Índice

Ejercicios Unidad 14 - Roles entre clases. Todo o Parte

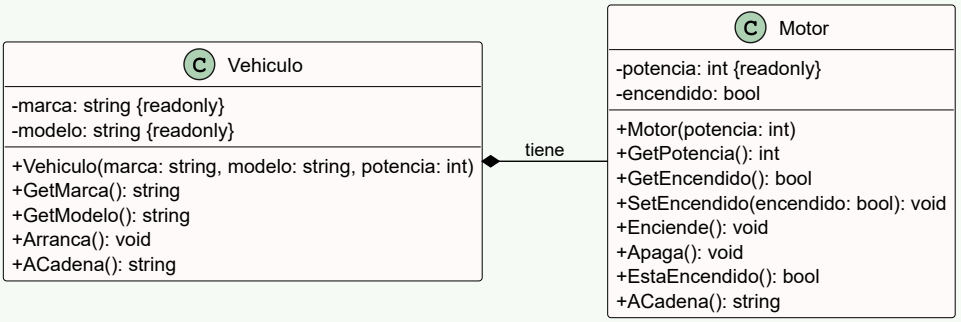
Descargar estos ejercicios

> **Antes de empezar**
>
> Para realizar estos ejercicios, deberás descargar los recursos del enlace de proyecto_roles_todo_parte. Como puedes ver, la solución está compuesta de varios proyectos. Cada uno de ellos corresponde con un ejercicio, deberás implementar todo el código, tanto de la Main como de los métodos que se piden en cada ejercicio. Cada proyecto contiene el test correspondiente, que deberás pasar para comprobar que has hecho el ejercicio correctamente.

Ejercicio 1. Composición: Vehículo y Motor

Implementa una relación de **composición** entre las clases `Vehiculo` y `Motor`. El motor es parte integral del vehículo y no puede existir sin él.

```
Ejercicio 1: Composición Vehículo-Motor

Creando vehículo con motor integrado...

Vehículo: Toyota Corolla
Motor: 120 CV - Estado: Apagado

Arrancando el vehículo...

Vehículo: Toyota Corolla
Motor: 120 CV - Estado: Encendido

Presiona cualquier tecla para salir...
```



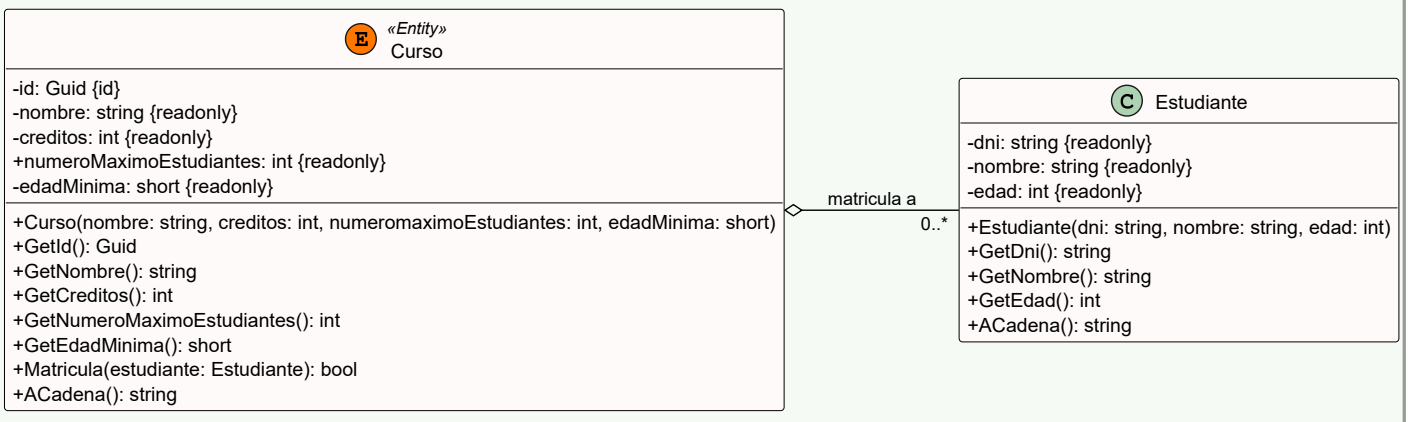
Requisitos:

- Crea las clases necesarias para implementar correctamente el diagrama de clases. Sobre todo, fíjate en la relación entre clases y lo que conlleva.
- En el programa principal, crea las instancias de los objetos necesarias y Muestra el estado inicial como se ve en la salida
- Arrancar el vehículo y muestra el estado del vehículo después de arrancado.

Ejercicio 2. Agregación: Curso y Estudiantes

Implementa una relación de **agregación** 1 a muchos entre las clases `Curso` y `Estudiante`. Un curso puede tener varios estudiantes matriculados, pero los estudiantes existen independientemente del curso.

```
Ejercicio 2: Agregación Curso-Estudiantes (1 a muchos)

Creando curso independiente...
Curso: Programación (6 créditos) - 0 estudiantes matriculados

Matriculando estudiantes en el curso...
Matriculando a Ana García...
Matriculando a Luis Pérez...
Matriculando a María López...

Curso después de las matriculaciones:
Curso: Programación (6 créditos) - 3 estudiantes matriculados
Estudiantes:
  - Ana García (20 años)
  - Luis Pérez (19 años)
  - María López (21 años)

Matriculando estudiantes en el curso...
Matriculando a Pedro Sanchez...
Lo siento, este estudiante no tiene la edad adecuada, debe ser mayor de 17 años.

Matriculando a Marisa Rodríguez...
Matriculando a Rosa Palacios...
Lo siento, ya no se pueden matricular más estudiantes, el curso está completo.

Curso después de las matriculaciones:
Curso: Programación (6 créditos) - 3 estudiantes matriculados
Estudiantes:
  - Ana García (20 años)
  - Luis Pérez (19 años)
  - María López (21 años)
  - Marisa Rodríguez (32 años)

Presiona cualquier tecla para salir...
```



Requisitos:

- La relación la representaremos con una lista privada de estudiantes matriculados, como se puede ver en el tema.
- Método `Matricula(Estudiante estudiante)` para matriculaf un nuevo estudiante, deberán de quedar plazas y el estudiante debe tener la edad adecuada. Devolviendo true en caso de que la matrícula sea exitosa.
- El programa principal, tendrá un método público y estático `GestionMatricula` que creara los objetos necesarios y las llamadas a los métodos para conseguir la salida especificada.

Ejercicio 3. Sistema de coordenadas con records

Implementa un sistema de coordenadas geográficas usando records para representar Value Objects inmutables.

```
Ejercicio 3: Sistema de coordenadas con records

Creando puntos de interés turístico...
Punto 1: Torre Eiffel
Coordenadas: 48.8584° N, 2.2945° E
Altitud: 300 metros

Punto 2: Sagrada Familia
Coordenadas: 41.4036° N, 2.1744° E
Altitud: 170 metros

Punto 3: Big Ben
Coordenadas: 51.4994° N, -0.1245° E
Altitud: 96 metros

Calculando distancias entre puntos...
Distancia entre Torre Eiffel y Sagrada Familia: 831.45 km
Distancia entre Torre Eiffel y Big Ben: 344.12 km
Distancia entre Sagrada Familia y Big Ben: 1137.89 km

--- Verificación de hemisferios ---
Torre Eiffel: Hemisferio Norte (true), Hemisferio Este (true)
Sagrada Familia: Hemisferio Norte (true), Hemisferio Este (true)
Big Ben: Hemisferio Norte (true), Hemisferio Este (false)

--- Movimiento individual de coordenadas ---
Torre Eiffel movida 0.01° al Norte: 48.8684° N, 2.2945° E, 300m
Sagrada Familia movida 0.05° al Este: 41.4036° N, 2.2244° E, 170m

Creando ruta Europa Occidental...
Agregando Torre Eiffel a la ruta...
Agregando Sagrada Familia a la ruta...
Agregando Big Ben a la ruta...

--- Estado inicial de la ruta ---
Ruta Europa Occidental:
1. Torre Eiffel (París) - 48.8584° N, 2.2945° E, 300m
2. Sagrada Familia (Barcelona) - 41.4036° N, 2.1744° E, 170m
3. Big Ben (Londres) - 51.4994° N, -0.1245° E, 96m

--- Cálculos generales ---
Distancia total de la ruta: 1175.57 km
Altitud promedio de la ruta: 188.67 metros
¿La ruta tiene más puntos al Este? true (2 puntos en hemisferio Este vs 1 en Oeste)

--- Moviendo toda la ruta 0.1° al Norte ---
Aplicando movimiento a todos los puntos...

--- Estado después del movimiento ---
Ruta Europa Occidental (movida al Norte):
1. Torre Eiffel (París) - 48.9584° N, 2.2945° E, 300m
2. Sagrada Familia (Barcelona) - 41.5036° N, 2.1744° E, 170m
3. Big Ben (Londres) - 51.5994° N, -0.1245° E, 96m

--- Nuevos cálculos tras el movimiento ---
Nueva distancia total: 1175.32 km
Nueva altitud promedio: 188.67 metros (sin cambios)
¿La ruta modificada sigue teniendo más puntos al Este? true
```

```
"Presiona cualquier tecla para salir..."
```

Requisitos:

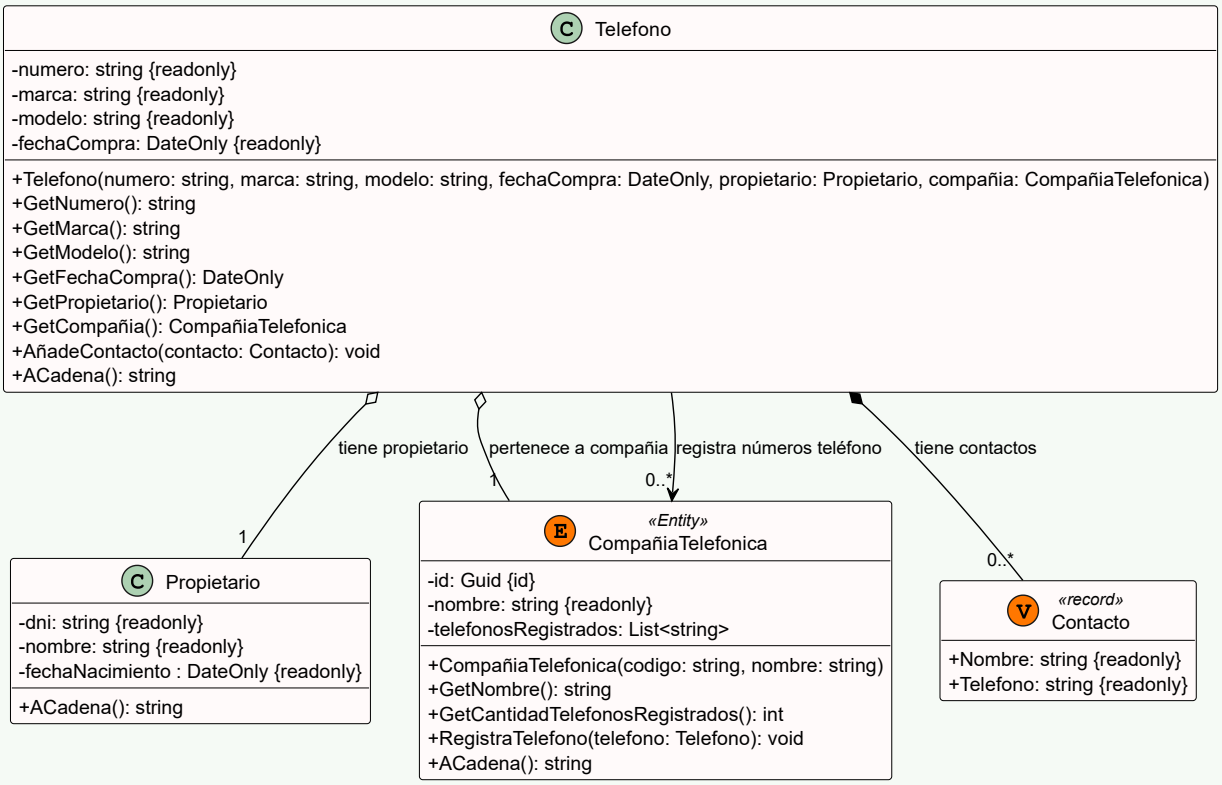
- **Record Coordenada**:
  - Con dos propiedades de tipo double Latitud, Longitud y una entera Altitud.
  - Este record deberá validar las entradas por lo que será de tipo extendido. Validará latitud entre -90 y 90 grados, validará longitud entre -180 y 180 grados y validará altitud >= 0. De forma que si alguno de los valores sobrepasa los límites, se asignará por defecto el valor del límite.
  - **Métodos en Coordenada**:
    - `DistanciaA(Coordenada otra)` método que devuelve un double con la distancia entre dos coordenada usando la fórmula de Haversine en km (busca en la nube como es esta fórmula).
    - `EsHemisferioNorte()` método que devolverá true si la latitud >= 0.
    - `EsHemisferioEste()` método que devolverá true si la longitud >= 0.
    - `MueveNorte(double grados)` Crea una nueva Coordenada y la devuelve con, solo, la latitud modificada. Teniendo en cuenta que si la latitud sobrepasa 90 o -90 se tendrá que dejar en sus valores tope 90 o -90
    - `MueveEste(double grados)` Crea una nueva Coordenada y la devuelve con, solo, la longitud modificada. Teniendo en cuenta que si la nueva longitud supera los 180 o los -180 grados, habrá que restar 360 o sumar 360 respectivamente.
- **Record PuntoInteres de tipo simplificado**:
  - Con las propiedades de tipo string, Nombre y de tipo Coordenada, Ubicacion.
- **Clase RutaTuristica**:
  - Contiene unà propiedad id privada y autoimplementada de tipo Guid.
  - Contiene una lista privada de `PuntoInteres` que representará la relación de agregación.
  - Contiene una propiedad de solo lectura `Nombre` y pública de acceso.
  - Contiene una propiedad de solo lectura `Descripción` y pública de acceso.
  - `AgregaPunto(PuntoInteres punto)` método que añade un punto de interes a la ruta.
  - `CalculaDistanciaTotal()` método que devuelve un doble con la distancia total de toda la ruta.
  - `CalculaAltitudPromedio()` método que devuelve un doble con la altitud promedio de toda la ruta.
  - `MueveNorteRuta(double grados)` método que mueve al norte todos los puntos de la ruta, con los grados de entrada.
  - `RutaMasAlEste` método que devolverá true, si hay más puntos situados en el Hemisferio Este.
  - `MuestraRuta()` método que muestra los datos de la ruta.
- **Clase Program**:
  Crear un método en el programa principal `GestionRutas` que realice las siguientes funciones:
  - cree los 3 puntos de interés indicados
  - Calcular distancias entre puntos usando `DistanciaA()`
  - Verificar hemisferios con `EsHemisferioNorte()` y `EsHemisferioEste()`
  - Demostrar movimientos individuales con `MueveNorte()` y `MueveEste()`
  - Crear ruta turística y agregar puntos con `AgregaPunto()`
  - Calcular estadísticas de ruta con `CalculaDistanciaTotal()` y `CalculaAltitudPromedio()`
  - Verificar distribución hemisférica con `RutaMasAlEste()`

- Aplicar movimiento a toda la ruta con `MueveNorteRuta()`
- Mostrar ruta completa con `MuestraRuta()`

Ejercicio 4. Sistema de gestión de teléfonos

Implementa un sistema de gestión de teléfonos con diferentes tipos de relaciones entre clases: agregación (1:1), composición (1:n), asociación (1:n) y uso de tipos valor.

```
Ejercicio 4: Sistema de gestión de teléfonos

Creando propietario del teléfono...
Ana García
DNI: 12345678A
Fecha de nacimiento: 15/05/1990

Creando compañía telefónica...
Compañía: Movistar
Código: ES001
Id: 8866cea1-38a3-469f-aceb-057addbd4cfd

Creando teléfono con propietario...
Teléfono Número: 699000111
Marca: iPhone, Modelo: 15 Pro
Fecha de compra: 15/08/2025
Propietario: Ana García (DNI: 12345678A)
Compañía: Movistar (ES001)
Contactos almacenados: 0

Añadiendo contactos al teléfono...
Contacto añadido: Luis Pérez - 666111222
Contacto añadido: María López - 677333444
Contacto añadido: Pedro Ruiz - 688555666

Registrando teléfono en la compañía...
Teléfono registrado en Movistar
Teléfonos registrados en Movistar: 1

=== ESTADO FINAL DEL SISTEMA ===
--- Teléfono ---
Teléfono Número: 699000111
Marca: iPhone, Modelo: 15 Pro
Fecha de compra: 15/08/2025
Propietario: Ana García (DNI: 12345678A)
Compañía: Movistar (ES001)
Contactos almacenados: 3
  - Luis Pérez: 666111222
  - María López: 677333444
  - Pedro Ruiz: 688555666
--- Compañía ---
Compañía: Movistar (ES001)
Teléfonos registrados: 1

Presiona cualquier tecla para salir...
```



Requisitos:

- En la clase **CompañiaTelefonica**:
  - Se deberá crear una lista privada de strings con números de teléfonos registrados representará la (asociación).
  - Método `RegistraTelefono` que **extrae el número del teléfono** y lo añade a la lista de strings. La relación de asociación se produce únicamente a través de este método ya que en este método se usa el TAD Teléfono.
- En el programa principal se creará un método público y estático `GestionTelefono`. En el que se implementarán los objetos y las llamadas a los métodos necesarios para conseguir la salida.

Ejercicio 5. Agregación múltiple: Sistema de refugio de animales

Implementa un sistema de refugio de animales con agregación múltiple (muchos a muchos) entre `Animal` y `Cuidador`, donde un animal puede tener varios cuidadores y un cuidador puede cuidar varios animales. Esta relación se traspasa a una clase nueva Refugio.

```
Ejercicio 5: Sistema de refugio con agregación múltiple

Creando refugio...
Refugio creado: Hogar Feliz

Registrando cuidadores en el refugio...
Cuidador añadido: Ana López (Veterinaria) - Máximo 5 mascotas
Cuidador añadido: Carlos Ruiz (Entrenador) - Máximo 3 mascotas
Cuidador añadido: María Pérez (Voluntaria) - Máximo 2 mascotas

Registrando animales en el refugio...
Animal añadido: Max (Perro) - Edad: 3 años
Animal añadido: Luna (Gato) - Edad: 2 años

Realizando asignaciones de cuidadores...
Asignando Max a Ana López...
  - Verificando disponibilidad de Ana López... Disponible
  - Max ha sido asignado correctamente
Asignando Max a Carlos Ruiz...
  - Verificando disponibilidad de Carlos Ruiz... Disponible
  - Max ha sido asignado correctamente
Asignando Luna a Ana López...
  - Verificando disponibilidad de Ana López... Disponible
  - Luna ha sido asignado correctamente
Asignando Luna a María Pérez...
  - Verificando disponibilidad de María Pérez... Disponible
  - Luna ha sido asignado correctamente

Estado actual del refugio:
Refugio: Hogar Feliz
Total de animales: 2
Total de cuidadores: 3
Total de asignaciones: 4
--- Información de animales ---
Animal: Max (Perro) - 3 años
  Estado: Asignado
  Número de cuidadores: 2

Animal: Luna (Gato) - 2 años
  Estado: Asignado
  Número de cuidadores: 2

--- Información de cuidadores ---
Cuidador: Ana López (Veterinaria)
  Mascotas asignadas: 2/5
  Disponible para más asignaciones: Sí
Cuidador: Carlos Ruiz (Entrenador)
  Mascotas asignadas: 1/3
  Disponible para más asignaciones: Sí
Cuidador: María Pérez (Voluntaria)
  Mascotas asignadas: 1/2
  Disponible para más asignaciones: Sí
--- Resumen del refugio ---
Detalle de asignaciones:
  - Max  Ana López
  - Max  Carlos Ruiz
  - Luna  Ana López
  - Luna  María Pérez
```

Requisitos:

- **Clase Animal**:
  - Propiedades `id` (Guid), `Nombre`, `Especie` (string) de solo lectura.
  - Propiedad privada `Asignado` (bool).
  - Proiedad `Edad` (int) pública de acceso y modificación.
  - Método `EstaAsignado()` para notificar al refugio que la mascota ha sido asignada mediante un boolean.
  - Método `Libera` que modifica la propiedad Asignado a false.
  - Método `Asigna` que permite asignar la propiedad a true.
  - Métodos `ACadena()`
- **Clase Cuidador**:
  - Propiedades `Dni`, `Nombre`, `NumeroMaximoMascotas` y `Especialidad` (string) de solo lectura.
  - Propiedad pública `NumeroMascotasAsignadas` de tipo entero.
  - Método público `AsignaMascotaSiDisponible` que comprueba si todavía puede ser asignado a una mascota. Devolviendo true en caso afirmativo e incrementando el número de mascotas asignadas.
  - Método públic `Libera` que decrementará el número de mascotas asignadas en uno.
  - Métodos `ACadena()`
- **Clase Refugio** (entidad externa gestora):
  - Propiedad `id` (Guid)
  - Propiedad `Nombre` (string, readonly)
  - Lista privada de animales gestionados (agregación)
  - Lista privada de cuidadores gestionados (agregación)
  - Lista privada de asignaciones (animal-cuidador) tipo tupla.
  - Métodos privados `EliminaCuidadorPorDNI`, `EliminaMascotaPorID` al que le llegará un dni y un id respectivamente y se encargarán de eliminar el cuidador o la mascota de las listas correspondientes.
  - Métodos públicos `AñadeCuidador`, `EliminaCuidador`. Añadirá un cuidador a la lista de **cuidadores** siempre que no se haya añadido antes (buscar por dni). Eliminará un cuidador de la lista de **asignaciones** buscando por dni y lo eliminará de esta lista si lo encuentra y llamará a `EliminaCuidadorPorDNI` para mantener la integridad referencial.
  - Métodos públicos `AñadeAnimal`, `EliminaAnimal`. Idénticos a los anteriores pero con las mascotas.
  - Método `AsignaAnimalACuidador()` que gestiona la relación entre animal y cuidador. Teniendo en cuenta los métodos creados en las anteriores clases para gestionar los cuidados de forma correcta.
  - Propiedad calculada `NumeroAsignaciones` para contar las asignaciones.
  - Constructores necesarios y método a `ACadena()` que devuelva una cadena con toda las información que se muestra en la salida.
  - **Importante!!** Se deberá gestionar correctamente la integridad referencial de las listas, es decir, si se elimina una mascota o cuidador, deberemos gestionar correctamente la lista de asignaciones.

Ejercicio 6. Relación de uso. Sistema de dibujo con

herramientas

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

```
Console.WriteLine("Ejercicio 6: Sistema de dibujo con herramientas");
Console.WriteLine();
Compas compas = new Compas();
Circulo circulo = compas.DibujaCirculo(3.5f);
Rotulador rotulador = Estuche.GetRotuladores()
                      [
                          new Random().Next(0, Estuche.NUMERO_ROTULADORES)
                      ];
rotulador.Rotula(circulo.Perimetro());
Pincel pincel = new Pincel();
pincel.Color= Color.Verde;
pincel.Pinta(circulo.Area());
Console.WriteLine("\n¡Dibujo completado con éxito!");
Console.WriteLine("Presiona cualquier tecla para salir...");
Console.ReadKey();
```

```
Ejercicio 6: Sistema de dibujo con herramientas

Dibujado un círculo de radio 3,5 cm
Rotulado el perímetro de 21,99 cm de color Negro.
Pintada el área de 38,48 cm² de color Verde.

¡Dibujo completado con éxito!
Presiona cualquier tecla para salir...
```

Requisitos:

- El circulo tendrá un atributo radio.
- El rotulador tendrá un atributo color de tipo enumerado y solo rotula perímetros. Al constructor de Rotulador le llegará el color como string.
- Habrá una clase estática Estuche con un solo método también estático que devolverá un array de rotuladores con colores creados de forma aleatoria.
- El pincel también tiene un atributo color y solo pinta áreas.