Ejercicios Herencia y Abstracción

Descargar estos ejercicios

Índice

- Ejercicio 1
- Ejercicio 2
- Ejercicio 3
- ✓ Ejercicio 4
- Ejercicio 5
- ✓ Ejercicio 6

Crea las clases silla, mesa y sofá hijas de la clase mueble del ejercicio 5 de la entrega anterior. A cada una de ellas le deberás de añadir los campos y métodos necesarios que identifiquen a dichos elementos según el texto del ejercicio 5.

Crea un método común para todos los muebles que nos permita calcular el precio del transporte según su peso, dimensiones y fabricante (puedes usar la fórmula que consideres para hacer este cálculo).

Finalmente, haz un programa principal que nos cree instancias de los tres tipos de muebles, que llame a los métodos de las clases para mostrar la información y que llame al método para calcular el precio del transporte y lo muestre.

Nota: Debes pensar en que clase o clases vas a incluir el método de calcular precio, para seguir las normas que se explican en el tema.

Se pide implementar usando POO y herencia las siguientes entidades:

- La clase Local :
 - Con los campos ciudad, calle, numero plantas y dimensiones.
 - Con la operación GetNumeroPlantas para poder acceder al atributo numeroPlantas.
 - o Con la operación ACadena que devolverá un string con los datos de un local.
- La clase LocalComercial que heredará de Local.
 - Con los campos razón social y número licencia.
 - Con la operación ACadena que devolverá un string con los datos de un local comercial.
- La clase Cine que heredará de LocalComercial.
 - o Con el campo aforo sala.
 - Con la operación ACadena que devolverá un string con los datos de un cine.

Deberemos implementar un programa principal que cree un array de 3 cines y lo inicialice con datos de supuestos cines. Después deberá mostrar por pantalla la información completa de cada uno de los cines mediante un foreach, a través del método ACadena.

Nota: Deberás soobreescribir el método ACadena, para poder aprovechar el código implementado en las clases padre.

Para dar nuestros primeros pasos con el polimorfismo de datos del principio de sustitución de Liskov, vamos a recordar el cálculo de factorial y de número primo.

Vamos a necesitar una clase padre ElNoCalculador con un atributo

protected short numero, que será la base del calculo posterior, también tendrá dos métodos virtuales Factorial y Primo devolverán un double y un booleano respectivamente

(en esta clase no se realizarán los cálculos, los métodos devolverán 0 y false).

Además, crearemos una clase hija **ElCalculador** que sí implementa estos métodos de forma correcta a partir del numero del padre.

Para probar el funcionamiento del polimorfismo nos crearemos un objeto de la manera:

ElNoCalculador obj = new ElCalculador(num);

y llamaremos a los métodos Primo y Factorial.

Reflexiona las siguiente preguntas:

- 1. ¿A qué métodos se llama, a los de la clase padre o a los de la clase hija?
- 2. Crea en la clase hija un método MostrarResultado y llámalo con el objeto. ¿Qué ocurre?
- 3. y si comentas el método Primo de la hija. ¿Qué ocurre?



Ejercicio 4

Crea una clase Guerrero que herede de la clase Humano del ejercicio3 del tema anterior, añade los campos tipoArma y tipoArmadura y los métodos que creas necesarios, para hacerla funcional.

Crea también la clase Mago hija de humano con los atributos tipoLibroHechizos y tipoTúnica y los métodos necesarios.

Nota: Sería interesante que utilices tipos enumerados para los campos anteriormente descritos.

Declara una clase abstracta **Legislador** que herede de la clase **Persona**, que usamos en ejercicios anteriores, con un campo **povinciaQueRepresenta** tipo **string**, y si quieres otros atributos que creas necesarios.

Como puedes ver en el diagrama, la clase **Legislador** hereda de la clase **Persona** que no es abstracta, lo cual significa que puede haber instancias de **Persona** pero no de **Legislador**.

Declara un método abstracto **GetCamaraEnQueTrabaja** (devolverá un string indicando la cámara a la que pertenece), ToString (devolverá un string con los datos a mostrar de la clase incluida la provincia que representa).

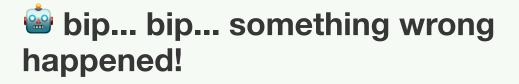
Crea las siguientes clases que heredarán de Legislador e implementarán los métodos necesarios:

- Diputado
- Senador

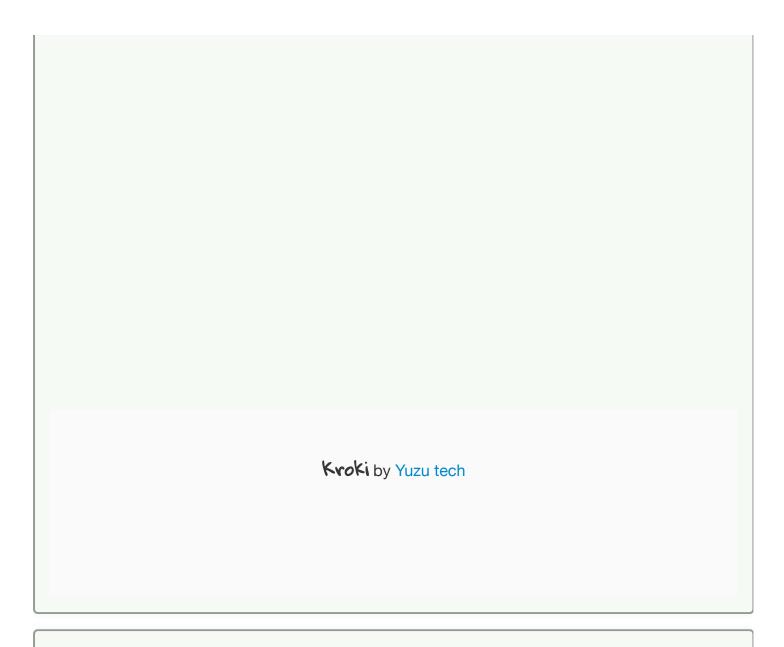
Crea un **array de tres legisladores** de distintos tipos, usa **polimorfismo de datos** y muestra por pantalla la cámara en que trabajan y otros datos que creas necesarios.



Creates diagrams from textual descriptions!



The page you are looking for is temporarily unavailable. Please try again later.





Crear una clase llamada **TablaEnteros** que **no permita** que se creen objetos a partir de ella. Esta clase almacenará una tabla de enteros **de una dimensión** y que será un atributo protegido, el tamaño debe ser especificado mediante su constructor.

La clase debe obligar a que cualquier clase que herede de ella y no quiera ser una clase abstracta, implemente un método llamado GuardarNumerosenTabla.

Además, TablaEnteros dispondrá de dos métodos:

• **DevuelveTabla**, método **redefinible**, que servirá para devolver la cadena con el contenido completo de la tabla.

• **SumaPropia** que se encargará de comprobar si existen más números positivos o negativos en la tabla y devolverá la suma de aquellos de los que hay mayor cantidad.

Crear a partir de esta clase dos nuevas clases llamadas:

- TablaImpares : que solo guardará números impares en el atributo del padre.
- TablaPares : que solo guardará números pares, el el atributo del padre.

Ambas, lo harán mediante el método **GuardarNumerosEnTabla**, antes mencionado, que seleccionará los números apropiados (pares o impares) a guardar a partir de número generados aleatoriamente.

En el programa principal, crea instancias de cada una de estas clases, dales valores y muestra las tablas y la suma propia de ambos objetos por pantalla.