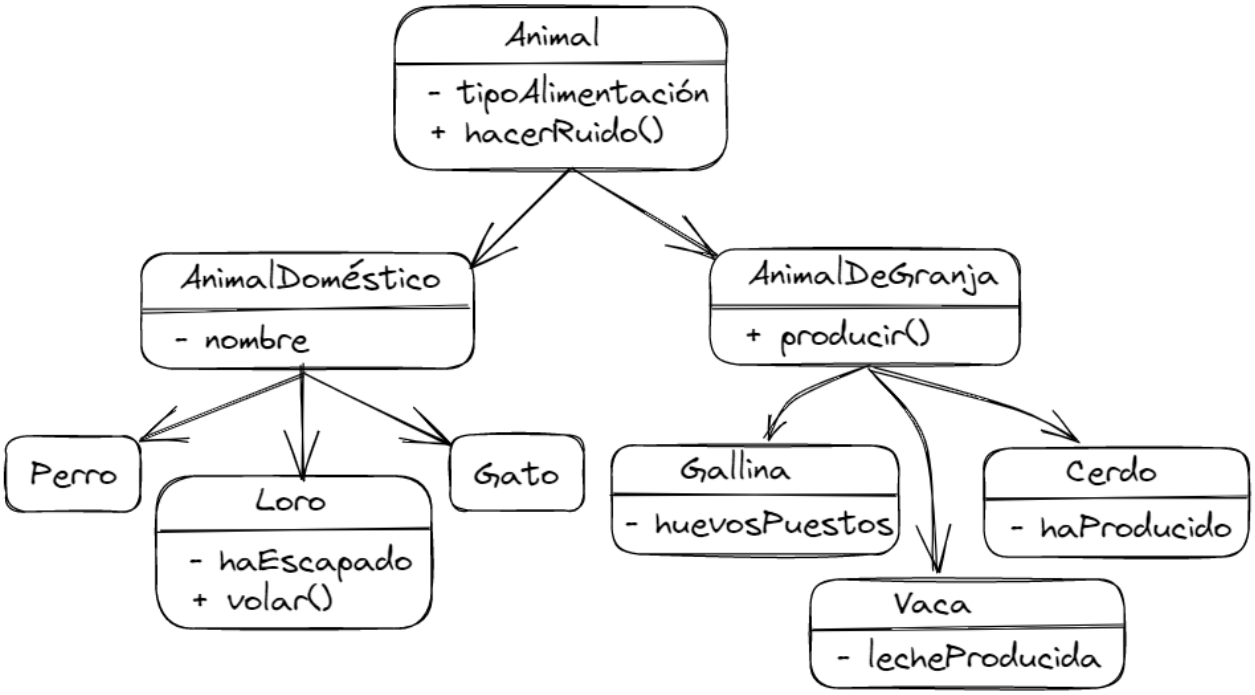
UD4 - Actividades 2 – Herencia

Actividad 0. Proyecto

a) Crea un proyecto con el nombre “ActividadesHerencia” para realizar todas las actividades de este documento. Recuerda mantener una estructura de paquetes adecuada y usar el modificador “protected” para que funcione la herencia de clase.

Actividad 1. Queremos crear una jerarquía de clases de Java de animales de distinta índole que cumpla con la estructura del siguiente diagrama:



a) Crea una clase para cada elemento del diagrama. Aquello señalado con “-” corresponde a atributos, mientras que los “+” corresponden a métodos.

b) En la clase Animal, implementa el método público “hacerRuido” para que muestre por pantalla “Un animal está haciendo ruido”.

c) En la clase AnimalDoméstico, añade un constructor que inicialice el nombre.

d) En la clase AnimalDeGranja, implementa el método público producir para que muestre por consola “Un animal de granja ha producido”.

e) Para las clases Loro y Cerdo, los booleanos “haEscapado” y “haProducido” son por defecto false. Para Gallina y Vaca, sus atributos enteros por defecto son cero.

f) Para todas las clases del final de la jerarquía (Perro, Gato, Loro, Gallina, Vaca y Cerdo), sobreescribe el método “hacerRuido” para que muestre por consola el sonido correspondiente a cada animal.

C/ MARÍA AUXILIADORA 14 | 38320 LA CUESTA (TENERIFE) | T 922 646 012 | WWW.SALESIANOS-LACUESTA.COM

g) Crea una clase ejecutable "MainAnimales" y crea un objeto de cada clase. Con cada objeto, llama al método "hacerRuido" y comprueba que funcionan correctamente.

h) En la clase Gallina, sobreescribe el método "producir" para que muestre por pantalla el mensaje "Una gallina ha puesto un huevo" y aumente una unidad al atributo "huevosPuestos".

i) En la clase Vaca, sobreescribe el método "producir" para que muestre por pantalla el mensaje "Una vaca ha dado leche" y aumente una unidad al atributo "lecheProducida".

j) En la clase Cerdo, sobreescribe el método producir para que muestre por pantalla el mensaje "Un cerdo ha producido" si no ha producido previamente. En caso contrario, muestra un mensaje "Lo siento, este cerdo ya ha producido".

k) Vuelve a la clase ejecutable y haz que los objetos de las clases de animales de granja produzcan varias veces para comprobar que todo funciona correctamente.

l) En la clase Loro, implementa el método "volar" para que muestre por pantalla el mensaje: "Tu loro <NOMBRE> ha salido volando…". Asegúrate de cambiar el valor del booleano que indica que ha escapado. Si ya ha escapado previamente, debe salir por pantalla: "¿Cuantas veces quieres que se escape <NOMBRE>?".

m) Ve a la clase ejecutable y haz que el objeto loro vuele.

n) Añade o modifica el constructor de todas las clases del final de la jerarquía (Perro, Gato, Loro, Gallina, Vaca y Cerdo) para que inicialicen el atributo tipoAlimentacion.

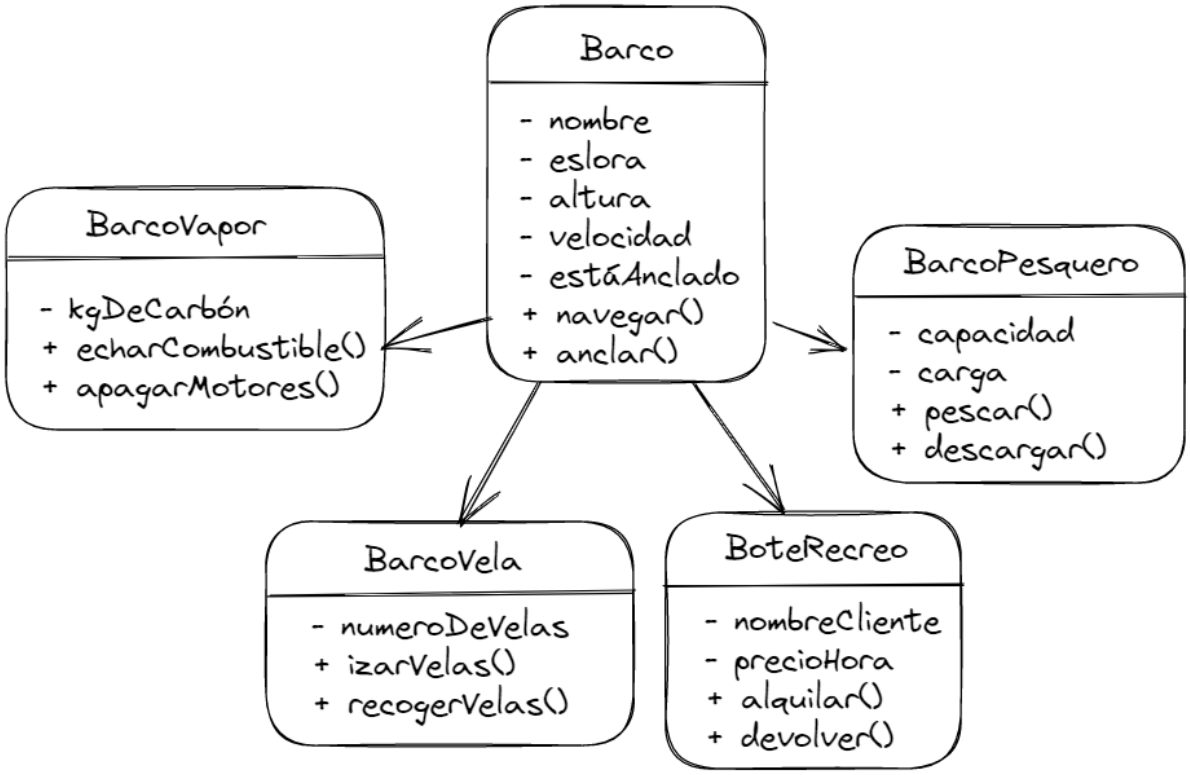
o) Finalmente, sobreescribe el método toString para los seis animales del final de la jerarquía de modo que devuelvan los mensajes como indica la siguiente tabla:

Animal	Mensaje
Perro	"Este perro se llama **<NOMBRE>** y se alimenta a base de **<ALIMENTACION>**"
Gato	"Este gato se llama **<NOMBRE>** y se alimenta a base de **<ALIMENTACION>**"
Loro	"Este loro se llama*/ba* **<NOMBRE>** y se alimenta*/ba* a base de **<ALIMENTACION>**"
Gallina	"Esta gallina ha producido **<N>** huevos"
Vaca	"Esta vaca ha producido **<N>** cubos de leche"
Cerdo	"Este cerdo ya/*no* ha producido"

p) Ve a la clase ejecutable y comprueba que el toString se ha sobreescrito correctamente para todos los objetos correspondientes.

C/ MARÍA AUXILIADORA 14 | 38320 LA CUESTA (TENERIFE) | T 922 646 012 | WWW.SALESIANOS-LACUESTA.COM

Actividad 2. Queremos crear una jerarquía de clases de Java de barcos de distinta índole que cumpla con la estructura del siguiente diagrama:



a) Define una clase padre **Barco** con los atributos y métodos que indica el diagrama cumpliendo las siguientes condiciones:
- El constructor debe inicializar el nombre, la eslora y la altura del barco.
- Por defecto, la velocidad será cero y el barco estará anclado.
- Los métodos deben ser públicos.
- Navegar hará que el barco no esté anclado y tome una velocidad crucero de 10 nudos por hora. Si el barco ya está navegando, debe mostrarse un mensaje por pantalla.
- El método anclar hará que el barco deje de moverse y cambiará el valor de estáAnclado. Si el barco estuviera ya anclado, debe mostrarse un mensaje.
- El barco no podrá anclar si la velocidad es mayor que la velocidad crucero, por lo que debe notificarse en un mensaje por consola.

b) Crea una clase hija **BarcoVapor** que extienda de Barco y cumpla con las siguientes condiciones:
- El constructor debe inicializar su único atributo y los de la clase antecesora.
- Siempre que quede carbón, echarCombustible hará que el barco consuma 10 kg de combustible y aumente su velocidad en 5 nudos por hora.
- El método apagarMotores hará que el barco vuelva a la velocidad crucero.

c) Crea una clase hija **BarcoVela** que extienda de Barco y cumpla con las siguientes condiciones:
- El constructor debe inicializar su único atributo y los de la clase antecesora.
- El método izarVelas aceptará un parámetro entero de intensidad del viento. La velocidad cambiará con el resultado del producto entre esta intensidad y el número de velas que posea el barco.
- El método recogerVelas hará que el barco vuelva a la velocidad crucero.
- Solo se podrá izar o recoger velas si el barco está navegando.

d) Crea una clase hija **BoteRecreo** que extienda de Barco y cumpla con las siguientes condiciones:
- El constructor debe inicializar sus atributos y los de la clase antecesora.
- El método alquilar debe aceptar un parámetro numero de horas y nombre de cliente para mostrar por consola un mensaje: “El barco <NOMBRE> ha sido alquilado a nombre de <CLIENTE> por <N> horas por un precio de <PRECIO> euros”.
- El método devolver debe mostrar por consola un mensaje: “El barco <NOMBRE> ha sido devuelto por <CLIENTE>”.
- Tanto para alquilar como para devolver, el barco debe estar anclado.

e) Crea una clase hija **BarcoPesquero** que extienda de Barco y cumpla con las siguientes condiciones:
- El constructor debe inicializar su único atributo y los de la clase antecesora.
- El método pescar aumentará la carga del barco una cantidad aleatoria entre 10 y 100 kg. Si se ha alcanzado la capacidad del barco, ya no se podrá guardar más pescado.
- El método descargar devuelve la carga a cero.
- El barco solo puede pescar cuando está navegando, mientras que solo puede descargar estando anclado.

f) Crea una clase ejecutable llamada “MainBarcos” e instancia un barco de cada tipo probando todas las funcionalidades que has implementado.

g) En cada una de las clases, sobreescribe el método toString para que retorne un mensaje especial para cada tipo de barco indicando toda su información actual.

h) Vuelve a la clase ejecutable y llama al método toString en cada uno de los objetos creados en el apartado f.

C/ MARÍA AUXILIADORA 14 | 38320 LA CUESTA (TENERIFE) | T 922 646 012 | WWW.SALESIANOS-LACUESTA.COM