Polimorfismo, Escapsulación y Propiedades

NOTA

A partir de este bloque de ejercicios, en todos ellos, deberemos usar los conceptos aprendidos en temas anteriores que nos den como resultado una mejora en la modularidad y la claridad del código:

- Utilización de métodos con nombres explicativos, siguiendo los convenios para los identificadores.
- Control de excepciones,
- Uso de los modificadores de acceso de forma correcta.
- Uso de enumeraciones, estructuras, expresiones regulares, etc.

Ejercicio 1

Para dar nuestros primeros pasos con el polimorfismo de datos, vamos a recordar el cálculo de factorial y de número primo. Vamos a necesitar una clase padre **ElNoCalculador** con los métodos virtuales Factorial() y Primo() que devolverán 0 y un booleano respectivamente.

Además, crearemos una clase hija **ElCalculador** que sí implementa estos métodos de forma correcta. Se deberá controlar las excepciones necesarias (desbordamiento, etc). Para probar el funcionamiento del polimorfismo nos crearemos un objeto de la manera:

```
ElNoCalculador obj=new ElCalculador(num);
```

Y llamaremos a los métodos correspondientes.

¿A qué métodos se llama, a los de la clase padre o a los de la clase hija?

Crea en la clase hija un método MostrarResultado y llámalo con el obj.

¿Qué ocurre?

Y si comentas el método Primo de la hija, ¿que ocurre?

Ejercicio 2

A partir de la clases Humano, Guerrero y Mago que implementaste en ejercicios anteriores. Mejoralas, utilizando **propiedades** que sustituyan a los métodos accesores Get y Set que tenían.

Será importante que solo crees las propiedades que sean necesarias y les des el nivel de acceso oportuno. Por ejemplo, para el campo inteligencia es absurdo que se permita modificar directamente a través de la propiedad, tiene más sentido que se modifique después de ocurrir algún proceso.

En cuanto al nivel de acceso de nombre, edad, etc. es más conveniente que se puedan modificar a través de la hija o de un constructor, pero que sí se puedan mostrar desde cualquier ámbito.

Con las clases Guerrero y Mago haz la misma mejora añadiendo los métodos que creas necesarios.

En el programa principal, crea la clase Humano y en las clases hijas, como mínimo dos métodos que realicen algo productivo en el código, con los que pruebes el polimorfismo de datos.

Ejercicio 3

Vamos a ampliar el ejercicio de la cuenta corriente que hicimos en excepciones. Modificando ligeramente su definición inicial, para modelar los tipos de cuenta de un banco algo usurero.

De tal manera que cuenta va a simbolizar un tipo de cuenta genérico del que sólo podremos ingresar, retirar fondos y tener unos datos de usuario.

Además, añadiremos una propiedad autodefinida Saldo donde el get será público para poder verlo y el set protegido.

Además, tendremos varias especificaciones nuevas para cuenta:

Cuenta de ahorros

- Se especializa respecto a la genérica mediante la aplicación de intereses al saldo actual cuando se le indique.
- Por ejemplo, si una cuenta tiene un saldo de 1000€ y la tasa de interés es del 2%, después del pago de intereses el nuevo saldo será de 1020€.
- Este tipo de cuentas no permitirá crédito y los intereses se sumarán al finalizar el mes sobre el saldo en ese momento.

```
saldo = saldo + (saldo * interes_tpu);
```

- Redefiniremos el método ToString para que indique el tipo de cuenta antes de llamar al de la clase base para que muestre sus datos básicos.

Cuenta de depósito

- También aplica intereses al saldo actual, por lo cual hereda de cuenta de ahorros. Pero si el titular realiza un reintegro de parte del capital antes del **vencimiento** del plazo fijo, el banco deducirá un porcentaje sobre el reintegro.
- Por ejemplo, si el titular retira 1000€ antes del vencimiento del plazo y hay un recargo del 5% sobre el reintegro, el saldo disminuirá en 1000€ pero el titular sólo recibirá 950€.
- Si el plazo de la cuenta venció, no se cobrará recargo por el reintegro.
- Este tipo de cuenta no permite crédito y los intereses se abonarán al finalizar el mes sobre el saldo en ese momento.

```
this.Saldo = this.Saldo - cantidad;
// Si el depósito aún no venció.
cantidad = cantidad - (cantidad * recargo_tpu);
```

- Redefiniremos el método ToString para que indique el tipo de cuenta antes de llamar al de la clase base para que muestre sus datos básicos.

Nota: Para poder probar la clase, el vencimiento se controlará desde fuera de la clase y en esta quedará reflejado mediante un valor booleano.

Cuenta corriente

- No aplica intereses al saldo, pero permite al titular girar cheques y realizar reintegros a través del cajero automático. No obstante, el banco restringe el número de transacciones mensuales a una determinada cantidad y si el titular excede las transacciones; el banco le cobrará un recargo por transacción.
- Este tipo de cuentas no permite crédito y los posibles recargos se aplicarán mensualmente.

- Por ejemplo si tenemos 5 transacciones y realizamos 8 y el recargo por transacción adicional es de 1€, el banco nos cobrará 3€.
- Este recargo por pasar el número de transacciones, se realizara al finalizar el mes. Momento en el cual, se reiniciará el número de transacciones a cero.

```
Saldo = Saldo - (numeroTrasaccionesMesActual -
maximoTrasaccionesGratuitasPorMes) * recargoXTransaccionAdicional_Euros;
```

- Redefiniremos el método ToString para que indique el tipo de cuenta antes de llamar al de la clase base para que muestre sus datos básicos.

Cuenta crédito

- Permite al titular retirar dinero adicional al que indica su saldo, hasta un límite de crédito. Pero esto no es gratuito; al finalizar el mes, el banco aplicará una tasa de interés al saldo negativo o descubierto en ese momento.
- Por ejemplo, si nuestro saldo es -1000€ al 20%, pagaremos un recargo de 200€, de tal manera que el nuevo saldo será de -1200€.
- A diferencia de la cuenta corriente, la cuenta de crédito no tiene límite para la cantidad de transacciones que se pueden realizar sobre ella. Pues al banco le interesa que nos quedemos sin dinero para poder “chuparnos la sangre”.
- En todos los casos el control de tiempo se realizará de forma externa a las clases y los intereses, recargos, etc… se gestionarán a través de operaciones sobre los diferentes objetos de cuenta.
- Definiremos una excepción personaliza da **CreditoMaximoExcedidoException** como clase anidada pública. Esta escepción se generará si el usuario al intentar realizar un resintegro, superase el límite de credito establecido para la cuenta.
- Redefiniremos el método ToString para que indique el tipo de cuenta antes de llamar al de la clase base para que muestre sus datos básicos.

Programa principal

- Deberemos diseñar la jerarquía de clases que creamos más conveniente y probar el siguiente código…

```
class Ejercicio3 {
  static CuentaAhorro ca = new CuentaAhorro(
                    "2085 0103 92 0300731702", "Nicolas", .02d);
  static CuentaDeposito cd = new CuentaDeposito(
                    "2100 1162 43 0200084482", "Juan", .06d, .05d);
  static CuentaCorriente cc = new CuentaCorriente(
                    "2100 0721 09 0200601249", "Jhon", 2, 3d);
  static CuentaCredito cr = new CuentaCredito(
                    "0049 0345 31 2710611698", "Jose", .18d, 2000);

  static void SaldoActual(Cuenta cuenta){
        Console.WriteLine($"Saldo tras operación {cuenta.Saldo:C}\n");
  }
  static void Ingresa(Cuenta cuenta, double[] cantidades){
        foreach (double cantidad in cantidades){
              Console.WriteLine($"Ingresando {cantidad:C} en {cuenta}.");
              cuenta.Ingreso(cantidad);
              SaldoActual(cuenta);
        }
  }
  static void Retira(Cuenta cuenta, double[] cantidades){
        foreach (double cantidad in cantidades){
```

```
            try{
                  Console.WriteLine(
                              $"Retirando {cantidad:C} en {cuenta}.");
                  cuenta.Reintegro(cantidad);
            }
            catch (SaldoInsuficException e){
                  Console.WriteLine(e.Message);
            }
            catch (CuentaCredito.CreditoMaximoExcedidoException e){
                  Console.WriteLine(e.Message);
            }
            SaldoActual(cuenta);
      }
}
static void FinalizaMes(){
      Console.WriteLine("Finalizando mes actual ...");
      Console.WriteLine($"Aplicando intereses en {ca}...");
      ca.SumaInteres();
      SaldoActual(ca);
      Console.WriteLine($"Aplicando intereses en {cd}...");
      cd.SumaInteres();
      SaldoActual(cd);
      Console.WriteLine($"Aplicando recargos en {cc}...");
      cc.AplicaRecargosMes();
      SaldoActual(cc);
      Console.WriteLine(
            $"Revisando cargo sobre saldo negativo en {cr}...");
      double cargo = cr.CargaInteresesMes();
      if (cargo > 0d)
            Console.WriteLine(
            $"Has tenido un cargo de {cargo:C} por saldo negativo.");
      SaldoActual(cr);
}
static void Main(){
      Ingresa(ca, new double[] { 1000, 1000 });
      Ingresa(cd, new double[] { 10000 });
      Retira(ca, new double[] { 10000000, 500 });
      Ingresa(cc, new double[] { 1000, 2000 });
      Ingresa(cc, new double[] { 2000 });
      Retira(cc, new double[] { 500, 10000000, 500 });
      Ingresa(cr, new double[] { 1000 });
      Retira(cr, new double[] { 500 });
      Retira(cd, new double[] { 10000000, 1000 });
      FinalizaMes(); cd.Vencimiento = true;
      Ingresa(ca, new double[] { 1000, 1000 });
      Retira(ca, new double[] { 500 });
      Ingresa(cc, new double[] { 1000 });
      Retira(cc, new double[] { 100, 200, 100, 100, 100 });
      Retira(cr, new double[] { 1000 });
      Retira(cd, new double[] { 1000 });
      FinalizaMes();
      Retira(cr, new double[] { 11000 });
}
}
```

El resultado por consola de ejecutar el programa será parecido a el siguiente…

```
C:\> Ejercicio3.exe
Ingresando 1.000,00 € en Cuenta AhorroPresione una tecla para continuar . . .
Numero de cuenta: 2085-0103-92-0300731702
Titular: Nicolas
Saldo: 0,00 €.
Saldo tras operación 1.000,00 €

Ingresando 1.000,00 € en Cuenta Ahorro
Numero de cuenta: 2085-0103-92-0300731702
Titular: Nicolas
Saldo: 1.000,00 €.
Saldo tras operación 2.000,00 €

Ingresando 10.000,00 € en Cuenta Depósito
Cuenta Ahorro
Numero de cuenta: 2100-1162-43-0200084482
Titular: Juan
Saldo: 0,00 €.
Saldo tras operación 10.000,00 €

Retirando 10.000.000,00 € en Cuenta Ahorro
Numero de cuenta: 2085-0103-92-0300731702
Titular: Nicolas
Saldo: 2.000,00 €.
Saldo 2.000,00 € insuficiente para reintegro de 10.000.000,00 €.
En cuenta: 2085-0103-92-0300731702.
Saldo tras operación 2.000,00 €

Retirando 500,00 € en Cuenta Ahorro
Numero de cuenta: 2085-0103-92-0300731702
Titular: Nicolas
Saldo: 2.000,00 €.
Saldo tras operación 1.500,00 €

Ingresando 1.000,00 € en Cuenta Corriente
Numero de cuenta: 2100-0721-09-0200601249
Titular: Jhon
Saldo: 0,00 €.
Saldo tras operación 1.000,00 €

Ingresando 2.000,00 € en Cuenta Corriente
Numero de cuenta: 2100-0721-09-0200601249
Titular: Jhon
Saldo: 1.000,00 €.
Saldo tras operación 3.000,00 €

Ingresando 2.000,00 € en Cuenta Corriente
Numero de cuenta: 2100-0721-09-0200601249
Titular: Jhon
Saldo: 3.000,00 €.
Saldo tras operación 5.000,00 €

Retirando 500,00 € en Cuenta Corriente
Numero de cuenta: 2100-0721-09-0200601249
Titular: Jhon
Saldo: 5.000,00 €.
Saldo tras operación 4.500,00 €

Retirando 10.000.000,00 € en Cuenta Corriente
Numero de cuenta: 2100-0721-09-0200601249
Titular: Jhon
Saldo: 4.500,00 €.
Saldo 4.500,00 € insuficiente para reintegro de 10.000.000,00 €.
En cuenta: 2100-0721-09-0200601249.
Saldo tras operación 4.500,00 €

Retirando 500,00 € en Cuenta Corriente
Numero de cuenta: 2100-0721-09-0200601249
```

```
Titular: Jhon
Saldo: 4.500,00 €.
Saldo tras operación 4.000,00 €

Ingresando 1.000,00 € en Cuenta Crédito
Numero de cuenta: 0049-0345-31-2710611698
Titular: Jose
Saldo: 0,00 €.
Saldo tras operación 1.000,00 €

Retirando 500,00 € en Cuenta Crédito
Numero de cuenta: 0049-0345-31-2710611698
Titular: Jose
Saldo: 1.000,00 €.
Saldo tras operación 500,00 €

Retirando 10.000.000,00 € en Cuenta Depósito
Cuenta Ahorro
Numero de cuenta: 2100-1162-43-0200084482
Titular: Juan
Saldo: 10.000,00 €.
Saldo 10.000,00 € insuficiente para reintegro de 10.000.000,00 €.
En cuenta: 2100-1162-43-0200084482.
Saldo tras operación 10.000,00 €

Retirando 1.000,00 € en Cuenta Depósito
Cuenta Ahorro
Numero de cuenta: 2100-1162-43-0200084482
Titular: Juan
Saldo: 10.000,00 €.
Saldo tras operación 9.050,00 €

Finalizando mes actual ...
Aplicando intereses en Cuenta Ahorro
Numero de cuenta: 2085-0103-92-0300731702
Titular: Nicolas
Saldo: 1.500,00 €...
Saldo tras operación 1.530,00 €

Aplicando intereses en Cuenta Depósito
Cuenta Ahorro
Numero de cuenta: 2100-1162-43-0200084482
Titular: Juan
Saldo: 9.050,00 €...
Saldo tras operación 9.593,00 €

Aplicando recargos en Cuenta Corriente
Numero de cuenta: 2100-0721-09-0200601249
Titular: Jhon
Saldo: 4.000,00 €...
Saldo tras operación 3.988,00 €

Revisando cargo sobre saldo negativo en Cuenta Crédito
Numero de cuenta: 0049-0345-31-2710611698
Titular: Jose
Saldo: 500,00 €...
Saldo tras operación 500,00 €

Ingresando 1.000,00 € en Cuenta Ahorro
Numero de cuenta: 2085-0103-92-0300731702
Titular: Nicolas
Saldo: 1.530,00 €.
Saldo tras operación 2.530,00 €

Ingresando 1.000,00 € en Cuenta Ahorro
Numero de cuenta: 2085-0103-92-0300731702
Titular: Nicolas
Saldo: 2.530,00 €.
Saldo tras operación 3.530,00 €
```

```
Retirando 500,00 € en Cuenta Ahorro
Numero de cuenta: 2085-0103-92-0300731702
Titular: Nicolas
Saldo: 3.530,00 €.
Saldo tras operación 3.030,00 €

Ingresando 1.000,00 € en Cuenta Corriente
Numero de cuenta: 2100-0721-09-0200601249
Titular: Jhon
Saldo: 3.988,00 €.
Saldo tras operación 4.988,00 €

Retirando 100,00 € en Cuenta Corriente
Numero de cuenta: 2100-0721-09-0200601249
Titular: Jhon
Saldo: 4.988,00 €.
Saldo tras operación 4.888,00 €

Retirando 200,00 € en Cuenta Corriente
Numero de cuenta: 2100-0721-09-0200601249
Titular: Jhon
Saldo: 4.888,00 €.
Saldo tras operación 4.688,00 €

Retirando 100,00 € en Cuenta Corriente
Numero de cuenta: 2100-0721-09-0200601249
Titular: Jhon
Saldo: 4.688,00 €.
Saldo tras operación 4.588,00 €

Retirando 100,00 € en Cuenta Corriente
Numero de cuenta: 2100-0721-09-0200601249
Titular: Jhon
Saldo: 4.588,00 €.
Saldo tras operación 4.488,00 €

Retirando 100,00 € en Cuenta Corriente
Numero de cuenta: 2100-0721-09-0200601249
Titular: Jhon
Saldo: 4.488,00 €.
Saldo tras operación 4.388,00 €

Retirando 1.000,00 € en Cuenta Crédito
Numero de cuenta: 0049-0345-31-2710611698
Titular: Jose
Saldo: 500,00 €.
Saldo tras operación -500,00 €

Retirando 1.000,00 € en Cuenta Depósito
Cuenta Ahorro
Numero de cuenta: 2100-1162-43-0200084482
Titular: Juan
Saldo: 9.593,00 €.
Saldo tras operación 8.593,00 €

Finalizando mes actual ...
Aplicando intereses en Cuenta Ahorro
Numero de cuenta: 2085-0103-92-0300731702
Titular: Nicolas
Saldo: 3.030,00 €...
Saldo tras operación 3.090,60 €

Aplicando intereses en Cuenta Depósito
Cuenta Ahorro
Numero de cuenta: 2100-1162-43-0200084482
Titular: Juan
Saldo: 8.593,00 €...
Saldo tras operación 9.108,58 €

Aplicando recargos en Cuenta Corriente
```

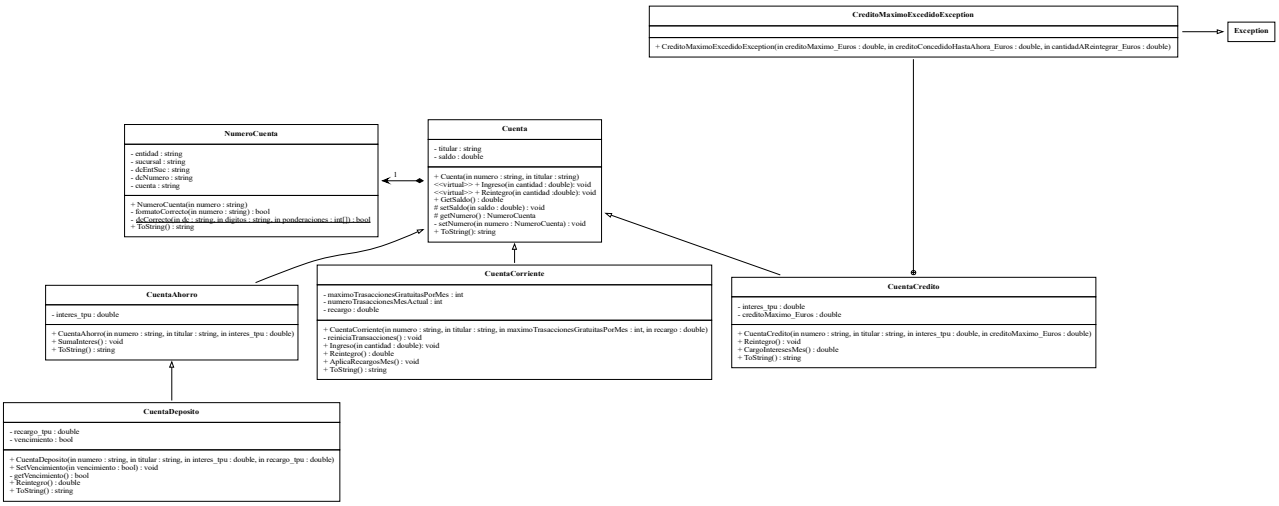
```
Numero de cuenta: 2100-0721-09-0200601249
Titular: Jhon
Saldo: 4.388,00 €...
Saldo tras operación 4.376,00 €

Revisando cargo sobre saldo negativo en Cuenta Crédito
Numero de cuenta: 0049-0345-31-2710611698
Titular: Jose
Saldo: -500,00 €...
Has tenido un cargo de 90,00 € por saldo negativo.
Saldo tras operación -590,00 €

Retirando 11.000,00 € en Cuenta Crédito
Numero de cuenta: 0049-0345-31-2710611698
Titular: Jose
Saldo: -590,00 €.
No puedes retirar 11.000,00 €.
Tu crédito máximo es de 2.000,00 € del cual se te ha concedido ya 590,00 €
Saldo tras operación -590,00 €
```

Diagrama de clases UML orientativo

- La siguiente imagen es un UML orientativo para las clases y métodos propuestos en el ejercicio.
- Solo úsalo si tienes dudas con el enunciado pues posiblemente en un examen no dispongas de él y debes saber si tu interpretación de el enunciado te da un resultado parecido.
- Puedes hacer zoom para verlo con más claridad.



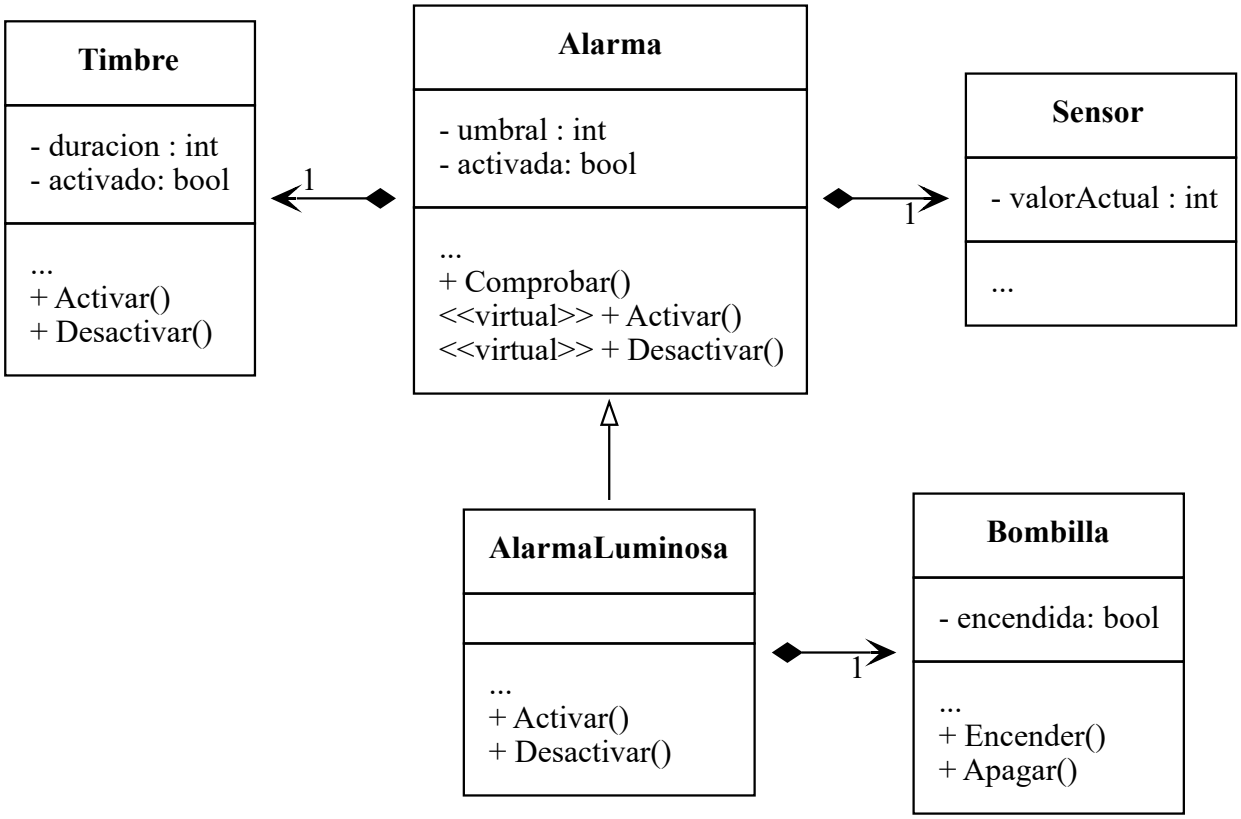
Ejercicio 4

Crea una clase denominada Alarma cuyos objetos activen un objeto de tipo Timbre cuando el valor medido por un Sensor supere un umbral preestablecido.

Implementa en C# todo el código necesario para el funcionamiento de la alarma, suponiendo que la alarma comprueba si debe activar o desactivar el timbre cuando se invoca el método comprobar().

Crea una subclase de Alarma denominada AlarmaLuminosa que, además de activar el timbre, encienda una luz (que representaremos con un objeto de tipo Bombilla).

El siguiente UML te puede ayudar a entender lo que se pide.



Nota: Procura eliminar la aparición de código duplicado al crear la subclase de Alarma y asegúrate de que, cuando se activa la alarma luminosa se enciende la luz de alarma y también suena la señal sonora asociada al timbre.