Ejercicios Tema 11. Herencia y Polimorfismo

*Todos los ejercicios de este boletín deben estar dentro de un **package** llamado **tema11** (siguiendo las reglas que hemos visto en clase, ej: com.germangascon.tema11).*

1. Escribe una clase **Punto** que tenga:
   a) Dos atributos (x e y) de tipo double para representar la posición del punto.
   b) Un constructor con dos parámetros de tipo double que inicialice los dos atributos.
   c) Un constructor por defecto (sin parámetros) que inicialice los dos atributos al valor 0.
   d) Tenga un método distancia() que reciba un parámetro de tipo Punto y devuelva un double.
   e) Un método toString() que devuelva la información del Punto de la siguiente manera (x,y).

2. Utilizando la clase **Punto** del ejercicio anterior, escribe una clase **Poligono**. Esta clase tendrá como atributo una lista de objetos Punto. Se consideran adyacentes los objetos Punto que están en posiciones consecutivas de la lista y los puntos que están en la primera y última posició. Esta clase debe tener los siguientes métodos:
   a) Un constructor, que tendrá como parámetro la lista con los objetos Punto que definen el polígono.
   b) traslada(), que recibe como parámetro el desplazamiento en la coordenada x y el desplazamiento en la coordenada y.
   c) numVertices(): devuelve el número de vértices del polígono.
   d) toString(): devuelve la información de los puntos del polígono, cada uno en una línea.
   e) perimetro():devuelve el perímetro del polígono.

   Escribe una aplicación que:
   - Cree un Polígono de cuatro vértices, que serán (0,0), (2,0), (2,2), (0,2).
   - Muestre la información del polígono y su perímetro.
   - Traslade el polígono 4 unidades en el eje x y -3 en el eje y.
   - Y finalmente vuelva a mostrar la información del polígono por pantalla.

3. Escribe una clase **Coche** de la cual heredarán **CocheCambioManual** y **CocheCambioAutomatico**.
   a) Los atributos de los coches son la matrícula, la velocidad actual, la marcha actual y un array de enteros de N posiciones (donde N es el número de marchas del coche) que indica la velocidad máxima que se puede alcanzar con cada marcha. Para esta actividad no se permite la marcha

hacia atrás, por lo tanto no se permite ni velocidad negativa ni marcha negativa. El constructor recibe el valor de la matrícula por parámetro y se inicializa el valor de la velocidad y la marcha a 0. Además, deberá tener los siguientes métodos:

- getters y setters para las variables de instancia
- acelerar(): recibe por parámetro el valor con el cual se quiere acelerar el coche.
- frenar(): recibe por parámetro un valor con el cual frenar el coche.
- toString(): devuelve en forma de String la información del coche.
- cambiarMarcha(): recibe por parámetro la marcha a la cual se tiene que cambiar el coche. Este método se tendrá que definir de tal forma que puedan acceder a él las clases que heredan de Coche, pero no las clases otros packages.

b) La clase CocheCambioManual sobrescribirá el método cambiarMarcha() para que pueda ser invocado desde cualquier clase. No se permite que se cambie a una marcha negativa.

c) La clase CocheCambioAutomatico sobrescribirá los métodos acelerar() y frenar() para que cambie automáticamente de marcha a medida que se va acelerando y frenando.

d) Escribe una aplicación que solicite por teclado la matrícula de un coche y pregunte si el coche es con cambio automático o no. Posteriormente, tiene que crear un coche con las características indicadas por el usuario y mostrarlo. Acelerar el coche hasta 60 km/h, si es un coche con cambio manual, cambiar la marcha a tercera y volver a mostrar la información.

4. Crea una superclase llamada **Electrodomestico** con las siguientes características:

a) Sus atributos son precioBase, color, consumo energético (letras entre A y F) y peso. Indica que se podrán heredar.

b) Por defecto, el color será blanco, el consumo energético será F, el precioBase será de 100 € y el peso de 5 kg. Usa constantes para ello.

c) Los únicos colores disponibles son blanco, negro, rojo, azul y gris.

d) Los constructores que deben implementarse són:

- Un constructor por defecto.
- Un constructor con el precio y peso. El resto por defecto.
- Un constructor con todos los atributos.

e) Los métodos que deben implementarse son:

- Métodos get de todos los atributos.
- comprobarConsumoEnergetico(char letra): comprueba que la letra es correcta, sino es correcta utiliza la letra por defecto. Se invocará al crear el objeto y no será visible.

- comprobarColor(String color): comprueba que el color es correcto, si no lo es, usa el color por defecto. Se invocará al crear el objeto y no será visible.
- precioFinal(): según el consumo energético, aumenta el precio, y según su tamaño, también.

Esta es la lista de precios:

LETRA	PRECIO
A	100 €
B	80 €
C	60 €
D	50 €
E	30 €
F	10 €

TAMAÑO	PRECIO
Entre 0 y 19 kg	10 €
Entre 20 y 49 kg	50 €
Entre 50 y 79 kg	80 €
Mayor que 80 kg	100 €

f) Crearemos una subclase llamada **Lavadora** con las siguientes características:

- Su atributo es carga, ademas de los atributos heredados.
- Por defecto, la carga es de 5 kg. Usa una constante para ello.
- Los constructores que deben implementarse son:
  - Un constructor por defecto.
  - Un constructor con el precio y peso. El resto por defecto.
  - Un constructor con la carga y el resto de atributos heredados. Recuerda que tienes que invocar al constructor de la clase padre.
- Los métodos que deben implementarse son:
  - Método get de carga.
  - precioFinal(): si tiene una carga mayor de 30 kg, aumenta el precio 50 €, si no es así, no se incrementa el precio. Invoca al método padre y añade el código necesario. Recuerda que las condiciones que hemos visto en la clase Electrodomestico también deben afectar al precio.

g) Crea una subclase llamada **Television** con las siguientes características:

- Sus atributos son resolución (en pulgadas) y smartTV (booleano), además de los atributos heredados.
- Por defecto, la resolución será de 20 pulgadas y smartTV será false.
- Los constructores que deben implementarse son:

- Un constructor por defecto.
- Un constructor con el precio y peso. El resto por defecto.
- Un constructor con la resolución, smartTV y el resto de atributos heredados. Recuerda que debes invocar al constructor de la clase pare.

- Los métodos que se tienen que implementar són:
  - Método get de resolución y smartTV.
  - precioFinal(): si tiene una resolución mayor de 40 pulgadas, se incrementa el precio un 30% y si és smartTV, se incrementa en 50 €. Recuerda que las condiciones de la clase Electrodomestico también tienen que afectar el precio.

h) Finalmente crea una clase que realice lo siguiente:

- Crea una lista/array de 10 Electrodomesticos.
- Asigna a cada posición un objeto de las clases anteriores con valores aleatorios.
- Recorre la lista/array y ejecuta el método precioFinal().
- Deberás mostrar el importe total 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 (puedes crear objetos Electrodomestico, pero recuerda que Television y Lavadora también son electrodomésticos). Recuerda el uso operador 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 Electrodomesticos, 200 para lavadora y 500 para televisión.

5. Escribe un programa que permita simular el funcionamiento del **inventario del juego Minecraft**. Si no has jugado nunca a Minecraft, puedes ver una explicación de su funcionamiento al siguiente enlace https://minecraft-es.gamepedia.com/inventario. La implementación que vamos a realizar debe cumplir con los siguientes requisitos:

a) Tendremos solo 7 casillas donde podremos guardar diferentes objetos del juego.

b) Algunos objetos se podrán apilar en una sola casilla en grupos de hasta 64, otros en grupos de 16 y otros no se podrán apilar.

c) Para simplificar tendremos los siguientes objetos disponibles:

- Pico, no apilable.
- Espada, no apilable.
- Piedra, apilable en grupos de 64.
- Madera, apilable en grupos de 64.

- Huevo, apilable en grupos de 16.
- Perla de Ender, apilable en grupos de 16.

d) Crea las clases que consideres oportunas para llevar a cabo la implementación de esta versión de inventario.

e) Una vez tengas hecha la implementación haz un menú como el siguiente:

1. Añadir al inventario
2. Eliminar del inventario
3. Mostrar inventario
0. Salir

f) Como viene siendo habitual, para facilitar las pruebas, añade al inventario de forma aleatoria algunos objetos al ejecutar la aplicación.

g) Ampliaciones: cualquier ampliación que quieras hacer será bienvenida. Ideas: añadir más tipos de objetos con nuevas formas de apilarse, uso de los objetos y su durabilidad, etc.

6. Escribe una clase llamada **Multimedia** para almacenar información de objetos de tipo multimedia (películas, videojuegos, ...).

a) Esta clase contiene el título, el autor, formato y año como características.

El formato puede ser uno de los siguientes: CD, DVD, BLU-RAY y ARCHIVO (tipo enum que puedes crear en un archivo público aparte dentro del paquete de la actividad). El valor de todos los atributos se pasarán como parámetros en el momento de crear el objeto. Esta clase, además, contiene un método para dar valor y para devolver cada uno de los atributos y un método toString() que devolverá la información del objeto. Por último, un método llamado equals que devuelva true en caso de que el título y el autor sean iguales a los del objeto que invoca al método y false en caso contrario.

b) Escribe una clase **Pelicula** que herede de la clase anterior. La clase Pelicula tendrá también una duración, un actor principal y una actriz principal. La clase deberá tener 3 métodos para obtener los 3 nuevos atributos y tendrá que sobrescribir el método toString() para que devuelva, además de la información heredada, la nueva información.

c) Escribe una clase **Videojuego** que también herede de la clase Multimedia. Además de los atributos de la clase padre, tendrá un atributo para almacenar la plataforma (Playstation 5, X Box, Nintendo Switch, etc..). Genera los métodos adecuados de esta clase sin olvidar toString().

d) Escribe una clase **Socio** con un NIF, nombre, fecha de nacimiento y población. Solo pueden ser socios los mayores de edad.

e) Los objetos multimedia se alquilan a los socios durante un periodo máximo de 3 días. El alquiler

tendrá un precio base de 4 €. El alquiler se ve rebajado 1 € si la pelicula es anterior al año 2012 o si el videojuego es anterior al año 2010.

f) Cuando el socio devuelve el objeto multimedia se debe comprobar que está dentro del plazo de alquiler de 3 días. Por cada día que pase del mencionado periodo, el socio deberá pagar un recargo de 2 €.

g) Un socio no puede alquilar mientras tenga recargos pendientes de pagar.

h) Crea un menú con las siguientes opciones:

- Altas ...
  - Alta de una nueva Película
  - Alta de un nuevo Videojuego
  - Alta de un nuevo socio.
- Alquilar multimedia a socio.
- Devolver multimedia
- Listados ...
  - Listado de todos los objetos multimedia
  - Listado de todas las películas ordenadas por título
  - Listado de todos los videojuegos ordenados por año
  - Listado del histórico de alquileres de un socio ordenados por fecha de alquiler
  - Listado de los alquileres actuales de un socio
  - Listado de los socios con recargos pendientes

i) Crea de forma aleatoria unos cuántos objetos de cada, utiliza la librería Java Faker si lo consideras oportuno.

7. Necesitamos una aplicación para gestionar la **venta de entradas** para un estadio de fútbol.

a) El estadio está dividido en zonas y cada una de estas zonas tiene un número de asientos numerados organizados por filas. Por simplicidad, supondremos que todas las zonas tienen la misma cantidad de asientos y de filas.

b) Hay zonas especiales llamadas zonas VIP.

c) El precio de las entradas es el mismo para todos los asientos de una determinada zona, pero puede variar según el tipo de partido, que podrá ser BAJA_AFLUENCIA, MEDIA_AFLUENCIA, ALTA_AFLUENCIA. El precio de las entradas se calcula de la siguiente forma:

- Cada zona tiene un precio base inicial y según el tipo de partido incrementará o reducirá su precio.

- Los partidos de BAJA_AFLUENCIA tendrán un precio final = precio base * 75%
- Los partidos de MEDIA_AFLUENCIA tendrán un precio final = precio base
- Los partidos d'ALTA_AFLUENCIA tendrán un precio final = precio base * 130%

d) De un partido además de su tipo, nos interesa, la fecha del partido y el nombre de los equipos local y visitante.

e) Hay dos tipos de entradas: entrada Normal y VIP. Ambos tipos tienen en común algunas propiedades: un número de entrada (autonumérico), el partido al que hacen referencia, la zona, la fila y el número de asiento.

f) Se decide crear tres clases: una clase padre **Entrada** y dos subclases **EntradaNormal** y **EntradaVip**.

- La entrada VIP, además de lo anterior, tiene un código alfanumérico (único y no deducible) especial que sirve como password para abrir la taquilla para dejar objetos personales y le permite al portador acceder a las zonas VIP.
- La entrada normal tiene además de lo definido en la clase padre Entrada, un número entre 1 y el número de asientos del estadio, que actúa como un número de lotería para un sorteo que se hace en el descanso de los partidos.

g) Cuando una entrada es vendida, el asiento correspondiente debe marcarse como ocupado.

h) El menú principal de la aplicación será similar al siguiente:

1. Nuevo partido
2. Gestión de entradas
0. Salir

Al seleccionar **Nuevo partido**, solicitará la fecha del partido, el tipo de partido y el nombre de los equipos local y visitante.

Al seleccionar la opción **Gestión de entradas**, mostrará el listado de partidos (solo aquellos que todavía no se han jugado) y permitirá seleccionar para qué partido del listado queremos gestionar entradas.

- Una vez seleccionado el partido, mostrará un menú similar al siguiente:

1. Venta de entradas
2. Devolver una entrada
3. Listado de localidades ocupadas
4. Listado de localidades libres
5. Mostrar recaudación del partido
0. Volver al menú principal

Al seleccionar la opción **Venta de entradas**, solicitará la zona, la fila y el número de asiento e imprimirá la entrada por pantalla indicando todos los datos y su precio. Si la zona seleccionada se trata de una zona VIP, se creará una entrada VIP, si no se creará una entrada normal.

Al seleccionar la opción **Devolver una entrada**, solicitará el número identificador de la entrada, marcará el asiento como libre y borrará la entrada.

Al seleccionar la opción **Listado de localidades ocupadas**, mostrará el listado de asientos ocupados.

Al seleccionar la opción **Listado de localidades libres**, mostrará el listado de asientos libres.

Al seleccionar la opción **Mostrar recaudación del partido**, mostrará el dinero acumulado por la venta de entradas del partido actual.