Tarea presencial.
Unidad 5. Desarrollo de clases

Indicaciones de entrega

Una vez realizada la tarea, el envío se realizará a través de la plataforma Moodle, en la tarea "Tarea 4. Desarrollo de clases".

Debes comprimir la carpeta que contiene el proyecto en un archivo. Este archivo comprimido se nombrará de la siguiente forma:
Apellido1_Apellido2_Nombre_Tarea4.zip

Introducción

A lo largo de esta unidad has ido aprendiendo a crear tus propias clases, así como sus distintos miembros (atributos y métodos). Has experimentando con la encapsulación y accesibilidad (modificadores de acceso a miembros), has creado miembros estáticos (de clase) y de instancia (de objeto), has escrito constructores para tus clases, has sobrecargado métodos y los has utilizado en pequeñas aplicaciones. También has tenido tu primer encuentro el concepto de herencia, que ya desarrollarás en unidades más avanzadas junto con otros conceptos avanzados de la Programación Orientada a Objetos.
Una vez finalizada la unidad se puede decir que tienes un dominio adecuado del lenguaje Java como para desarrollar tus propias clases y utilizarlas en una aplicación final que sea capaz de manipular un conjunto de datos simple. Dada esa premisa, esta tarea tendrá como objetivo escribir una pequeña aplicación en Java empleando algunos de los elementos que has aprendido a utilizar.

Enunciado

Se trata de desarrollar una aplicación Java en consola que permita gestionar un artículo concreto de una **tienda de deportes**.
Además, se elaborará un **documento PDF** en el que habrá pantallazos suficientes que demuestren que todo se puede ejecutar y funciona. En ese documento se pueden realizar las explicaciones necesarias para facilitar la comprensión del programa.

Conviene leer todo por completo, antes de implementar nada, ya que en el enunciado se explican ciertos elementos que no van necesariamente en el orden en el que se presentan.

Para que tengas una idea general del programa antes de explicar el detalle, realizará lo siguiente:
1. Al empezar dará un mensaje de Bienvenida de la Tienda (inventa un nombre para la misma)
2. Pedirá un código de artículo completo.
3. Pedirá una Descripción de ese artículo.
4. Si todo es correcto, se mostrará el menú que permitirá realizar las operaciones que se explican más abajo.
5. Al salir se mostrará un mensaje de despedida de la tienda.

Clase LanzadorTienda.java

Es la clase donde irá el main, y que lanzará el menú y se llamará LanzadorTienda.java
En la medida que estimes oportuno hay que realizar las diferentes funcionalidades en métodos independientes (uno para mostrar el menú, otro para leer del teclado, otro para leer la opción, etc). En el enunciado se da una idea de nomenclaturas y estructura para dividir en métodos, pero se admite otra estructura siempre que esté justificado y cumpla las funcionalidades.
El programa mostrará un menú se podrán realizar determinadas opciones:
1. Ver el código completo del artículo: consta de un código de 14 dígitos numéricos. Los 2 primeros dígitos hacen referencia a la ciudad, los 2 siguientes la tienda, los 2 siguientes al tipo de artículo, los 6 siguientes es el código de artículo en sí mismo, y los 2 últimos es un número de control.
El número de control se calcula sumando el código de ciudad, más el código de tienda, más el código del artículo y dividiendo entre 99. El resto de dicha división entera, es el código de control.
Ejemplo: 130100009914 sería la ciudad 13, tienda 01, tipo 03, artículo 000099, y dígito de control 14.
13+01+000099= 113
113/99= Cociente 1, Resto 14 (por eso el código completo termina en 14)

2. Ver la descripción del artículo: El artículo es una cadena de texto de máximo 40 caracteres.
Ejemplo: “Bicicleta MTB Trek X-Caliber 9”

3. Ver el código de la ciudad. (obtenido del código completo del artículo). Ejemplo: 13

4. Ver el código de la tienda. (obtenido del código completo del artículo). Ejemplo: 01

5. Ver el código del artículo. (obtenido del código completo del artículo, se refiere solamente al número del artículo, sin ciudad, tienda ni dígito de control).
Ejemplo: 000009

6. Ver el tipo de artículo. Siendo 01 para textil, 02 para zapatería y 03 para complementos. Si es un complemento, además deberá indicar el tipo. (obtenido del código completo del artículo). Ejemplo: Complementos. Bicicleta.

7. Crear artículo. Habrá que solicitar por teclado la información del artículo, que se añadirá a la lista de artículos. Si existe previamente se incrementará en dicha cantidad. A su vez reutilizará el leerTeclado()). Con esos datos y una vez cumpla las validaciones que se explican después, creará un objeto de tipo ArticuloDeportivoXxxxx (clase explicada más adelante), que servirá para realizar el resto de operativas.

Datos a leer:

- Descripción del artículo (con un máximo de 40 caracteres).

- Código del artículo completo: Se tiene que leer completo, no vale ir pidiéndolo

troceado.

En los métodos validarDescripcion y validarCodigoArticulo (ambos estáticos y privados), deberá asegurarse que la descripción no supera los 40 caracteres, y que el código de artículo completo es válido mediante la comprobación de que:

- Son 14 caracteres numéricos. (quizás para el tratamiento posterior te conviene que sea una cadena, pero comprobar toda la cadena son dígitos numéricos)

- El dígito de control es válido (según la fórmula que se vio anteriormente)

Si alguno de esos datos es incorrecto el programa mostrartá un error y no añadirá el artículo, indicando en un mensaje que no se pudo añadir.

Además, deberá a partir del código completo del artículo, rellenar los atributos del objeto (ciudad, tienda, tipo, código, dígito de control). Es decir, se lee el código completo, pero hay que trocearlo, para poder luego crear bien el objeto **ArticuloDeportivoXxxx** con los atributos que más adelante se explican.

NOTA: Como mejora voluntaria, podrías no finalizar la ejecución del programa, y volver a pedir los datos que hayan sido incorrectos hasta que sea correcto, o bien hasta que el usuario introduzca una S, que indica que quiere salir.

8. Decrementar unidades. Habrá que solicitar por teclado la cantidad que se desea decrementar y el total se decrementará en dicha cantidad (controlando que no queda un balance negativo si es así el total se deja en 0, y se saca un mensaje por pantalla indicando el hecho).

9. Consultar Unidades del artículo que hay en ese momento. (con las anteriores opciones se incrementa o decrementa, y esta se pueden ver, por ejemplo, cuántas bicicletas de ese modelo quedan en la tienda)

10. Salir de la aplicación. (al salir se mostrará toda la información del artículo, usando el toString() que se explicará más adelante y se mostrará un mensaje de despedida con el nombre de la tienda)

Consideraciones sobre esta clase:

La visualización del Menu estará en un método llamado, por ejemplo: mostrarMenu (privado y estático)

Además, dentro de la clase crearemos los siguientes métodos privados y estáticos:

• String leerTeclado(): leerá desde el teclado cuando se necesite leer y nos devuelve un String con lo tecleado.

Pondremos un try-catch y propagaremos la excepción, si salta alguna.

• int leerOpcion (): método que utilizará el anterior y nos devolverá una opción válida (entre 0 y 9).

Pondremos un try-catch y propagaremos la excepción, si salta NumberFormatException, IOException u otra.

• Antes de que aparezca este menú, el programa podrá crear tantos artículos como se crea oportuno mediante un método llamado inicializarDatosArticulos.

Clase ArticuloDeportivo.java

Otra clase, será **ArticuloDeportivo.java** que proporcionará todas las herramientas necesarias para trabajar con este tipo de información y que debe de contener además de lo que veas oportuno, lo siguiente. Será una clase abstracta de la que heredarán los 3 tipos de artículos descritos más adelante.

- Tres constructores:
  - Uno con valores por defecto.
  - Otro recibiendo todo lo necesario para dar identidad al artículo.
  - Constructor copia a partir de otro objeto artículo. (clone)
- toString(): Devuelve una cadena con toda la información del artículo.
- Atributos, necesarios para almacenar los datos del articulo explicados anteriormente, con los tipos y visibilidades que estimes oportuno, justificando el porqué.
- Métodos asociados a los atributos que permitan modificar y obtener el valor, con la visibilidad que determines correcta (getters y setters).

  Ojo, el atributo **dígito de control es calculado**, por lo tanto, no puede modificarse desde fuera. Eso sí, siempre que se modifique el código de ciudad, de tienda o de artículo de ese artículo, se tiene que recalcular mediante un método

privado (no se permite acceder como interfaz desde fuera) y no estático (ya que es del objeto concreto, no de la clase). El método se llamará calcularDigitoControl()

- Unidades (atributo que indica cuántas unidades de ese artículo hay, empezará siendo cero, salvo que se construya con un número concreto).
- Incremento y decremento de las unidades (métodos con visibilidad que estimes correcta que incrementan o decrementan las unidades como se explicaba anteriormente).
- Consulta de las unidades (método con visibilidad que estimes correcta, que muestra las unidades).
- Otras: Aquellas herramientas auxiliares necesarias para poder trabajar cómodamente con el objeto. Tipos de datos, visibilidades, algunas podrán ser específicas de clase y otras podrán ser de objeto (métodos de objeto privados, métodos estáticos públicos, etc.) que estimes oportuno y justifiques.

Clase ArticuloDeportivoTextil.java

Heredará de la clase ArticuloDeportivo y tendrá el atributo talla, que solo podrá ser S, M, L y XL

Clase ArticuloDeportivoZapateria.java

Heredará de la clase ArticuloDeportivo y tendrá los atributos talla, que solo podrá ser un número entero entre el 27 y el 46; y horma que podrá ser ancha, regular o estrecha.

Clase ArticuloDeportivoComplemento.java

Heredará de la clase ArticuloDeportivo y tendrá el atributo tipo que será una cadena de texto de una sola palabra (“bicicleta”, “raqueta”, “balón”, etc.).

Clase ExcepcionTiendaDeportes.java

En general, deberías incluir excepciones para controlar aquellos casos en los que el uso de un método no sea posible (intentar introducir una descripción con más caracteres de los permitidos, intentar ingresar o retirar una cantidad negativa, introducir un código de ciudad, tienda o artículo no permitido etc.).

Cuando sea una excepción controlada y generada por ti, deberás utilizar una clase que tú crees y que llamarás ExcepcionTiendaDeportes y será ese el tipo de excepción que lances y controlarás, con try catch donde creas oportuno.

Si es de otro tipo de excepción, que genera el sistema, también la capturarás o lanzarás como nuestro propio tipo, traspasando la información que sea necesaria de la excepción original.

El código fuente debería incluir comentarios en cada atributo (o en cada conjunto de atributos) y método (o en cada conjunto de métodos del mismo tipo) indicando su utilidad y las explicaciones necesarias.