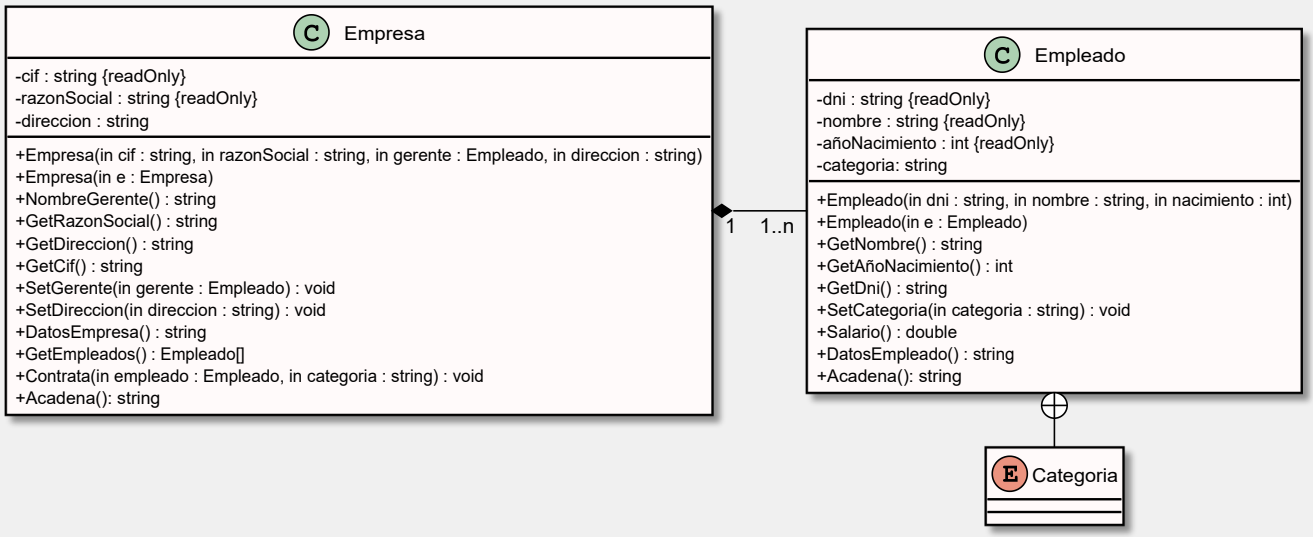
Ejercicios Objetos Básicos y Objetos Valor

Índice

Ejercicio 1

A partir del siguiente diagrama de clases, crea las clases necesarias que permitan reflejar los elementos que ves representados en él, y la aplicación para probarlas. Las categorías de los trabajadores pueden ser:
Subalternos=10, Administrativos=20, JefesDepartamento=40, Gerente=60.

Nota: El salario del empleado se calculará a partir de la categoría de trabajo que desempeñe, las categorías tienen asociadas el incremento en porcentaje que se tiene que hacer al salario base, que será una constante con el valor de 1200.



Una posible salida sería:

```
La Empresa S.L
María Soto del Rio
Calle el Pozo, 34 Bajo
El empleado María Soto del Rio con dni: 23453456L tiene un salario 1920 y su categoria es: Gerente
El empleado Juanma Perez Ortiz con dni: 14568712G tiene un salario 1440 y su categoria es: Administrativos
El empleado Pedro Martinez Sancho con dni: 12346123K tiene un salario 1440 y su categoria es: Administrativos
```

Ejercicio 2

Crea un programa que permita guardar información sobre las características gráficas de un equipo informático. La información relevante es:

- Número de pulgadas (short)
- Controlador gráfico (string)
- Número de colores (short)
- Pixeles eje x (int)
- Pixeles eje y (int)

Debe ser posible cambiar el número de colores, la resolución de la pantalla y el controlador en cualquier momento, así como consultar toda la información concerniente a las características gráficas. ***Crea alguna instancia de pantalla y prueba el funcionamiento.***
Nota: todos los campos serán privados, y deberemos crear **solo** los métodos y accesores o modificadores necesarios para poder hacer funcional la aplicación.

☑ Ejercicio 3

Crea un proyecto con **los TAD necesarios** para que el siguiente código perteneciente a la Main, pueda ser ejecutado sin problemas:

```
Compas compas = new Compas();
Circulo circulo = compas.DibujaCirculo(3.5f);
Rotulador rotulador = Estuche.GetRotuladores()[8]??new Rotulador("Negro");
rotulador.Rotula(circulo.Perimetro());
Pincel pincel=new Pincel();
pincel.SetColor(Color.Verde);
pincel.Pinta(circulo.Area());
```

👉**Pista:** El circulo tendrá un atributo rádio.El rotulador tendrá un atributo color de tipo enumerado. Habrá una clase estática Estuche con un solo método también estático que devolverá un array de rotuladores. El pincel también tiene un atributo color.
La salida por pantalla del programa podría ser algo como lo siguiente:

```
Dibujado un círculo de rádio 3,5
Rotulado el perimetro de  38,48 cm. de color Azul.
Pintada el area de  21,99 cm. de color Verde.
```

Ejercicio 4

A partir del siguiente texto:
Una cadena de muebles a nivel nacional tiene tiendas ubicadas por toda la geografía española.
Cada una de estas tiendas de muebles almacena información sobre: sofás, sillas y mesas. De todos ellos es común, como mínimo, color(enumeración), peso, dimensiones (estructura), fabricante y precio. En cambio los sofás incluye información de la tela con el que está tapizado y si es abatible o no, las sillas la longitud del respaldo y las mesas el tipo de madera que está hecho.

Crea **una clase Mueble** que contenga los campos, constructores y métodos necesarios para recoger la información de un mueble y devolver mediante un método ACadena esa información. **Importante que al recoger los valores de color pertenezcan a la enumeración**.

Nota: Debes de fijarte que el ejercicio pide que hagas **solo la clase Mueble**, con las características comunes a todos los muebles, el resto de texto se usará en ejercicios posteriores.

Ejercicio 5

Crea una clase Humano con los campos, como mínimo: nombre, edad, peso, sexo, inteligencia, fuerza, destreza y energía.
Los métodos MostrarInformación, SetNombre, SetEdad,...y los constructores que creas necesarios.

☑ Ejercicio 6

Debes definir un **tipo valor** que represente un **Naipe** de la baraja Española de 48 cartas. El tipo estará compuesto por dos miembros: un valor y un palo, este último sera de tipo enumerado con los siguientes valores posibles: Oros, Copas, Bastos, Espada.
Crea un método en la clase principal que utilizando el tipo Naipe nos devuelva una baraja con las 48 cartas, usa una matriz `Naipe[,] baraja= new Naipe[4,12]` e inicialízala suponiendo que cada fila representa un palo.
Crea un método que nos mezcle la baraja para que queden sus cartas desordenadas.
Crea un programa que muestre el resultado de la utilización de los métodos anteriores.

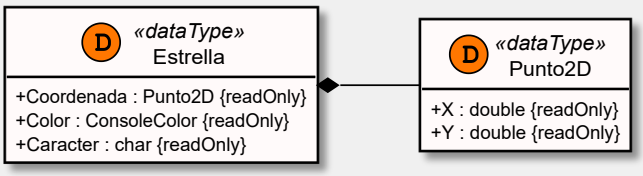
Nota: Vuelve a repasar los apuntes donde se explica la creación del tipo Valor y sigue las normas que se explican para este tipo.

Ejercicio 7

Utilizando el tipo valor Punto definido en los apuntes, vamos a crear otro tipo valor **Estrella** formado por un punto, el caracter ***** y un color aleatorio de la enumeración ConsoleColor.
En el programa principal seguiremos los siguientes pasos:

- Deberemos introducir por teclado la cantidad de estrellas a dibujar.
- Además almacenaremos los datos en una tabla definida de la siguiente manera `Estrella[] estrellas = new Estrella[NumEstrellas];`
- Generaremos la posición del Punto y el Color de cada estrella de forma aleatoria.
- Borraremos la pantalla con el método Clear() de la clase Console.
- Dibujaremos todo el array de estrellas en cada una de las coordenadas introducidas y con el color generado aleatorio.

Nota: para posicionar la estrella en la pantalla usaremos el método de la clase Console `void SetCursorPosition(int letf, int top);` que situará el cursor en una coordenada dentro de las 24 filas y 80 columnas que puede tener una consola con origen en la esquina superior izquierda (para generar los puntos, utilizaremos en Next de la clase Random con el rango entre 24 y 80). Para pintar el color utilizaremos la propiedad, de la clase Console, ForegroundColor.



Ejercicio 8

Crea las clases silla, mesa y sofá hijas de la clase mueble del ejercicio 4. A cada una de ellas le deberás de añadir los campos y métodos necesarios que identifiquen a dichos elementos según el texto del ejercicio 4.
Crea un método común para todos los muebles que nos permita calcular el precio del transporte según su peso, dimensiones y fabricante (puedes usar la fórmula que consideres para hacer este cálculo).
Finalmente, haz un programa principal que nos cree instancias de los tres tipos de
muebles, que llame a los métodos de las clases para mostrar la información y que llame al método para calcular el precio del transporte y lo muestre.
Nota: Debes pensar en que clase o clases vas a incluir el método de calcular precio, para seguir las normas que se explican en el tema.

Ejercicio 9

Se pide implementar usando POO y herencia las siguientes entidades:

- La clase **Local**:
  - Con los campos ciudad, calle, numero plantas y dimensiones.
  - Con la operación GetNumeroPlantas para poder acceder al atributo numeroPlantas.
  - Con la operación ACadena que devolverá un string con los datos de un local.
- La clase **LocalComercial** que heredará de Local.
  - Con los campos razón social y número licencia.
  - Con la operación ACadena que devolverá un string con los datos de un local comercial.
- La clase **Cine** que heredará de LocalComercial.
  - Con el campo aforo sala.
  - Con la operación ACadena que devolverá un string con los datos de un cine.

Deberemos implementar un programa principal que cree un array de 3 cines y lo inicialice con datos de supuestos cines. Después deberá mostrar por pantalla la información completa de cada uno de los cines mediante un foreach, a través del método ACadena.
Nota: Deberás soobreescribir el método ACadena, para poder aprovechar el código implementado en las clases padre.

Ejercicio 10

Para dar nuestros primeros pasos con el polimorfismo de datos **del principio de sustitución de Liskov**, vamos a recordar el cálculo de factorial y de número primo.
Vamos a necesitar una clase padre **ElNoCalculador** con un atributo **protected short numero**, que será la base del calculo posterior, también tendrá dos métodos virtuales **Factorial** y **Primo** devolverán un double y un booleano respectivamente
`(en esta clase no se realizarán los cálculos, los métodos devolverán 0 y false)` .
Además, crearemos una clase hija **ElCalculador** que sí implementa estos métodos de forma correcta a partir del numero del padre.
Para probar el funcionamiento del polimorfismo nos crearemos un objeto de la manera: **ElNoCalculador obj=new ElCalculador(num);**
Y llamaremos a los métodos Primo y Factorial. Reflexiona las siguiente preguntas:
`¿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`

☑ Ejercicio 11

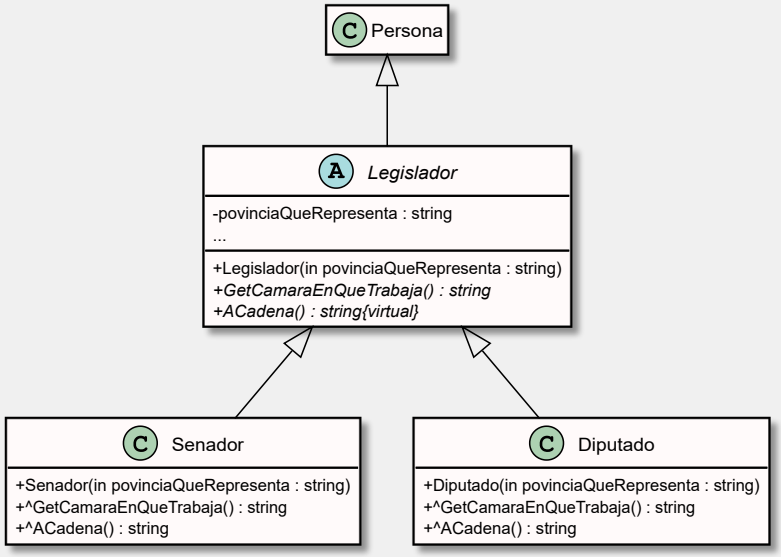
Crea una clase **Guerrero** que herede de la clase **Humano** del ejercicio anterior, añade los campos tipoArma y tipoArmadura y los métodos que creas necesarios, para hacerla funcional.
Crea también la clase **Mago** hija de humano con los atributos tipoLibroHechizos y tipoTúnica y los métodos necesarios.
Nota: Sería interesante que utilices tipos enumerados para los campos anteriormente descritos.

Ejercicio 12

Declara una clase abstracta **Legislador** que herede de la clase **Persona**, que usamos en ejercicios anteriores, con un campo **povinciaQueRepresenta** tipo String, y si quieres otros atributos que creas necesarios.
Como puedes ver en el DEM, la clase Legislador hereda de la clase Persona que no es abstracta, lo cual significa que puede haber instancias de Persona pero no de Legislador.
Declara un método abstracto GetCamaraEnQueTrabaja (devolverá un string indicando la cámara a la que pertenece) y uno virtual ACadena (devolverá un string con los datos a mostrar de la clase).
Crea las siguientes clases que heredarán de Legislador e implementarán su método abstracto y virtual:

- **Diputado.**
- **Senador.**

Crea un **array de tres legisladores** de distintos tipos, usa **polimorfismo de datos** y muestra por pantalla la cámara en que trabajan y otros datos que creas necesarios.



☑ Ejercicio 13

Crear una clase llamada **TablaEnteros** que **no permita** que se creen objetos a partir de ella.
Esta clase almacenará una tabla de enteros de una dimensión, el tamaño debe ser especificado mediante su constructor.
`La clase debe obligar a que cualquier clase que herede de ella y no quiera ser una clase abstracta, implemente un método llamado GuardarNume`
.
Además, **TablaEnteros** dispondrá de dos método:

- **MostrarTabla**, método `redefinible` , que servirá para mostrar la tabla completa por pantalla.
- **SumaPropia** que se encargará de comprobar si existen más números positivos o negativos en la tabla y devolverá la suma de aquellos de los que hay mayor cantidad.

Crear a partir de esta clase dos nuevas clases llamadas:

- **TablaImpares**: que solo guardará números impares.
- **TablaPares**: que solo guardará números pares.

Ambas, lo harán mediante el método **GuardarNumerosEnTabla** antes mencionado que seleccionará los números apropiados (pares o impares) a guardar, de una lista introducida por el usuario.
En el programa principal, crea instancias de cada una de estas clases, dales valores y muestra las tablas y la suma propia de ambos objetos por pantalla.