

Realizar una aplicación que gestione el alquiler de embarcaciones. La jerarquía de clases será la siguiente:

1. **Clase Barco**: Clase abstracta. *(Se facilita parte del código) implementa la interface IComparable. (1 punto)*
  - Atributos:
    - *String* **matricula**
    - *float* **eslora**
    - *float* **manga**
    - *boolean* **conPatron**
    - *boolean* **alquilado**
  - Métodos:
    - Constructor con parámetros: **Barco**(*String matricula, float eslora, float manga, boolean conPatron*)
    - Getters.
    - *float* **precioAlquiler()**, método abstracto.
    - *void* **verDatos()**, implementado.
2. **Clase Entrada**: para pedir datos por teclado. *(Se facilita el código).*
3. **Clase Cuenta**: similar a la utilizada en ejercicios anteriores. *(Se facilita el código).*
4. **Clase Lancha**: Clase hija de la clase **Barco**. *(0.5 punto)*
  - Atributos:
    - *float* **potencia**.
  - Métodos:
    - Constructor con parámetros, se añaden los atributos propios de la clase.
    - Getters.
    - **precioAlquiler**: Será la suma de:
      - Potencia.
      - Eslora.
      - Si tiene patrón, se le suma 50€.
    - **verDatos**: mostrará "Lancha", los datos del padre y los atributos propios.
5. **Clase Velero**: Clase hija de la clase **Barco**. *(0.5 puntos)*
  - Atributos:
    - *int* **numeroMastiles**.
    - *float* **superficieVela**.
  - Métodos:
    - Constructor con parámetros, se añaden los atributos propios de la clase.
    - Getters.
    - **precioAlquiler**: Será la suma de:
      - Eslora multiplicada por el número de mástiles.
      - Superficie de la vela.
      - Si tiene patrón, se le suma 120€.
    - **verDatos**: mostrará "Velero", los datos del padre y los atributos propios.
6. **Interface IComparable**: Esta interfaz define métodos para comparar objetos.
  - **menorQue**(otroObjeto: IComparable, opcion: int): tipo **boolean**. Devuelve **true** si el objeto que llama al método es menor que el objeto pasado por parámetro según el criterio de comparación especificado por la opción.
  - **mayorQue**(otroObjeto: IComparable, opcion: int): tipo **boolean**. Devuelve **true** si el objeto que llama al método es mayor que el objeto pasado por parámetro según el criterio de comparación especificado por la opción.
  - **igualQue**(otroObjeto: IComparable, opcion: int): tipo **boolean**. Devuelve **true** si el objeto que llama al método es igual al objeto pasado por parámetro según el criterio de comparación especificado por la opción.
7. **Interface IOrdenaArray**: Esta interfaz define métodos para ordenar un array.
  - **ordenaPorB**(opcion: int): Ordena el array por el método de la burbuja en función del criterio indicado por el parámetro.
  - **ordenaPorS**(opcion: int): Ordena el array por el método de selección en función del criterio indicado por el parámetro.
8. **Clase EmpresaAlquiler**: implementa la interface **IOrdenaArray**. *(2.5 puntos)*
  - Atributos:
    - *String* **nombre**

- *String* propietario
- *String* cif
- *Barco[]* embarcaciones
- *Cuenta* cuenta
- *int* numeroEmbarcaciones
- Métodos:
  - Constructor con parámetros. Los parámetros serán, nombre, propietario, cif y saldo de la cuenta. Creará el vector de tamaño 10, el número de embarcaciones lo inicializará a 0 y creará la cuenta con los datos propietario, cif y saldo.
  - Getters.
  - Setters: sólo el de propietario, no se podrá modificar por una cadena vacía.
  - **insertarEmbarcación**: de tipo *boolean*. Recibirá un objeto de la clase Barco y lo insertará en el vector, si puede.
  - **borrarEmbarcación**: de tipo *boolean*. Borrará del vector la embarcación de la posición que se le pase como parámetro. Para ello copiará el barco de la última posición rellena en esa posición y actualizará a null esa última posición, decrementando el atributo número de embarcaciones.
  - **buscarEmbarcación**: de tipo *int*. Recibirá como parámetro una matrícula y devolverá la posición en la que se encuentra, si existe o en caso contrario, **"-1"**.
  - **buscarEmbarcación**: de tipo *Barco*. Recibirá como parámetro la posición del vector y devolverá el objeto que hay en esa posición, o *null*, si la posición no es correcta.
  - **pedirDatosEmbarcación**: de tipo *Barco*, pedirá por teclado todos los datos de una nueva embarcación devolviendo un objeto de la clase *Barco* o null.
  - **verDatos**: de tipo *void*, mostrará el nombre, el propietario, el número de embarcaciones y el saldo de la cuenta.
  - **listarBarcos**: de tipo *void*, mostrará los datos de todas las embarcaciones.
  - **listarVeleros**: de tipo *void*, mostrará los datos de todos los veleros.
  - **listarLanchas**: de tipo *void*, mostrará los datos de todas las lanchas.

9. **Implementación de las interfaces:** (2 puntos) Se considerará el valor del parámetro opción en todos los métodos con los siguientes criterios:

- Si vale 1: ordena por las **matrículas** alfabéticamente en orden **ascendente**. Si vale -1, en orden **descendente**.
- Si vale 2: ordena por las **esloras** en orden **ascendente**. Si vale -2, en orden **descendente**.
- Si vale 3: ordena por los **precios de alquiler** en orden **ascendente**. Si vale -3, en orden **descendente**.

10. **Clase ExamenAbril24:** Será la clase de la aplicación.

- Métodos:
  - El método **main** tendrá un objeto de tipo *EmpresaAlquiler* que se construirá con unos datos concretos (sin pedir por teclado), siendo el nombre del propietario el del alumno que hace el examen. A través de un menú realizará las siguientes operaciones:

#### Alquiler de embarcaciones

1. Consultar datos de la empresa/¿Modificar propietario? (0.5 puntos)
2. Listado de embarcaciones: todas, veleros o lanchas. (0.5 puntos)
3. Insertar una nueva embarcación. (0.5 puntos)
4. Buscar una embarcación por su matrícula: ver datos y (¿borrarla?) (1 punto)
5. Ordenar las embarcaciones ascendentemente o descendentemente por: matrícula|eslora|precio de alquiler (1 punto)
0. Salir

Cada opción realizará lo siguiente:

1. Mostrará los datos de la empresa y preguntará si se quiere modificar el propietario.
2. Pedirá por teclado qué listado quiere mostrar y lo mostrará.
3. Pedirá los datos de la nueva embarcación creando el objeto e insertándolo en el array **embarcaciones**.
4. Pedirá la matrícula de la embarcación, la buscará mostrándola si existe. Preguntará si quiere borrarla.
5. Preguntará qué método de ordenación y qué criterio quiere ordenar las embarcaciones mostrando el resultado de esa ordenación.
0. Termina la ejecución.

Criterios de evaluación:

- Que funcione (debe compilar sin errores).
- Cumplimiento de los requisitos pedidos.
- Sintaxis, estructura y componentes de las clases.
- Nombres adecuados de objetos y variables.
- Uso de las estructuras de programación adecuadas.
- Uso correcto de métodos y parámetros.
- Comentarios de documentación
- En caso de detectarse *un examen copiado de otro*, **todos los alumnos implicados estarán suspensos**, aplicándoles las medidas disciplinarias oportunas.