Ejercicios Java

Herencia y Polimorfismo

Instrucciones para resolver los ejercicios

- Resuelve los ejercicios en el proyecto creado en repositorio “Ejercicios 2ª Evaluación”.
- Las clases definidas para cada ejercicio se han de definir en el paquete `unidad7`.
- Cada vez que se resuelva un ejercicio se realizará un *Commit and Push* con el mensaje “Ejercicio *n* de la unidad 7 resuelto”, donde *n* será el número de ejercicio.
- No es obligatorio resolver y confirmar los ejercicios en el orden de numeración.

Ejercicio 1: Biblioteca

Una biblioteca pública necesita un programa para llevar a cabo la gestión de libros y revistas. Su desarrollo se llevará a cabo mediante programación orientada a objetos con Java, comenzando con la creación de una jerarquía de clases a partir de las especificaciones siguientes:

- Todas las publicaciones se caracterizan por un código, un título y un año de publicación.
- Por su parte, las revistas se caracterizan por un número, el mes y el día de su publicación, y los libros se por el nombre de sus autores.
- Por su parte, los libros completan su descripción con el nombre de sus autores.
- La biblioteca ofrece un servicio de préstamo que en principio estará disponible únicamente para los libros, pero no descarta en un futuro adquirir otras publicaciones, como por ejemplo publicaciones no impresas, que puedan formar parte del servicio de préstamos. Por tanto, este tipo de publicaciones han de incorporar a su estado información que determine si han sido prestadas.
- Se sabe que las revistas nunca formarán parte del servicio de préstamos.

Todas las clases tienen que cumplir con los requisitos siguientes:

- Definir uno o varios constructores con inicialización por defecto de los atributos que corresponda.
- Definir los getters y setters que corresponda.
- Definir un orden natural para sus instancias.
- Redefinir el método toString para que retorne una representación textual del estado de sus instancias con el formato siguiente:

  tipo_de_publicación [*valores de los atributos separados por comas*]

  Por ejemplo:

  "Libro [1, Java 9, 2018]"

- (**opcional**) Posibilitar el almacenamiento de sus instancias en cualquier colección.

Ejercicio 2: Prueba de la clase Biblioteca

Definir una clase que ponga a prueba en el método *main* las clases definidas en el ejercicio anterior. Se han de utilizar sentencias que pongan de manifiesto el tratamiento polimórfico de varias publicaciones de cualquier tipo almacenadas en un mismo *ArrayList*. Indicar con un comentario en el código fuente qué sentencia pone de manifiesto el tratamiento polimórfico y porqué.

Ejercicio 3: Electrodomésticos

Crear una jerarquía de clases Java para representar los electrodomésticos que se venden en una tienda Online especializada en lavadoras, frigoríficos y televisiones. Todos los electrodomésticos comparten las siguientes características:

- Precio base con un valor por defecto de 100 €.
- Color, representado por un tipo enumerado que defina los valores BLANCO, NEGRO, ROJO, AZUL y GRIS. El valor por defecto será BLANCO.
- Consumo energético representados por una letra comprendida entre la A y la F. Su valor por defecto será la letra F.
- Peso con un valor por defecto de 5 kg.
- Un precio final que se calcula aumentando el precio base en un porcentaje que dependerá del consumo y el peso según las tablas siguientes:

CONSUMO	PORCENTAJE
A	30%
B	25%
C	20%
D	15%
E	10%
F	5%

PESO	PORCENTAJE
Entre 0 y 19	5%
Entre 20 y 49 kg	10%
Entre 50 y 79 kg	15%
Más de 80 kg	20%

Todas las clases deben implementar o redefinir como mínimo los métodos siguientes:

- Constructores:
    - Constructor con parámetros para inicializar precio y peso.
    - Constructor con parámetros para inicializar todos los atributos.
- Getters y setters.
- Método toString.

Todos los electrodomésticos se podrán almacenar en cualquier colección.

Las lavadoras se caracterizan por soportar una carga según modelo que puede ser de 4, 5, 6, 7, 8, 10, 11 o 13 kg. El valor por defecto para la carga será de 5 kg. Su precio final aumentará en un 10% si la capacidad de carga es mayor de 8 kg.

Los frigoríficos pueden tener o no tener capacidad No Frost. Por defecto no la tendrán.

Las televisiones se caracterizan por su tamaño en pulgadas y por el tipo de sintonizador (DVB-T o DVB-T2). Los valores por defecto serán 20 pulgadas y DVB-T respectivamente.

Ejercicio 4: Empleados y Empresas

Una determinada empresa desea comenzar el desarrollo de una aplicación para gestionar el pago de las nóminas de sus empleados y de los servicios que contrata. Teniendo en cuenta que se utilizará el lenguaje de programación Java, se pide crear las jerarquías de clases e interfaces necesarias para representar los tipos de objetos que se describen en las especificaciones siguientes:

- Existen dos tipos de empleados: asalariados y contratistas. Todos ellos se caracterizan por su nombre, apellidos, fecha de contratación y número de cuenta bancaria. El valor de estos atributos se podrá consultar, pero no modificar, excepto el n.º de cuenta que se podrá consultar y modificar.
- Los contratistas se caracterizan además por poseer una o varias sociedades anónimas. Las sociedades anónimas son empresas especializadas en la realización de determinados trabajos para los que los empleados asalariados no están cualificados. Una sociedad anónima se caracteriza por su nombre (se puede consultar, pero no modificar) y por la lista de trabajos en los que está especializada que podrá variar a lo largo del tiempo.
- La empresa también contrata empresas de servicios (catering, telefonía, suministro eléctrico, limpieza, etc.) que se caracterizan por su nombre, el tipo de servicio prestado (ambos se pueden consultar, pero no modificar) y un número de cuenta bancaria (se puede modificar y consultar).
- Tanto los empleados como las empresas de servicios dispondrán de la misma funcionalidad para la realización del pago mediante ingreso en cuenta bancaria del importe de la nómina o del importe del servicio prestado. Simular el pago mediante mensajes por pantalla que indiquen que se realiza un ingreso en cuenta en concepto de lo que corresponda en cada caso e indicando el importe. El mensaje ha de ser específico para cada caso (asalariado, contratista o empresa de servicios).

Ejercicio 5: Prueba de Empleados y Empresas

Definir una clase que ponga a prueba en el método *main* las clases definidas en el ejercicio anterior. Se han de utilizar sentencias que pongan de manifiesto el tratamiento polimórfico de empleados y empresas de servicios, almacenándolos en un mismo *ArrayList*. La prueba ha de realizar, como mínimo, las acciones siguientes:

- Almacenar en el *ArrayList* al menos un empleado de cada tipo y al menos una empresa de servicios.
- Iterar sobre el ArrayList para realizar el pago del salario o de los servicios prestados, según corresponda.

El código ha de incluir un comentario que indique en qué sentencia pone de manifiesto el uso del polimorfismo y por qué.

(**Opcional)** En el mismo método *main*, se creará otro *ArrayList* y se almacenarán en él varios empleados de los dos tipos. A continuación, se ordenará mediante la clase *Collections* según el orden natural de los objetos almacenados, mostrando el resultado sin iterar sobre la colección. Ordenar de nuevo el *ArrayList* con un criterio de ordenación diferente.

Ejercicio 6: Blackjack

Desarrollar un programa para jugar a una versión simplificada del blackjack en la que no se permitirán apuestas y, por tanto, todas las reglas y opciones de juego afectadas por esta restricción no se aplicarán.

Se han de definir las clases siguientes:

- *Mano*: extenderá la clase Mazo para representar las manos de cada participante adaptadas al juego de blackjack.
- *Blackjack*: representará el juego de blackjack en el que la función del crupier siempre la desempeña el ordenador. La funcionalidad de esta clase ha de permitir la incorporación de jugadores nuevos en tiempo de ejecución (cuando una mano finaliza y antes del reparto de la siguiente).

En principio, la idea es desarrollar, utilizando las clases anteriores, una versión del juego para consola de un solo jugador. También se prevé el desarrollo en un futuro de una versión multijugador con interfaz gráfica de usuario. Esta segunda versión se ha de poder desarrollar utilizando las clases *Mano* y *Blackjack* sin necesidad de modificarlas lo más mínimo.

La versión para consola estará basada en la ejecución de comandos en una línea de comando que se mostrará como sigue:

```
Blackjack>
```

Los comandos que se podrán ejecutar en la línea de comando son los siguientes:

COMANDO	FUNCIÓN
`Blackjack> repartir`	Realiza el reparto inicial, mostrando la mano del crupier (una carta vista y una oculta) y la mano del jugador (las dos cartas vistas) con el valor de la misma. Muestra un error si la mano está en curso.
`Blackjack> pedir`	El crupier reparte una carta al jugador mostrando las cartas de nuevo y el valor de su mano. Muestra un error si hay una mano finalizada y no se ha vuelto a repartir.
`Blackjack> plantarse`	Muestra la mano del crupier y su valor, la mano del jugador y su valor y el resultado. Muestra un error si hay una mano finalizada y no se ha vuelto a repartir.
`Blackjack> fin`	Finaliza el programa pidiendo confirmación si hay una mano en curso.