Guía de ejercicios POO - Unidad 6 - Interfaces y Clases Abstractas

1. Cree una jerarquía de herencia de Roedores: Raton, Gerbo, Hamster.
    a. En la clase base, proporcione los métodos que son comunes a todos los roedores (comer, olfatear, correr, trepar), y redefina aquellos en las clases derivadas para que tengan diferentes comportamientos dependiendo del tipo específico de roedor (con hacer print de “Estoy comiendo como un Hamster” o cualquier otra acción, es suficiente).
    b. Cree una lista con objetos hijos de Roedor, rellenelo con distintos tipos de roedores y llame a los métodos de la clase base para observar lo que ocurre (comer, olfatear, correr, trepar). Sacar una conclusión.
    c. Empezando con la jerarquía anterior de Roedor, herede un HamsterAzul de Hamster, sobreescriba los métodos de la clase base y muestre que el código que llama a los métodos de clase base no necesitan cambiar para adecuarse al nuevo tipo.
    d. Modifique Roedor para convertirlo en una clase base abstracta.

2. Crear una clase abstracta que se llame Figura que tenga:
    a. Los métodos get_nombre() , calcular_area(), calcular_perimetro()
    b. El método dibujar() que imprima en consola la figura. Por ejemplo: si ingresé un cuadrado de lado 3, deberá imprimir un cuadrado de 3 x 3
    c. Luego, crear clases concretas como Rectangulo y Cuadrado los cuales implementarán los métodos de las clases abstractas.
    d. Se deberán definir los atributos que requiera cada clase concreta para poder implementar la solución (buscar en internet formulas o consultar al profesor).

3. Continuando con el ejercicio anterior, agregaremos una clase concreta Perro que tenga un método dibujar() e imprima en consola la cara o silueta de un perro. (https://medium.com/analytics-vidhya/how-to-print-emojis-using-python-2e4f93443f7e).
    a. Ahora, se nos presenta un problema, tanto las Figuras como el Perro pueden dibujarse, entonces… Modifiquen el esquema de clases de tal forma que Perro herede de algún padre el método dibujar() y que las figuras puedan heredar de esa misma clase y también hereden los métodos correspondientes a Figura.

~~4. Define una clase abstracta Cuenta~~
    ~~a. Con los siguientes atributos:~~
        ~~- numerocuenta : entero largo~~
        ~~- saldo : double~~
        ~~- cliente : atributo de la clase Persona (que tiene nombres, apellidos, dni).~~
    ~~b. Con los siguientes métodos:~~
        ~~- Constructor que recibe como argumentos un objeto cliente y un número de cuenta~~
        ~~- ingresar(double): permitirá ingresar una cantidad en la cuenta.~~
        ~~- retirar(double): permitirá sacar una cantidad de la cuenta (si hay saldo). No se implementa, depende del tipo de cuenta~~
        ~~- actualizarSaldo(): actualizará el saldo de la cuenta, pero cada cuenta lo hace de una forma diferente~~
    ~~c. Define las subclases de Cuenta que se describen a continuación:~~
        ~~- CuentaCorriente: Cuenta normal con un interés fijo del 1.5%. Incluir constructor parametrizado y método __str__.~~
        ~~- CuentaAhorro: Esta cuenta tiene como atributos el interés variable a lo largo del año y un saldo mínimo necesario. Al retirar dinero hay que tener en cuenta que no se~~

~~sobrepase extraiga máyor saldo que el mínimo. Incluir constructor parametrizado, método __str__() y método para cambiar el interés.~~

~~d. Comprobar funcionamiento~~

5. Se trata de crear una pequeña base de datos de personas de una universidad. De momento definiremos y probaremos las siguientes clases:
   a. Direccion. Con los atributos: calle, ciudad, código postal, pais
   b. Persona: Clase ya creada (con nombre, apellidos y DNI, ver ejercicio anterior) a la que añadiremos el atributo dirección (del tipo o clase Direccion).

   c. Estudiante: Subclase de Persona.
      i. Atributos: ID de estudiante
      ii. Constructores : constructor parametrizado que admita pasar el ID como argumento.

   d. Profesor: Subclase de Persona.
      i. Atributos : despacho
      ii. Constructores: constructor que admita el despacho.

   e. La clase Persona, implementa la interface Humano, con un método indentificate(), que muestra el tipo de la clase que lo implementa (el tipo de persona, en este caso).

   f. Escribir programa main que corrobore el funcionamiento.

6. Definir los siguientes elementos:
   a. Interface Puerta : con los métodos abrir y cerrar.
   b. Interface PuertaBloqueable : derivada de Puerta, con los métodos bloquear() y desbloquear().
   c. Interface Alarma : con los métodos activarAlarma() y desactivarAlarma().
   d. Clase ComponentedeCoche: con los atributos descripción, peso y coste, y un método que muestre dichos atributos.
   e. Clase PuertaCoche, con el atributo boolean estaBloqueada, y que extienda la clase ComponentedeCoche e implemente las interfaces Alarma y PuertaBloqueable.
   f. Escribir programa main que pruebe la clase PuertaCoche