Desarrollo de Aplicaciones Web (D.A.W.)

Examen **Segundo Parcial 2024/25**

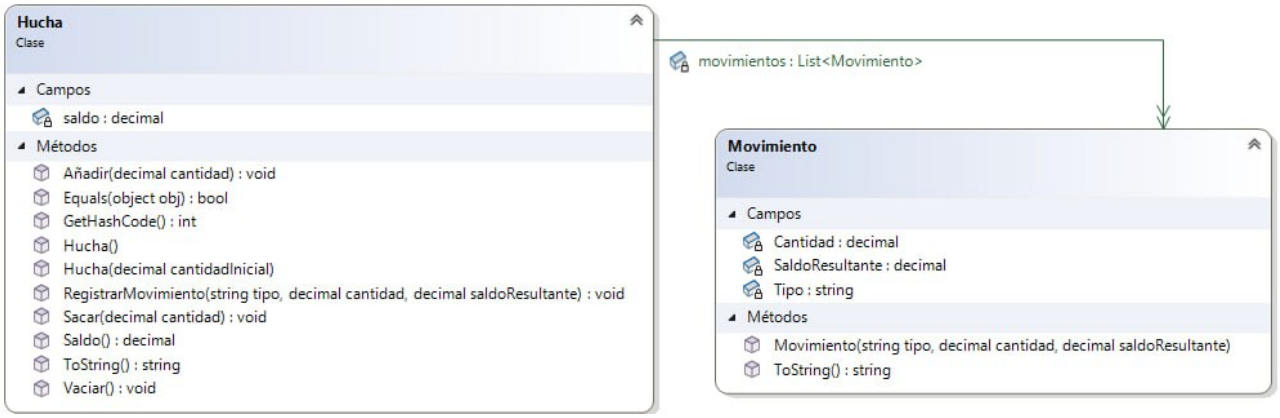
Nombre:_____

1.- **Implementa la clase Hucha** que simule el funcionamiento de una hucha, permitiendo realizar operaciones básicas de ahorro y gestión de dinero.
Debe permitir realizar las operaciones **Añadir** dinero, **Sacar** dinero, **Vaciar** la hucha y consultar el **Saldo** actual.

Además, debe mantener un registro detallado de todos los movimientos realizados mediante una lista de objetos **Movimiento** *(cuya clase se facilita)*.

- La clase **Hucha** debe tener las siguientes características:
  - Un constructor por defecto que inicialice la hucha con 10 euros.
  - Un constructor que permita inicializar la hucha con una cantidad específica.
  - Métodos para Añadir y Sacar dinero.
  - Un método para Vaciar completamente la hucha.
  - Un método para consultar el Saldo actual.
  - Un método ToString() que muestre todos los movimientos y el saldo final.
  - Un método Equals(…) que permita comparar Hucha y que las considere iguales si tienen el mismo saldo.
- La clase debe manejar lo siguiente:
  - Lanzar Exception solo en caso de inicializar la hucha con una cantidad negativa.
  - Registrar como "Operación anulada" los intentos de añadir o sacar cantidades negativas.
  - Registrar como "Operación anulada" los intentos de sacar más dinero del disponible.
  - Registrar como "Añadido" "Sacado" ó "Vaciado" las acciones correctas de añadir, sacar o vaciar la hucha.
- Utilizar la clase Movimiento proporcionada para registrar cada operación realizada en la hucha.

Diagrama de clases:



Clase Movimiento (proporcionada):

```
public class Movimiento {
    private string Tipo;
    private decimal Cantidad;
    private decimal SaldoResultante;

    public Movimiento(string tipo, decimal cantidad, decimal saldoResultante) {
        if (cantidad < 0 || saldoResultante < 0)
            throw new Exception("Solo se admiten cantidades positivas");
        Tipo = tipo;
        Cantidad = cantidad;
        SaldoResultante = saldoResultante;
    }

    public override string ToString() {
        return $"{Tipo}: {Cantidad} (Saldo: {SaldoResultante})";
    }
}
```

Ejemplo de uso:

```
class ProgramaPruebaHucha {
    static void Main(string[] args) {
        Console.Write("\n************************************  ");
        Console.WriteLine("Creando una hucha con 50 euros...");
        Hucha miHucha = new Hucha(50);
        Console.WriteLine(miHucha);

        Console.Write("\n************************************  ");
        Console.WriteLine("Añadiendo 20 euros...");
        miHucha.Añadir(20);
        Console.WriteLine(miHucha);

        Console.Write("\n************************************  ");
        Console.WriteLine("Intentando añadir una cantidad inválida (-10 euros)...");
        miHucha.Añadir(-10);
        Console.WriteLine(miHucha);

        Console.Write("\n************************************  ");
        Console.WriteLine("Sacando 30 euros...");
        miHucha.Sacar(30);
        Console.WriteLine(miHucha);

        Console.Write("\n************************************  ");
        Console.WriteLine("Intentando sacar más dinero del que hay en la hucha (100 euros)...");
        miHucha.Sacar(100);
        Console.WriteLine(miHucha);

        Console.Write("\n************************************  ");
        Console.WriteLine("Vaciando la hucha...");
        miHucha.Vaciar();
        Console.WriteLine(miHucha);

        Console.Write("\n************************************  ");
        Console.WriteLine("Intentando sacar dinero de una hucha vacía (10 euros)...");
        miHucha.Sacar(10);
        Console.WriteLine(miHucha);

        Console.Write("\n************************************  ");
        Console.WriteLine("Terminamos metiendo dinero (2130 euros)...");
        miHucha.Añadir(2130);
        Console.WriteLine(miHucha);
    }
}
```

Resultado de su ejecución:

```
************************************  Creando una hucha con 50 euros...
--->  [Ahorrado: 50]

************************************  Añadiendo 20 euros...
Añadido: 20 (Saldo: 70)
--->  [Ahorrado: 70]

************************************  Intentando añadir una cantidad inválida (-10 euros)...
Añadido: 20 (Saldo: 70)
Operación anulada por valor negativo: 10 (Saldo: 70)
--->  [Ahorrado: 70]

************************************  Sacando 30 euros...
Añadido: 20 (Saldo: 70)
Operación anulada por valor negativo: 10 (Saldo: 70)
Sacado: 30 (Saldo: 40)
--->  [Ahorrado: 40]

************************************  Intentando sacar más dinero del que hay en la hucha (100 euros)...
Añadido: 20 (Saldo: 70)
Operación anulada por valor negativo: 10 (Saldo: 70)
Sacado: 30 (Saldo: 40)
Operación anulada por saldo insuficiente : 100 (Saldo: 40)
--->  [Ahorrado: 40]

************************************  Vaciando la hucha...
Añadido: 20 (Saldo: 70)
Operación anulada por valor negativo: 10 (Saldo: 70)
Sacado: 30 (Saldo: 40)
Operación anulada por saldo insuficiente : 100 (Saldo: 40)
Vaciado: 40 (Saldo: 40)
--->  [Ahorrado: 0]

************************************  Intentando sacar dinero de una hucha vacía (10 euros)...
Añadido: 20 (Saldo: 70)
Operación anulada por valor negativo: 10 (Saldo: 70)
Sacado: 30 (Saldo: 40)
Operación anulada por saldo insuficiente : 100 (Saldo: 40)
Vaciado: 40 (Saldo: 40)
Operación anulada por saldo insuficiente : 10 (Saldo: 0)
--->  [Ahorrado: 0]

************************************  Terminamos metiendo dinero (2130 euros)...
Añadido: 20 (Saldo: 70)
Operación anulada por valor negativo: 10 (Saldo: 70)
Sacado: 30 (Saldo: 40)
Operación anulada por saldo insuficiente : 100 (Saldo: 40)
Vaciado: 40 (Saldo: 40)
Operación anulada por saldo insuficiente : 10 (Saldo: 0)
Añadido: 2130 (Saldo: 2130)
--->  [Ahorrado: 2130]
```

2.- Implementa una función **Sorteo(…)** en una nueva clase **HuchaHija** (que herede de **Hucha**) que reciba un array de objetos Hucha y que retorne una de ellas elegida al azar de entre aquellas que no estén vacías.

```
 … …
Hucha resultado = HuchaHija.sorteo(Hucha[] huchas);
 …   … …
```

consideraciones.- *- El atributo saldo de la clase Hucha es privado. Esto no se debe cambiar.*
- Hucha hija solo dispone de un constructor, que inicializa su saldo a cero euros.

Asignación de Puntos (por fragmento de ejercicio correctamente implementado):

1er ejercicio	3 puntos - Atributos, constructores y registrar movimientos. 3 puntos – añadir, sacar y saldo dejando registro de operaciones 2 puntos – ToString, Equals (y GetHashCode).
2º ejercicio	2 puntos