\clases\BankAccount.cs:line 28
   at CuentaBancaria.Program.Main(String[] args) in C:\VSProjects\CuentaBancaria\CuentaBancaria\Program.cs:line 20
Error al intentar un descubierto
System.InvalidOperationException: No existe suficiente saldo para el reintegro
   at CuentaBancaria.BankAccount.MakeWithdrawal(Decimal amount, DateTime date, String note) in C:\VSProjects\CuentaBancaria\CuentaBancaria\clases\BankAccount.cs:line 48
   at CuentaBancaria.Program.Main(String[] args) in C:\VSProjects\CuentaBancaria\CuentaBancaria\Program.cs:line 29
Fecha      Cantidad      Apunte
20/11/2019    1000      Saldo inicial
20/11/2019    -500      Pago del alquiler
20/11/2019     100      Ingreso nómina

Pulsa una tecla para finalizar
```

18. Explica el resultado de su ejecución.
19. Finaliza en entorno de desarrollo *Microsoft Visual Studio*.