

Programación Orientada a Objetos 4

Ejemplo Control Bancario (2)

Contenido

Creando clases derivadas de CuentaBancaria

Crear clase CuentaDeAhorros

Crear clase CuentaDeCheques

Creación de una colección de Cuentas



Creando clases derivadas de CuentaBancaria (1)

- En este ejercicio, creará dos subclases de la clase CuentaBancaria:
 - CuentaDeAhorros y
 - CuentaDeCheques.



Creando clases derivadas de CuentaBancaria (2)

- Estas cuentas tendrán las siguientes reglas:
 - Una cuenta de ahorros gana intereses.
 - El banco permite a los clientes guardar dinero en una cuenta de ahorro y mensualmente la cuenta de ahorro le otorgará intereses de acuerdo a la siguiente fórmula:
 - $\text{saldo} = \text{saldo} + (\text{tasaDeInteres} * \text{saldo});$
 - Una cuenta de cheques le permite al cliente hacer cualquier número de depósitos y retiros.
 - Para proteger a sus clientes, el banco le da una *protección de sobregiro*. Esta protección le permite al saldo del usuario hacerse negativo, pero no menos de la protección de sobregiro otorgada



Crear clase CuentaDeAhorros

- Modifique la clase CuentaBancaria.
 - El atributo y el constructor debe ser cambiados para que sean *protected* (en vez de *public* o *private*)
- Declare la clase **CuentaDeAhorros** para que herede de CuentaBancaria:
 - Agregue el atributo **tasaDeInteres** a la clase CuentaDeAhorros
 - Agregue un constructor público que reciba dos parámetros: el saldo inicial y la tasa de interés. Pase el primero al constructor de la clase padre y el segundo asígnelo al atributo correspondiente:



**CuentaBancaria.cs**

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Programa01
{
    public class CuentaBancaria
    {
        protected double saldo; // podra ser accedido por las clases derivadas
        public CuentaBancaria2(double saldo)
        {
            this.saldo = saldo;
        }
        public double ObtenSaldo()
        {
            return saldo;
        }
        public void Deposita(double cantidad)
        {
            saldo += cantidad;
        }
        // metodo virtual que podra ser sobre escrito
        public virtual bool Retira(double cantidad)
        {
            if (saldo >= cantidad)
            {
                saldo -= cantidad;
                return true;
            }
            else return false;
        }
    }
}
```



Outline



CuentaDeAhorros.cs

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Programa01
{
    class CuentaDeAhorros : CuentaBancaria
    {
        private double tasaDeInteres;

        public CuentaDeAhorros(double saldo, double tasa)
            : base(saldo)
        {
            tasaDeInteres = tasa;
        }
    }
}
```

Crear clase CuentaDeCheques

- Declare la clase **CuentaDeCheques** para que herede de CuentaBancaria.
 - Agregue el atributo **proteccionSobregiro** a la clase CuentaDeCheques
 - Agregue un constructor público que reciba dos parámetros: el saldo inicial y la protección de sobregiro. Pase el primero al constructor de la clase padre y el segundo asígnelo al atributo correspondiente:





```
using System;
using System.Collections.Generic;
using System.Text;
namespace Programa01
{
    class CuentaDeCheques : CuentaBancaria
    {
        private double proteccionSobregiro;
        public CuentaDeCheques(double saldo, double proteccion)
            : base(saldo)
        {
            proteccionSobregiro = proteccion;
        }
        public override bool Retira(double cantidad)
        {
            bool resultado = true;
            if (cantidad > saldo) {
                double proteccionRequerida = cantidad - saldo;
                if (proteccionSobregiro < proteccionRequerida)
                    return false;
                else {
                    saldo = 0.0;
                    proteccionSobregiro -= proteccionRequerida;
                }
            }
            else {
                saldo -= cantidad;
                return true;
            }
            return resultado;
        }
    }
}
```




PruebaBanco.cs (1)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Program1
{
    class PruebaBanco
    {
        public static void Main(string[] args)
        {
            Banco mibanco = new Banco();
            Cliente cliente;
            CuentaBancaria cuenta;

            // Crea mibanco, clientes y sus cuentas
            Console.WriteLine("Creando al cliente Juan Camaney.");
            mibanco.AgregaCliente("Juan", "Camaney");
            cliente = mibanco.ObtenCliente(0);
            Console.WriteLine("Crean su cuenta de ahorros con un saldo de 500.00 y una tasa de interes del 3%");
            cliente.EstableceCuenta(new CuentaDeAhorros(500, 0.03));
            Console.WriteLine("Creando al cliente Tony Soprano.");
            mibanco.AgregaCliente("Tony", "Soprano");
            cliente = mibanco.ObtenCliente(1);
            Console.WriteLine("Creando su cuenta de cheques con un saldo de 500.00 sin proteccion de sobregiro");
            cliente.EstableceCuenta(new CuentaDeCheques(500, 0));
            Console.WriteLine("Creando al cliente Jessica Alba.");
            mibanco.AgregaCliente("Jessica", "Alba");
            cliente = mibanco.ObtenCliente(2);
            Console.WriteLine("Creando su cuenta de cheques con un saldo de 500.00 con proteccion de sobregiro de 500.0");
            cliente.EstableceCuenta(new CuentaDeCheques(500.0, 500.0));
            Console.WriteLine("Creando al cliente Jack Bauer.");
            mibanco.AgregaCliente("Jack", "Bauer");
            cliente = mibanco.ObtenCliente(3);
            Console.WriteLine("Jack comparte su cuenta de cheques con su esposa Jessica.");
            cliente.EstableceCuenta(mibanco.ObtenCliente(2).ObtenCuenta());
            Console.WriteLine();

            // Se demuestra el comportamiento de las diversos tipos de cuenta
            // Prueba una cuenta de ahorros
            Console.WriteLine("Recuperando al cliente Juan Camaney y su cuenta de ahorros.");
            cliente = mibanco.ObtenCliente(0);
            cuenta = cliente.ObtenCuenta();

            // Realiza algunas transacciones
            Console.WriteLine("Retira 150.00: {0}", cuenta.Retira(150.0));
            Console.WriteLine("Deposita 22.50");
            cuenta.Deposita(22.5);
            Console.WriteLine("Retira 47.62 : {0} ", cuenta.Retira(47.62));
            Console.WriteLine("Retira 400.00: {0} ", cuenta.Retira(400.0));

            // Imprime su saldo final
            Console.WriteLine("Cliente [{0} {1}], tiene un saldo de {2} ", cliente.ObtenApellidoPat(), cliente.ObtenNombre(), cuenta.ObtenSaldo());
            Console.WriteLine();
        }
    }
}
```



PruebaBanco.cs (2)

```
// Prueba una cuenta de cheques sin proteccion de sobregiro
Console.WriteLine("Recuperando al cliente Tony Soprano y su cuenta de cheques sin proteccion de sobregiro.");
cliente = mibanco.ObtenCliente(1);
cuenta = cliente.ObtenCuenta();
// Realizamos algunas transacciones
Console.WriteLine("Retira 150.00: {0}", cuenta.Retira(150.0));
Console.WriteLine("Deposita 22.50");
cuenta.Deposita(22.5);
Console.WriteLine("Retira 47.62: {0}", cuenta.Retira(47.62));
Console.WriteLine("Retira 400.00: {0}", cuenta.Retira(400.0));

// Imprime su saldo final
Console.WriteLine("Cliente [{0} {1}], tiene un saldo de {2} ", cliente.ObtenApellidoPat(), cliente.ObtenNombre(),
cuenta.ObtenSaldo());
// Prueba una cuenta de cheques con proteccion de sobregiro
Console.WriteLine("Recuperando al cliente Jessica Alba y su cuenta de cheques con proteccion de sobregiro.");
cliente = mibanco.ObtenCliente(2);
cuenta = cliente.ObtenCuenta();

// realiza algunas transacciones
Console.WriteLine("Retira 150.00: {0}", cuenta.Retira(150.0));
Console.WriteLine("Deposita 22.50");
cuenta.Deposita(22.5);
Console.WriteLine("Retira 47.62: {0}", cuenta.Retira(47.62));
Console.WriteLine("Retira 400.00: {0}", cuenta.Retira(400.0));

// Imprime su saldo final;
Console.WriteLine("Cliente [{0} {1}], tiene un saldo de {2} ", cliente.ObtenApellidoPat(), cliente.ObtenNombre(),
cuenta.ObtenSaldo());
Console.WriteLine();
// Prueba una cuenta de cheques con proteccion de sobregiro
Console.WriteLine("Recuperando al cliente Jack Bauer y su cuenta de cheques compartida con Jessica.");
cliente = mibanco.ObtenCliente(3);
cuenta = cliente.ObtenCuenta();
// Realiza algunas transacciones
Console.WriteLine("Deposita 150.00");
cuenta.Deposita(150.0);
Console.WriteLine("Retira 750.00: {0}", cuenta.Retira(750.0));

// Imprime su saldo final
Console.WriteLine("Cliente [{0} {1}], tiene un saldo de {2} "
, cliente.ObtenApellidoPat(), cliente.ObtenNombre(), cuenta.ObtenSaldo());
Console.ReadLine();
    }
}
```



Salida PruebaBanco

```
Creando al cliente Juan Camaney.  
Crean su cuenta de ahorros con un saldo de 500.00 y una tasa de interes del 3%  
Creando al cliente Tony Soprano.  
Creando su cuenta de cheques con un saldo de 500.00 sin proteccion de sobregiro  
Creando al cliente Jessica Alba.  
Creando su cuenta de cheques con un saldo de 500.00 con proteccion de sobregiro de 500.0  
Creando al cliente Jack Bauer.  
Jack comparte su cuenta de cheques con su esposa Jessica.
```

```
Recuperando al cliente Juan Camaney y su cuenta de ahorros.  
Retira 150.00: True  
Deposita 22.50  
Retira 47.62 : True  
Retira 400.00: False  
Cliente [Camaney Juan], tiene un saldo de 324.88
```

```
Recuperando al cliente Tony Soprano y su cuenta de cheques sin proteccion de sobregiro.  
Retira 150.00: True  
Deposita 22.50  
Retira 47.62: True  
Retira 400.00: False  
Cliente [Soprano Tony], tiene un saldo de 324.88
```

```
Recuperando al cliente Jessica Alba y su cuenta de cheques con proteccion de sobregiro.  
Retira 150.00: True  
Deposita 22.50  
Retira 47.62: True  
Retira 400.00: True  
Cliente [Alba Jessica], tiene un saldo de 0
```

```
Recuperando al cliente Jack Bauer y su cuenta de cheques compartida con Jessica.  
Deposita 150.00  
Retira 750.00: False  
Cliente [Bauer Jack], tiene un saldo de 150
```

Creación de una colección de Cuentas (1)

- Creará un arreglo heterogéneo para representar la agregación de clientes a cuentas, es decir, un cliente dado puede tener varias cuentas de diferentes tipos.
 - Modificar la clase Cliente para dar soporte a una colección heterogénea de objetos de clase CuentaBancaria.
 - Modifique la clase Cliente de tal manera que se puedan manejar varias cuentas de diferentes tipos.
 - Debe incluir los métodos: **agregaCuenta**, **obtenCuenta** y **obtenNumDeCuentas**.



Creación de una colección de Cuentas (2)

- Agregue dos atributos a la clase Cliente:
 - *cuentas* que es un arreglo de objetos CuentaBancaria substituye al anterior atributo *cuenta*, que debería ser eliminado.
 - *numeroDeCuentas* que es entero.
- Modifique el constructor para inicializar el arreglo *cuentas*.
- Agregue el método **agregaCuenta** el cual recibe como parámetro la cuenta a agregar y la almacena en el arreglo cuentas. Este método reemplaza al método estableceCuenta el cual debe ser eliminado.



Creación de una colección de Cuentas (3)

- Agregue el método **obtenNumDeCuentas** regresa el atributo numeroDeCuentas
- Agregue el método **obtenCuenta** la cual regresa la cuenta asociada al índice recibido como parámetro. Este método remplaza al anterior método **obtenCuenta** el cual debe ser eliminado.



**Cliente.cs**

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Programa02
{
    public class Cliente
    {
        private string nombre;
        private string apellidopaterno;
        private CuentaBancaria[] cuentas;
        private int numeroDeCuentas;

        public Cliente(string nom, string apat)
        {
            this.nombre = nom;
            this.apellidopaterno = apat;
            cuentas = new CuentaBancaria[10];
            numeroDeCuentas = 0;
        }
        public string ObtenNombre()
        {
            return nombre;
        }
        public string ObtenApellidoPat()
        {
            return apellidopaterno;
        }
        public CuentaBancaria ObtenCuenta(int indice)
        {
            return cuentas[indice];
        }
        public void AgregaCuenta(CuentaBancaria cta)
        {
            int i = numeroDeCuentas++;
            cuentas[i] = cta;
        }
        public int ObtenNumDeCuentas()
        {
            return numeroDeCuentas;
        }
    }
}
```

Creación de una colección de Cuentas (4)

- Finalmente crearemos dos clases
 - ReporteCliente
 - Que genera un reporte de los Clientes y sus cuentas existentes
 - Se usa el operador **Is** para verificar el tipo de cuenta del cliente
 - PruebaReporteCliente
 - Que llena la clase Banco de clientes
 - A cada cliente le asigna varias cuentas





ReporteCliente.cs

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Programa02
{
    class ReporteCliente
    {
        private Banco mibanco;
        public ReporteCliente()
        {
        }
        public Banco obtenBanco()
        {
            return mibanco;
        }
        public void EstableceBanco(Banco mibanco)
        {
            this.mibanco = mibanco;
        }
        public void GeneraReporte()
        {
            // Imprime encabezado
            Console.WriteLine("\t\t\tREPORTE DE CLIENTES");
            Console.WriteLine("\t\t\t=====");
            // Para cada cliente...
            for (int i = 0; i < mibanco.ObtenNumeroDeClientes(); i++)
            {
                Cliente cliente = mibanco.ObtenCliente(i);
                // Imprime el nombre del cliente
                Console.WriteLine();
                Console.WriteLine("Cliente: "
                    + cliente.ObtenApellidoPat() + ", "
                    + cliente.ObtenNombre());
                // Para cada cuenta de este cliente...
                for (int j = 0; j < cliente.ObtenNumDeCuentas(); j++)
                {
                    CuentaBancaria cuenta = cliente.ObtenCuenta(j);
                    string tipo_cuenta = "";
                    if (cuenta is CuentaDeAhorros)
                        tipo_cuenta = "Cuenta de Ahorros";
                    else if (cuenta is CuentaDeCheques)
                        tipo_cuenta = "Cuenta de Cheques";
                    else
                        tipo_cuenta = "Cuenta Desconocida";
                    Console.WriteLine("{0} , saldo de {1}", tipo_cuenta, cuenta.ObtenSaldo());
                }
            }
        }
    }
}
```



PruebaReporteCliente

```
using System;
using System.Collections.Generic;
using System.Text;
namespace Programa02
{
    class PruebaReporteCliente
    {
        static void Main(string[] args)
        {
            Banco mibanco = new Banco();
            InicializaClientes(mibanco);
            // ejecuta el reporte de los clientes
            ReporteCliente reporte = new ReporteCliente();
            reporte.EstableceBanco(mibanco);
            reporte.GeneraReporte();
            Console.ReadLine();
        }
        private static void InicializaClientes(Banco mibanco)
        {
            Cliente cliente;
            // Crea varios clientes y sus cuentas
            mibanco.AgregaCliente("Juan", "Camaney");
            cliente = mibanco.ObtenCliente(0);
            cliente.AgregaCuenta(new CuentaDeAhorros(500.00, 0.05));
            cliente.AgregaCuenta(new CuentaDeCheques(200.00, 400.00));
            mibanco.AgregaCliente("Tony", "Soprano");
            cliente = mibanco.ObtenCliente(1);
            cliente.AgregaCuenta(new CuentaDeCheques(200.00,10));
            mibanco.AgregaCliente("Jessica", "Alba");
            cliente = mibanco.ObtenCliente(2);
            cliente.AgregaCuenta(new CuentaDeAhorros(1500.00, 0.05));
            cliente.AgregaCuenta(new CuentaDeCheques(200.00,10));
            mibanco.AgregaCliente("Jack", "Bauer");
            cliente = mibanco.ObtenCliente(3);
            // Jack y Jessica comparten la misma cuenta de cheques
            cliente.AgregaCuenta(mibanco.ObtenCliente(2).ObtenCuenta(1));
            cliente.AgregaCuenta(new CuentaDeAhorros(150.00, 0.05));
        }
    }
}
```

**Salida****PruebaReporteCliente****REPORTE DE CLIENTES**

=====

Cliente: Camaney, Juan

Cuenta de Ahorros , saldo de 500

Cuenta de Cheques , saldo de 200

Cliente: Soprano, Tony

Cuenta de Cheques , saldo de 200

Cliente: Alba, Jessica

Cuenta de Ahorros , saldo de 1500

Cuenta de Cheques , saldo de 200

Cliente: Bauer, Jack

Cuenta de Cheques , saldo de 200

Cuenta de Ahorros , saldo de 150