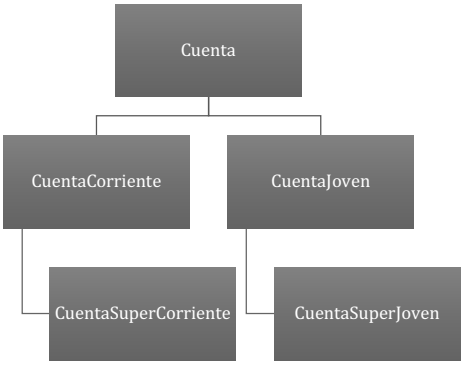
Parte práctica

PROGRAMACIÓN ORIENTADA A OBJETOS

1. BANCO

Crea la siguiente estructura de objetos para que funcione tal y como se indica en las especificaciones y reflexiona sobre las preguntas indicadas.



Especificaciones:

1. La cuenta tendrá tres atributos: numeroCuenta, saldo y contador. El contador se utilizará para almacenar el número de cuentas que se han creado.
2. La cuenta además tendrá un constructor por defecto y uno con tres parámetros. No tendrá métodos get y set de los atributos correspondientes. ¿Son necesarios?
3. Además, tendrá definido un método pagarIntereses que deberá ser redefinido en las clases derivadas.
4. No se podrán instanciar objetos de tipo Cuenta.
5. La clase CuentaCorriente será una clase derivada de la clase Cuenta y no deberá implementar el método pagarIntereses. ¿Qué ocurre si no lo implementas?
6. La clase CuentaJoven dispondrá del método pagarIntereses que incrementará el saldo de la cuenta en un 5%. ¿Puedes modificar el saldo con la información que tienes hasta ahora?
7. No podrán crearse clases derivadas de la clase CuentaJoven.
8. La clase CuentaMuyCorriente será una clase derivada de la clase CuentaCorriente.
9. La clase CuentaMuyJoven será una clase derivada de la clase CuentaJoven. ¿Qué ocurre cuando intentas crear esta clase?

***Nota**: Ejercicio resuelto en vídeo.*

2. BARCOS

Diseña un programa que considere una jerarquía de barcos. Todos ellos tienen un comportamiento en común: los métodos alarma() y mensajeSocorro().

Del barco de pasajeros se quiere guardar los metros de eslora y el número de camas.

Del portaaviones se desea almacenar el número de aviones y el número de marinos.

Del barco pesquero se guardan los metros de eslora, la potencia, el número de pescadores.

Cada vez que se cree un barco deberá mostrarse un mensaje indicándose los datos del barco creado.

El método alarma muestra un mensaje indicando desde qué tipo de barco se envía la alarma. Este método no podrá ser accedido desde fuera de la clase.

El método mensaje de socorro invoca al método alarma, y, además, muestra un mensaje a partir de una cadena que recibe por parámetro.

Completa los métodos correspondientes y crea un programa principal que cree barcos de diferentes tipos y ejecute los mensajes de socorro de cada uno.

3. CICLOS FP DE INFORMÁTICA

Diseña una estructura de clases que permita almacenar los ciclos de la familia de informática, teniendo en cuenta que se pueden clasificar según si son de FP Básica, Grado Medio o Grado Superior.

Incluye los atributos y métodos que creas necesarios (no olvides los constructores), y establece las relaciones que consideres para que la estructura sea lo más correcta posible: interfaces, clases o métodos asbtractos, clases o métodos estáticos, polimorfisco, clases finales, etc.

Te animamos a que, si lo deseas, comentes y/o compartas tu programa con el resto de compañeros.

4. PERSONA

Haz una clase llamada **Persona** que siga las siguientes condiciones:

- Sus atributos son: **nombre, edad, DNI, sexo** (H hombre, M mujer)**, peso y altura.** No queremos que se accedan directamente a ellos.
- Por defecto, todos los atributos menos el DNI serán valores por defecto según su tipo. Sexo será hombre por defecto: usa una constante para ello.

- Se implantarán varios constructores:
  - Un constructor por defecto.
  - Un constructor con el nombre, edad y sexo, el resto por defecto.
  - Un constructor con todos los atributos como parámetro.

- Los métodos que se implementaran son:
  - calcularIMC(): calculará si la persona está en su peso ideal (peso en kg/(altura^2 en m)), devuelve un -1 si está por debajo de su peso ideal, un 0 si está en su peso ideal y un 1 si tiene sobrepeso. Puedes utilizar constantes para devolver esos valores.
  - esMayorDeEdad(): indica si es mayor de edad.
  - comprobarSexo(char sexo): comprueba que el sexo introducido es correcto. Si no es correcto, se introducirá H. No será visible al exterior.
  - toString(): devuelve toda la información del objeto.
  - **Opcional**: generaDNI(): genera un numero aleatorio de 8 cifras, genera a partir de este su número su letra correspondiente. Este método será invocado cuando se construya el objeto. Puedes dividir el método para que te sea más fácil. No será visible al exterior.
  - Métodos set de cada parámetro, excepto de DNI.

A continuación, crea una clase ejecutable que haga lo siguiente:

- Pide por teclado el nombre, la edad, sexo, peso y altura.
- Crea 3 objetos de la clase anterior: el primero tendrá como atributos todos los introducidos por teclado; el segundo todos menos peso y altura, y el último los valores por defecto. En el caso del último, introduce los métodos set para asignar valores.
- Para cada objeto deberá comprobarse si está en su peso ideal, por encima o por debajo; si es mayor de edad; y la información del mismo.

5. SERIES Y VIDEOJUEGOS

Diseña una clase llamada Serie con las siguientes características:

- Atributos: título, numero de temporadas, entregado, género y creador.
- Por defecto, el número de temporadas es de 3 temporadas y entregado false. El resto de atributos serán valores por defecto según el tipo del atributo.

Crearemos una clase Videojuego con las siguientes características:

- Atributos: son título, horas estimadas, entregado, género y compañía.
- Por defecto, las horas estimadas serán de 10 horas y entregado false. El resto de atributos serán valores por defecto según el tipo del atributo.

Tanto Serie como Videojuego implementarán constructores y métodos:

- Los constructores que se implementarán serán:
  - Un constructor por defecto.
  - Un constructor con el título y creador en Serie, y con título y horas estimadas en Videojuego. El resto por defecto.
  - Un constructor con todos los atributos, excepto entregado.
- Los métodos que se implementara serán:
  - Métodos get de todos los atributos, excepto de entregado.
  - Métodos set de todos los atributos, excepto de entregado.
  - Sobrescribe los métodos toString.

Las clases anteriores no son base y derivada, pero como tienen varios métodos en común será necesario crear una interfaz llamada Entregable con los siguientes métodos que deberás implementar en cada clase (Videojuego y Serie):

- entregar(): cambia el atributo prestado a true.
- devolver(): cambia el atributo prestado a false.
- isEntregado(): devuelve el estado del atributo prestado.
- compareTo (Object a): compara las horas estimadas en los videojuegos y en las series el número de temporadas. Ten en cuenta que, al recibir un objeto, puede tratarse de una Serie (para las series) o de un Videojuego (para los videojuegos). Por tanto puedes hacer un casting de objetos utilizando el tipo de objeto entre paréntesis.

Crea una aplicación ejecutable y realiza lo siguiente:

- Crea dos arrays, uno de **Series** y otro de **Videojuegos**, de 5 posiciones cada uno.
- Crea un objeto en cada posición del array, con los valores que desees, puedes usar distintos constructores.
- Entrega algunos **Videojuegos** y **Series** con el método **entregar()**.
- Cuenta cuantos **Series** y **Videojuegos** hay entregados. Al contarlos, devuélvelos.
- Por último, indica el **Videojuego** tiene más horas estimadas y la serie con más temporadas. Muéstralos en pantalla con toda su información (usa el método toString()).