TAREA PARA PROG05

Detalles de la tarea de esta unidad.

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**. (de ese artículo se pueden tener muchas unidades como verás más adelante). Además, se elaborará un **documento PDF** en el que habrá pantallazos suficientes que demuestren que todo se puede **ejecutar** y **funciona**, y **acreditando identidad** mostrando el aula virtual para ello. 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. (Programa principal que utiliza Articulo, Articulo que utiliza ExcepcionTiendaDeportes etc.)

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 se justifique, cumpla las funcionalidades y sea correcta.** (y todo esté comentado poniendo tu nombre y apellidos además de las explicaciones que veas oportuno)

El programa realizará lo siguiente:

Mostrará un menú se podrán realizar determinadas opciones:

1. **Ver el código completo del artículo**: consta de un código de 12 dígitos numéricos. Los 2 primeros dígitos hacen referencia a la ciudad, los 2 siguientes la tienda, 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, 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 **dígito de control**. (obtenido del código completo del artículo). **Ejemplo**: 14
7. **Aumentar unidades** de ese artículo. Habrá que solicitar por teclado la cantidad que se desea aumentar y el total se incrementará en dicha cantidad.
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** lo introduciremos 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 mediante un método llamado **obtenerDatosArticulo**, tendrá que solicitar al usuario los siguientes datos (a su vez reutilizará el **leerTeclado**()). Con esos datos y una vez cumpla las validaciones que se

explican después, creará un objeto de tipo ArticuloDeportivo (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.

El programa, 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:

- Son 12 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 finalizará, indicando en un mensaje el motivo de la finalización. Todo esto hazlo en un método aparte.

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

(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)

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.
  - 3 **constructores**:
    - Uno con valores por defecto.
    - Otro recibiendo todo lo necesario para dar identidad al artículo.
    - Constructor copia a partir de otro objeto artículo. (como se ve en los apuntes)
  - **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.
    Ojo, el **atributo dígito de control** es calculado, por lo tanto, no puede modificarse desde fuera, no necesita un método modificar. Eso sí, siempre que se modifique el código de ciudad, de tienda o de artículo de ese artículo, se tiene que recalcular el dígito de control, 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 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** (como hiciste en la unidad anterior) y será ese el tipo de excepción que lances y controlarás, con **try catch** donde creas oportuno (siempre que lo justifiques).

Si es de otro tipo de excepción, que genera el sistema, también la capturarás o lanzarás, como mínimo de manera genérica (**Exception**) o se algún subtipo que creas conveniente, como **NumberFormatException**, **IOException** o cualquier otra.

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.

Como hemos visto, en total, como mínimo habrá **3 clases salvo que tú justifiques la creación y uso de alguna más.**

Como ya se dijo al principio, además del programa, deberás escribir también un documento, con pantallazos y las explicaciones que estimes oportunas, en **formato PDF** (no se admite otro formato).

Hay cierta flexibilidad en realizar el programa mientras cumpla los requisitos y **una buena explicación** de cómo lo has interpretado y realizado, ayudará positivamente.

El proyecto deberá contener al menos tres archivos fuente Java:

- **Programa principal** (clase con método main: **LanzadorTienda**.java).
- La clase **ArticuloDeportivo.java**.
- La clase **ExcepcionTiendaDeportes.java.**

Criterios de puntuación. Total 10 puntos. (+0,5 extra)

Para poder empezar a aplicar estos criterios es necesario que la aplicación compile y se ejecute correctamente. En caso contrario la puntuación será directamente mínima, dependiendo del informe entregado.

- **Clase LanzadorTienda**: 4,5 puntos
- **Clase: ArticuloDeportivo:** 4,5 puntos
- **Clase: ExcepcionTiendaDeportes:** 0,5 puntos
- **Aspectos generales**: 0,5 puntos: Buena presentación del documento explicativo y que acredita identidad, buen diseño proyecto, bien estructurado, etc.

En la puntuación total se podrá descontar por lo siguiente:

- **-0,5 puntos**: No se han incluido comentarios describiendo el funcionamiento de todos los métodos y atributos en ambas clases, como se ha pedido en el enunciado.
- **-1 punto**: No se ha entregado el informe explicativo o se trata de un informe explicativo insuficiente
- **-1 punto**: Alguna de las opciones de pedidas en el enunciado (menú del programa principal) no funciona correctamente: por cada opción que no funcione.
- **-0.5 puntos**: Faltan elementos que se piden en las especificaciones.

Dado que algunos criterios de puntuación son negativos, podría suceder que el balance final fuera negativo. En tal caso la puntuación final será simplemente la mínima.

Si se realiza el menú con la parte voluntaria, se incrementará **0,5 puntos**, no pudiendo obtener una puntuación de 10,5. Si se diese el caso, se dará una felicitación, pero la nota máxima que se puede registrar es de **10**. Ese medio punto sí ayudará a aumentar la puntuación si hay algún otro error, por ejemplo, puede hacer que una práctica de 4.5, se convierta en un 5, o de 9.5 en un 10.

Recursos necesarios para realizar la Tarea.

- Ordenador personal.
- Sistema operativo Windows o Linux.
- JDK y JRE de Java (preferiblemente últimas versiones)
- NetBeans IDE 6.9.1 o superior. (preferiblemente Apache NetBeans 12.5)

Consejos y recomendaciones.

Dentro de la carpeta de NetBeans, se habrá creado automáticamente la carpeta del proyecto. Comprueba que efectivamente está la carpeta y dentro todo el contenido que has creado.

Para entregar, **comprime en un único** archivo la **carpeta del proyecto** con todo su contenido, junto con el **documento PDF** y **pantallazos que demuestran la identidad**.

Indicaciones de entrega.

Se enviará un **único archivo comprimido que contendrá el proyecto, y un documento PDF**, con pantallazos que muestren identidad y explicaciones necesarias sobre el programa. Además, se tiene que ver los ejercicios realizados, completos y **ejecutados sin errores**, junto con vuestra identidad, poniendo de fondo la pantalla del aula virtual.

El envío se realizará a través de la plataforma y el archivo se nombrará siguiendo las siguientes pautas:

apellido1_apellido2_nombre_SIGxx_Tarea

Asegúrate que el nombre no contenga la letra ñ, tildes ni caracteres especiales extraños. Así por ejemplo la alumna Begoña Sánchez Mañas para la quinta unidad del MP de PROG, debería nombrar esta tarea como:

sanchez_manas_begona_PROG05_Tarea.