## DISEÑO DE CLASES

He elegido un diseño de herencia con una padre clase abstracta (Barco), con un método abstracto (calcularFactorBarco). De esta manera, sus subclases (Velero, Deportiva y Yate De lujo) heredarán este método y lo implementarán de forma única cada uno.
El barco posee unos atributos que las subclases heredan y cada subclase tiene unos atributos únicos de subclase.

-Alquiler es una clase distinta, la cual su objetivo es calcular el precio final del amarre del barco.

-UT6Problema2 es la clase principal, la cual contiene las pruebas del sistema.

## CÁLCULOS BARCOS

Para el cálculo del factor barco, he creado un método abstracto calcularFactorBarco() que devuelve un double. En todos los barcos se tiene en cuenta el tamaño de la eslora en metros (10*eslora).

-En el caso particular del velero, también se utiliza el número de masteleros (10*eslora)+numMasteleros*5.

-El deportivo usa la potencia del motor como variable (10*eslora)+potencia/2.

-Finalmente, el yate obtiene la variable empleando la potencia del motor y el número de cabinas (10*eslora)+potencia/2+numCabinas*10.

La clase Alquiler utiliza este método para recoger la variable y multiplicarla por los días de alquiler, dando como resultado el precio final.
Como complemento, he creado un sistema de descuentos mediante ifs:
Si los días de alquiler son entre 5 y 10, se descontará un 5%, si son entre 10 y 15, un 10% y si son 15 o más, un 15%.

## VENTAJAS DE USAR HERENCIA Y POLIFORMISMO

Creo que este es un claro ejemplo donde es muy útil hacer una jerarquía hereditaria, ya que al utilizar atributos y métodos compartidos, no solo es más fácil de visualizar y mucho más compacto, sino también resulta una manera muy sencilla de ejecutar nuevos alquileres y barcos, haciendo de éste un sistema muy eficiente.