GLOBAL CITIZEN
Academy

Examen de ingreso

Curso de Ingeniería de Software orientado a Videojuegos.

Antes de comenzar:

La siguiente ejercitación tiene como objetivo poner a prueba tus conocimientos sobre Programación Orientada a Objetos. Está dividida en módulos (parte A, parte B, parte C, etc) que se encuentran ordenados para ser resueltos de manera incremental, por lo que cada uno de ellos es dependiente de los anteriores. Si quieres tener éxito, te recomendamos que los resuelvas en orden.

Podrás utilizar C# para implementar el código y para el modelado UML te recomendamos usar draw.io (o alguna herramienta que domines con facilidad).

Con respecto al IDE de desarrollo, sugerimos que utilices Visual Studio versión 2015 en adelante. El proyecto debe desarrollarse sobre el *.NET Framework*. Puedes elegir entre un proyecto de consola para windows, o un proyecto de Windows Forms, **¡como más te guste!**

Importante: si estás utilizando MacOS o Linux podés desarrollar utilizando Mono. Otra opción es desarrollar con el ***.Net Core*** utilizando Visual Studio (sólo para usuarios de MacOS). Y como alternativa te damos la opción de que desarrolles el programa utilizando el lenguaje JAVA.

El proyecto deberá ser entregado en un repositorio público de github.

Sugerimos que no agregues el archivo .gitignore y que subas el proyecto completo (todos los archivos, inclusive el .exe generado en la última compilación).

En el repositorio de github, no te olvides incluir la imagen de los diagramas modelados y en el readme.md, puedes agregar las respuestas a las preguntas teóricas.

Ahora sí, comencemos!

A continuación encontrarás los módulos de la ejercitación, los cuales te brindarán la información necesaria para que puedas desarrollar la aplicación. El programa será utilizado por un vendedor de una tienda de ropa mayorista para realizar cotizaciones de sus productos.

Parte A

1. Realizar un diagrama de clases que modele la entidad **Vendedor**. El usuario del programa será el vendedor de una tienda de ropa mayorista.

En principio, un vendedor posee **Nombre, Apellido y Código de Vendedor.**

2. El vendedor, podrá realizar distintas **Cotizaciones** antes de llevar a cabo una Venta. Las cotizaciones se deberán almacenar en un historial del vendedor.

Una Cotización se compone de los siguientes atributos:

- Número de identificación
- Fecha y Hora de la cotización
- Código del Vendedor que realiza la cotización
- Prenda cotizada
- Cantidad de unidades cotizadas
- Resultado del cálculo de la cotización.

3. Implementar las clases creando los miembros necesarios.

Parte B

1. Modelar las entidades especificadas a continuación y relacionarlas según corresponda.

El Vendedor trabaja en una Tienda de ropa. Esta última posee un Nombre, Dirección y un listado de distintas Prendas para vender.

Una prenda tiene algunas propiedades como: calidad de prenda (standard o Premium), precio de la prenda y cantidad de unidades en stock.

2. Implementar las clases creando los atributos necesarios.

Parte C

1. Además, existen dos tipos de prendas: Pantalones y Camisas.
- Las camisas pueden ser de manga corta o larga.
- Además, las camisas pueden tener cuello mao.
- Los pantalones pueden ser comunes o chupines.

2. Completar el modelo de clases.

3. Es necesario que el programa permita marcar si el tipo de camisa es de manga corta o con cuello mao, ya que las mismas tienen un precio diferente a las camisas convencionales.
Lo mismo para los chupines (que tienen un precio diferente a los pantalones normales).

4. El precio final que calcula el programa (cotización) responderá a las siguientes reglas de negocio:
RN 1- Si la camisa es de tipo manga corta, el precio se rebaja en un 10%.
RN 2- Si la camisa tiene cuello mao, el precio aumenta en un 3%.
RN 3- Si la camisa es de manga corta y cuello mao, deben aplicarse las dos reglas anteriores (en el orden establecido).
RN 4- Si el pantalón es chupín, el precio se rebaja en un 12%.
RN 5- Si la calidad de la prenda es Standard: el precio no se modifica.
RN 6- Si la calidad de la prenda es Premium: el precio aumenta en un 30%.

5. Modificar la implementación contemplando los nuevos cambios.

Parte D

1. Hasta aquí, el frontend de la aplicación podría verse así:

Nota: como la aplicación puede ser de consola o de windows forms, se puede proponer alguna otra disposición de controles visuales, lo importante es que el vendedor pueda generar cotizaciones en base a la prenda seleccionada.

Observación: en la GUI hay un botón para poder visualizar el historial de cotizaciones que ha realizado el vendedor hasta el momento. Dicha lista podría ser mostrada en otra ventana.

Parte E

1. Al iniciar el programa, deberá crearse el objeto Tienda con datos ficticios, al igual que el objeto Vendedor.

Además, también habrá un número fijo de prendas en stock, siendo:

- 1000 camisas en total, de las cuales:
  - 500 de manga corta
    - 200 con cuello mao
    - 300 con cuello normal
  - 500 de manga larga
    - 150 con cuello mao
    - 350 con cuello normal

- 2000 pantalones en total, de los cuales:
  - 1500 son chupines
  - 500 son normales

Restricción: si el usuario intenta Cotizar sobre una cantidad de stock no disponible, el sistema emitirá un mensaje de error indicando que no se puede realizar una cotización sobre una cantidad de stock no disponible.

2. Se deberá poder consultar el Historial de Cotizaciones en cualquier momento de la ejecución del programa.

El historial será una lista sencilla que muestre todos los datos de las cotizaciones realizadas anteriormente por el Vendedor.

Parte F

1. Finalmente, se desea poder imprimir tanto la cotización actual como el historial completo de cotizaciones del vendedor. ¿Cómo se podría establecer a estos objetos como imprimibles?.

2. Modificar el diagrama de clases correspondiente.

Aclaración: habiendo planteado la solución en el diagrama de clases, no es necesario implementar la funcionalidad de impresión en el programa.

Parte G

A este punto ya deberías tener todo un esquema de clases implementado en código C# con el diagrama de clases correspondiente. Te sugerimos que revises el diagrama para chequear que no te falta modelar ninguna relación, atributo o método. Además, si desarrollaste una aplicación de consola, será necesario que implementes un menú de opciones que le permita al usuario utilizar la aplicación. Si trabajaste con Windows Forms, entonces las interfaces de usuario deberán responder correctamente a acciones del usuario.

Parte H

Responder las siguientes preguntas:

1. Por favor, especifica tu nombre completo: _____
2. ¿C# permite herencia múltiple?
3. ¿Cuándo utilizaría una Clase Abstracta en lugar de una Interfaz? Ejemplifique.
4. ¿Qué implica una relación de Generalización entre dos clases?
5. ¿Qué implica una relación de Implementación entre una clase y una interfaz?
6. ¿Qué diferencia hay entre la relación de Composición y la Agregación?
7. Indique V o F según corresponda. Diferencia entre Asociación y Agregación:
    a. Una diferencia es que la segunda indica la relación entre un “todo” y sus “partes”, mientras que en la primera los objetos están al mismo nivel contextual.
    b. Una diferencia es que la Agregación es de cardinalidad 1 a muchos mientras que la Asociación es de 1 a 1.
    c. Una diferencia es que, en la Agregación, la vida o existencia de los objetos relacionados está fuertemente ligada, es decir que si “muere” el objeto contenedor también morirán las “partes”, en cambio en la Asociación los objetos viven y existen independientemente de la relación.

Importante: la respuesta a las preguntas anteriores puedes colocarlas en el archivo readme.md de tu repositorio gitHub.