Ejercicios UT7 –Utilización avanzada de clases.

Programas en los que aparece la composición/agregación vs la asociación.

1. La competencia en materia de impuestos sobre vehículos de tracción mecánica pertenece en la actualidad a los ayuntamientos. Con objeto de gestionar el cobro de dicho impuesto, el ayuntamiento de "Elizabethtown" (Kentucky) dispone de la información de las personas empadronadas en él (TIN, First name, Family name, U.S. Street Address, City, State, Phone number y Email) y de los vehículos sujetos al impuesto (VIN, Year, Make, Model, Engine y Fuel). Desarrolle las clases necesarias para gestionar el cobro durante ejercicio tributario actual utilizando un modelo relacional y su alternativa utilizando un modelo orientado a objetos.

2. Incorpore en el diseño anterior la posibilidad de gestionar el cobro durante cualquier ejercicio tributario.

3. Los vehículos de la policía nacional denominados coches patrulla o vehículos tipo Z tienen en la parte superior de su carrocería un código formado por tres caracteres (letras y dígitos) que permite, por su ubicación y dimensiones, una fácil identificación desde localizaciones elevadas. En cada turno, el vehículo es asignado a dos funcionarios de la brigada de seguridad ciudadana identificados cada uno de ellos por su número de placa. En la actualidad existen tres turnos (7:00 a 15:00, 15:00-23:00 y 23:00 a 7:00). Desarrolle un modelo de datos y un programa que permita conocer quienes ocupaban un vehículo a partir de una fecha y hora concretas. No es necesario que programe las operaciones CRUD de las clases que identifique.

Programas en los que aparece la herencia simple.

4. Programe las clases Persona y Trabajador de acuerdo con las especificaciones:

   Persona
   - Atributos privados (nombre, teléfono y edad).
   - Constructor por defecto.
   - Constructor con 3 parámetros para las 3 variables de instancia.
   - Setters y getters.
   - Método toString.

   Trabajador (subclase de persona):
   - Atributos: categoría profesional (A, B ó C), Antigüedad (int).
   - Constructores: por defecto y con parámetros, incluso los de la clase Persona.
   - Métodos setters y getters.
   - Método toString.

   Desarrolle una clase de prueba que contendrá el método main, en la que se creen dos objetos de la clase Trabajador. El primero se instanciará con el constructor por defecto y posteriormente se le introducirá el contenido a sus atributos utilizando los correspondientes métodos setters. El segundo se instanciará utilizando el constructor que utiliza todos los parámetros y al que se le pasará como valores la información leída

desde teclado. A estos objetos de la clase Trabajador, también se les dará valores a sus atributos nombre, teléfono y edad que heredan de la clase Persona.

Visualice su contenido utilizando el método toString() que ha programado y visualice el nombre del trabajador de más antigüedad.

5. Crear un programa para gestionar el inventario de una tienda de mascotas. De cada mascota se deberá guardar nombre y fechaNacimiento. Habrá 3 tipos de mascotas: perros, gatos y loros. La configuración de las clases será la siguiente:

   ● De los perros se almacenan los atributos raza y pulgas. Y deberá tener un método que sea emiteSonido que indicará el sonido que hace el perro. Además, será posible mostrar todas sus características.

   ● De los gatos se almacenarán los atributos peloLargo y color. Y deberá tener un método que sea emiteSonido. Además será posible mostrar todas sus características.

   ● De los loros se necesitan los atributos origen y habla. Y deberá tener un método que sea emiteSonido, otro que sea volar y otro saluda. Además, será posible mostrar todas sus características.

   ● Crear una clase principal en la que se creen distintas instancias de las mascotas y se llamen a sus distintos métodos.

6. Escriba un programa para una biblioteca que contenga libros y revistas. Las características comunes que se almacenan tanto para las revistas como para los libros son el código, el título, el año de publicación y el número de páginas. Los libros tienen, además, un atributo autor. Las revistas tienen, además, un número de revista. Tanto las revistas como los libros deben tener (además de los constructores) los métodos getter y setters y el método toString.

Programas en los que aparece la herencia simple y el polimorfismo.

7. Escriba una superclase llamada Electrodomestico con las siguientes características:

   Sus atributos son precio base, color, consumo energético (letras entre A y F) y peso. Por defecto, el color será blanco, el consumo energético será F, el precioBase será de 100 € y el peso de 5 kg. Use constantes para ello. Los colores disponibles serán blanco, negro, rojo, azul y gris. No importa si el nombre está en mayúsculas o en minúsculas.

   Los constructores que se implementaran serán un constructor por defecto, un constructor con el precio, el peso y el resto de atributos con sus valores por defecto y un constructor con todos los atributos.

   Los métodos que implementara serán los métodos getter de todos los atributos. El método comprobarConsumoEnergetico(char letra) que comprobará que la letra es correcta, de manera que si no es correcta usará la letra por defecto. Se invocará al crear el objeto y no será visible. El método comprobarColor(String color) que comprobará que el color es correcto, si no lo es utilizará el color por defecto. Se invocará al crear el objeto y no será visible. El método precioFinal() que incrementará el precio según el consumo energético y según su tamaño de acuerdo con las tablas siguientes:

Letra	Precio
A	100€
B	80€
C	60€
D	50€
E	30e
F	10€

Tamaño	Precio
Entre 0 y 19 kg	10€
Entre 20 y 49 kg	50€
Entre 50 y 79 kg	80€
Mayor o igual a 80 kg	100€

Crear una subclase llamada Lavadora con el atributo carga junto con los atributos heredados. Por defecto, la carga será de 5 kg. Use una constante para ello. Los constructores que se implementarán serán un constructor por defecto, un constructor con el precio, el peso y el resto por defecto y un constructor con la carga y el resto de atributos heredados. Recuerde que debe llamar al constructor de la clase padre.

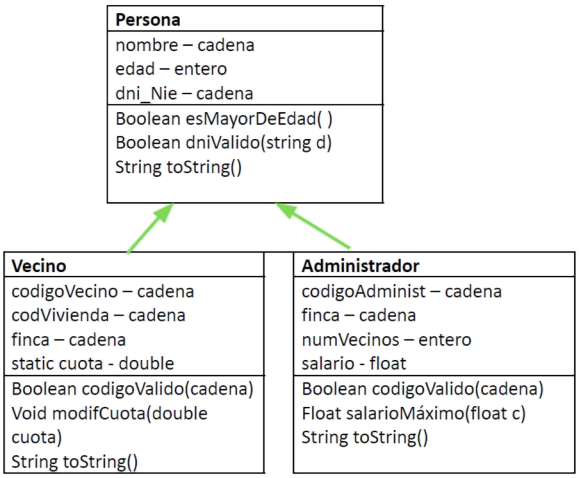
Los métodos que se implementara serán el método get de carga, el método precioFinal(), el cual, si tiene una carga mayor de 30 kg, aumentara el precio en 50 € y si no es así no se incrementara el precio. Invoque al método padre y añada el código necesario. Recuerde que las condiciones que se han visto en la clase Electrodomestico también deben afectar al precio.

Cree una subclase llamada Television con los atributos resolución (en pulgadas) y sintonizador TDT (booleano) además de los atributos heredados. Por defecto, la resolución será de 20 pulgadas y el sintonizador será false. Los constructores que se implementaran serán u constructor por defecto, un constructor con el precio, el peso y el resto por defecto y un constructor con la resolución, sintonizador TDT y el resto de atributos heredados. Recuerde que debe llamar al constructor de la clase padre.

Los métodos que se implementarán serán el método get de resolución y sintonizador TDT. El método precioFinal() de manera que si tiene una resolución mayor de 40 pulgadas se incrementará el precio un 30% y si tiene un sintonizador TDT incorporado aumentará 50€. Recuerde que las condiciones que se han visto en la clase Electrodomestico también deben afectar al precio.

Incorpore una clase Main que cree un array de Electrodomesticos de 10 elementos. Asigne a cada posición un objeto de las clases anteriores con los valores que desee. Recorra este array y ejecute el método precioFinal(). Deberá mostrar el precio de cada clase, es decir, el precio de todas las televisiones por un lado, el de las lavadoras por otro y la suma de los Electrodomesticos (puede crear objetos Electrodomestico, pero recuerde que Television y Lavadora también son electrodomésticos). Busque o pregunte sobre el operador Java *instanceof*. Por ejemplo, si tenemos un Electrodomestico con un precio final de 300, una lavadora de 200 y una televisión de 500, el resultado final será de 1000 (300+200+500) para electrodomésticos, 200 para lavadora y 500 para televisión.

8. Dado el siguiente diagrama de clases UML:



Desarrolle los constructores por defecto y con los valores de todos los atributos. En el caso de la clase Persona incluya un constructor que no contenga el DNI.

Desarrolle los métodos siguientes:

- esMayorDeEdad – comprueba si el atributo edad es mayor o igual que 18. Devuelve true o false.
- dniNieValido – Comprueba si el atributo DNI_NIE tiene un contenido válido, devuelve true si es así o false si no. Todos los DNI deben tener los 8 dígitos numéricos y las letras que correspondan. El formato del NIE consta de una letra inicial, siete números y un código de control (puede ser una letra o un número). Se adjunta un enlace donde se explica cómo se calcula el dígito de control: https://www.interior.gob.es/opencms/es/servicios-al-ciudadano/tramites-y-gestiones/dni/calculo-del-digito-de-control-del-nif-nie/
- codigoValido en Vecino – máximo 5 caracteres y debe comenzar por la letra V.
- codigoValido en Administrador – máx 5 caracteres. Debe comenzar por la letra A.
- actualizarCuota : método que recibe un valor double que utiliza para modificar el valor del atributo cuota que es static, es decir, el nuevo valor tendrá validez para todos los objetos de la clase vecino. No devuelve nada.
- salarioMáximo debe calcular el importe máximo del salario del administrador que será igual al 50% del numVecinos*cuota (c).
- toString() en las 3 clases.

Programe un método main() que haga lo siguiente:

- Instancia un objeto Vecino con edad 16 años y el resto de los valores inventados. (utilice el constructor que no incorpora el DNI). A continuación pide el valor de DNI y programa un bucle que se repita mientras el DNI introducido no sea un DNI válido. Utilice el método dniValido(). Cuando salga del bucle con un DNI válido le asignará al atributo DNI del objeto creado anteriormente el valor introducido por teclado.
- Compruebe si el vecino es mayor de edad y si no lo es sustituye edad por 18.

- Instancie un objeto Administrador con edad 16 años y el resto de los valores inventados. (utilice los constructores)
- Compruebe si es mayor de edad y si no lo es sustituye edad por 18.
- Compruebe si el salario que ha asignado al objeto es mayor que el salario máximo que nos calcula el método salarioMáximo(). A este método le pasará como parámetro un valor para cuota. Esta cuota la obtendrá del objeto de la clase Vecino, es uno de sus atributos. Si salario es mayor que salarioMáximo sustituya el atributo salario del objeto Administrador por el valor devuelto por la función.
- Utilice el método toString() de las 3 clases. Observe la diferencia de los resultados.

9. Desarrolle una aplicación que permita la Gestión de las nóminas de una empresa.

   En la empresa existen varios tipos de Trabajadores. De todos ellos nos interesa el dni, nombre, salario base y el salario final, que se calcula en función del tipo de trabajador y de los complementos. De manera que salario final es el salario base más un complemento.

   Los trabajadores se dividen en dos grupos: informáticos y gestión. De los informáticos nos interesa aparte de los datos citados, la titulación. Hay dos tipos de informáticos: Analistas: el complemento es del 30% y Programadores: el complemento es del 15%. Del personal de gestión nos interesa la antigüedad en la empresa. Hay dos tipos de personal de gestión: Administrativos: el complemento es fijo y supone 20€ por cada año de antigüedad y Auxiliares: el complemento es fijo de 100€ independiente de la antigüedad. Recuerde que el salario final es el salario base más un complemento. Por lo tanto, desarrolle:

   - La clase Trabajador con los atributos vistos anteriormente y sus clases derivadas Informático, Gestión, Analista, Programador, Administrativo y Auxiliar.
   - La clase Empresa con CIF y un nombre. También tendrá un array de Trabajadores. Y el método necesario para añadir los trabajadores a la empresa.
   - La clase Aplicación que hará lo siguiente:
     - Dará de alta una empresa (no hace falta pedir datos al usuario, crearlo directamente)
     - Añadir 2 trabajadores de cada tipo e introducirlos en la empresa que se ha creado previamente (no hace falta pedir datos al usuario, crearlo directamente).
     - Mostrar un listado de Trabajadores junto con su salarioBruto y salarioFinal. En el listado deben aparecer los trabajadores junto con su tipo al inicio:

Analista: {dni=204433963W, nombre=Carles, salarioB=2800.0, salarioF=3640.0}
Analista: {dni=204437956R, nombre=Maria, salarioB=3000.0, salarioF=3900.0}
Programador: {dni=204437963W, nombre=Alba, salarioB=1100.0, salarioF=1265.0}
Programador: {dni=204437964W, nombre=Marti, salarioB=1200.0, salarioF=1380.0}
Administrativo: {dni=20443300A, nombre=Nuria, salarioB=1300.0, salarioF=1400.0}
Administrativo: {dni=20443301A, nombre=Pepe, salarioB=1400.0, salarioF=1600.0}
Auxiliar: {dni=20443302A, nombre=Jose, salarioB=1000.0, salarioF=1100.0}
Auxiliar: {dni=20443303A, nombre=Ana, salarioB=1100.0, salarioF=1200.0}

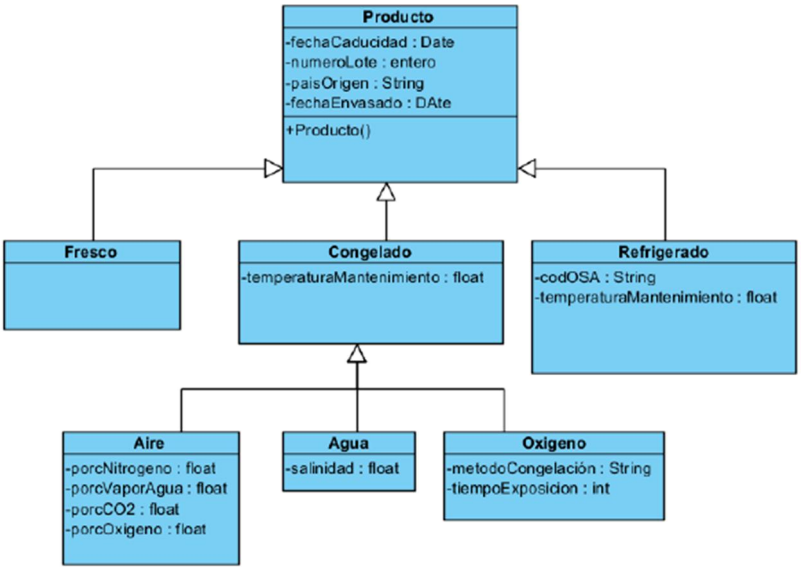
10. Se plantea desarrollar un programa Java que permita la gestión de una empresa agroalimentaria que trabaja con tres tipos de productos: productos frescos, productos refrigerados y productos congelados. Todos los productos llevan esta información común: fecha de caducidad y número de lote. A su vez, cada tipo de producto lleva alguna información específica. Los productos frescos deben llevar la fecha de envasado y el país de origen. Los productos refrigerados deben llevar el código del organismo de supervisión alimentaria, la fecha de envasado, la temperatura de mantenimiento recomendada y el país de origen. Los productos congelados deben llevar la fecha de envasado, el país de origen y la temperatura de mantenimiento recomendada

    Hay tres tipos de productos congelados: congelados por aire, congelados por agua, congelados por nitrógeno. Los productos congelados por aire deben llevar la información de la composición del aire con que fue congelado (% de nitrógeno, % de oxígeno, % de dióxido de carbono y % de vapor de agua). Los productos congelados por agua deben llevar la información de la salinidad del agua con que se realizó la congelación en gramos de sal por litro de agua. Los productos congelados por nitrógeno deben llevar la información del método de congelación empleado y del tiempo de exposición al nitrógeno expresada en segundos.

    Crear el código de las clases Java implementando una relación de herencia siguiendo estas indicaciones:

    a) En primer lugar, realizar un esquema con papel y bolígrafo (o una herramienta tipo VisualParadigm o similar) donde se represente cómo se van a organizar las clases cuando escriba el código. Estudiar los atributos de las clases y trasladar a la superclase todo atributo que pueda ser trasladado.
    b) Crear superclases intermedias (aunque no se correspondan con la descripción dada de la empresa) para agrupar atributos y métodos cuando sea posible. Esto corresponde a “realizar abstracciones” en el ámbito de la programación, que pueden o no corresponderse con el mundo real.
    c) Cada clase debe disponer de constructor y permitir establecer (set) y recuperar (get) el valor de sus atributos y tener un método que permita mostrar la información del objeto cuando sea procedente.

    Pista:



Crear una clase testHerencia con el método main donde se creen: 2 productos frescos, 3 productos refrigerados, 5 productos congelados (2 de ellos congelados por agua, otros 2 por aire y 1 por nitrógeno). Mostrar la información de cada producto por pantalla.

A partir de la jerarquía de clases programada anteriormente, de la Empresa Alimentaria, programe 4 métodos descripción() en las siguientes clases y con las siguientes características.

Los métodos devolverán una cadena determinada a partir de los atributos de los objetos que las invocan y consistentes en lo siguiente (X: representa un carácter y 9: representa un dígito numérico)

En la clase Congelados:
CONGELADO
Nº de Lote:..... XXXXXX
Temperatura de Mantenimiento: 99,9

En la clase Aire:
CONGELADO POR AIRE
Nº de Lote:..... XXXXXX
Temperatura de Mantenimiento: 99,9
Porcentaje Vapor Agua:..... 99,9
Porcentaje Nitrógeno: 99,9
Porcentaje CO2:..... 99,9
Porcentaje Oxigeno: 99,9

En la clase Agua:
CONGELADO POR AGUA
Nº de Lote:..... XXXXXX
Temperatura de Mantenimiento: 99,9
Salinidad: 99,9
En la clase Oxigeno:
CONGELADO POR OXÍGENO
Nº de Lote..... : XXXXXX
Temperatura de Mantenimiento: 99,9
Método de Congelación: 99,9
Tiempo de Exposición:..... 9999

Finalmente, en la clase controladora TestHerencia que contiene el método main(), programe un vector de objetos Congelados y almacene en él un objeto de la clase Congelado y los objetos de las clases Agua, Aire, Oxigeno. Recorra el vector y muestre las cadenas que nos devuelvan los diferentes métodos.

Programas que incorporan clases abstractas.

11. En este ejercicio se trata de crear una aplicación de gestión de personal de una empresa. Del personal se conocen: DNI, Nombre, Apellidos y Año de alta en la empresa. Hay dos tipos de empleados:
    - Empleados con salario fijo: estos empleados tienen un sueldo base y un porcentaje adicional en función del número de años que lleven en la empresa con los siguientes cálculos: Menos de 2 años solo tienen salario base. Entre 2 y 3 años un 7% del salario base. Entre 4 y 8 años un 11% del salario base y entre 9 y 15 años un 17% del salario base.

    • Empleados a comisión: estos empleados tendrán todos un salario fijo de 950 €, un número de clientes captados y una comisión por cada cliente captado. El sueldo es el resultado de multiplicar el número de clientes captados por la comisión por cliente. Si el resultado de este cálculo no llega al salario fijo, se cobra solo el salario fijo.

Requisitos de la aplicación:

- Clase padre Empleado que no se podrá instanciar.
- Clases EmpleadoAsalariado y EmpleadoComision que heredarán de Empleado.

Todas las clases deberán tener un constructor con parámetros y un constructor por defecto. La clase Empleado tendrá un método imprimir() y un método abstracto getSalario(). Clase principal donde se creará un array en el método main con los siguientes datos:

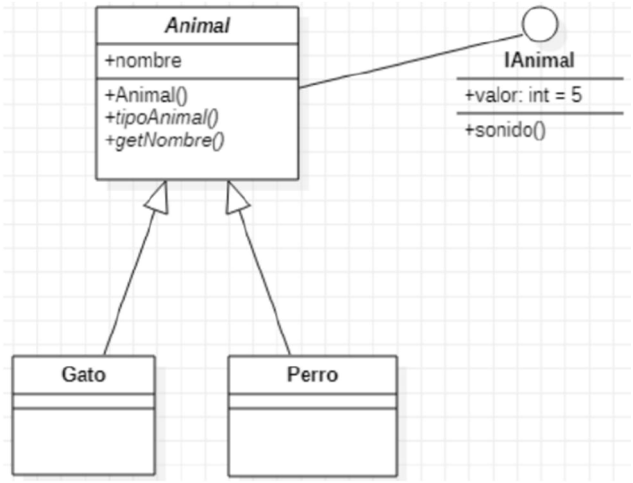
- Antonio López. DNI: 6546546Z, año de alta: 2014, salario fijo: 1125
- Sandra Nieto. DNI: 7879879S, año de alta: 2011, 169 clientes captados a 7.10 cada uno.
- Manuel Ruíz. DNI: 4654654D, año de alta: 2010, 35 clientes captados a 6.90 cada uno.
- María Ramos. DNI: 77879878F, año de alta: 2011, salario fijo: 1055

Los dos primeros empleados se crearán utilizando el constructor con parámetros. Los dos últimos con el constructor por defecto y los getters y setters correspondientes.

Desde el main de la clase principal se llamarán a dos métodos: sueldoMayor(). Este método recibirá un array de tipo Empleado por parámetros y devolverá el nombre apellido y salario del Empleado con el salario más alto. mostrarTodos(). Este método recibirá por parámetro un array de tipo Empleado y mostrará los datos de todos los Empleados del array.

Programas con interfaces.

12. Desarrolle el siguiente diagrama de clases:



La interface IAnimal es una interfaz que posee el método sonido() y el atributo valor el cual (al ser declarado en la interface) se comporta como una constante.

La clase abstracta Animal implementa la interface IAnimal y al mismo tiempo es clase Padre de las clases Gato y Perro. Incorpora los métodos String getNombre() y void tipoAnimal() siendo este último un método abstracto.

Las clases Gato y Perro heredan de la clase abstracta Animal, por lo tanto implementan el método tipoAnimal y como Animal que son implementan la interface IAnimal, por lo que están obligadas a implementar el método sonido().

Para probar complete la clase añadiendo en cada constructor un mensaje específico indicando que ha entrado en el constructor y su nombre. Cree un array de animales el cual contenga varios perros y gatos. Recorra el array mostrando el mensaje correspondiente del tipo de animal que es.