QUARK ACADEMY

# Examen de ingreso

## Programa de Ingeniería de Software en 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 querés tener éxito, es importante que los resuelvas en orden.

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

Con respecto al IDE de desarrollo, sugerimos que utilices Visual Studio community versión 2019. El proyecto debe desarrollarse sobre el *.NET Framework (no usar .NET Core)*. Puedes elegir entre un proyecto de consola para windows, o un proyecto de Windows Forms, **¡como más te guste!**

**Importante:** 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.

**Importante**: recordá agregar en el readme las instrucciones para la ejecución del programa.

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 el código completo de la aplicación (incluyendo el .exe generado en la última compilación).

**Importante**: para simplificar la corrección del examen, te pedimos que no lo hagas con base de datos. Los mismos podrán estar almacenados en archivos JSON o persistir temporalmente en memoria RAM (incrustados en el código fuente del programa).

## Ahora sí, comencemos!

A continuación encontrarás los módulos de la ejercitación, los cuales te brindarán la información necesaria para que puedas desarrollar una aplicación para la administración de una “Biblioteca”.

# Parte A

1. Realizar un diagrama de clases que modele la entidad **Libro**.

En principio, un libro posee un nombre, un código ISBN y un autor.

2. Implementar la clase creando los atributos necesarios.

# Parte B

1. Agregar al diagrama de clases, a la entidad **Socio** de la biblioteca.

En principio, un Socio tiene un nombre, un apellido y un número de identificación.

2. Implementar la clase creando los atributos necesarios.

# Parte C

Se requiere agregar al modelo anterior una nueva categoría de socios, los socios **VIP**. Dichos socios además del nombre, apellido y número de identificación, tienen un valor de **cuota mensual**.

1. Modificar el diagrama para que contemple la nueva categoría de socios.

2. Modificar la implementación contemplando los nuevos cambios. Crear las clases y atributos que sean necesarios.

# Parte D

1. Agregar al diagrama de clases la entidad **Ejemplar** de la biblioteca.

En principio, un ejemplar tiene un Libro, un número edición y una ubicación dentro de la biblioteca.

2. Implementar la clase creando los atributos que sean necesarios.

# Parte E

Un Libro de una biblioteca además de tener nombre, código ISBN y autor, posee una **lista de ejemplares** disponibles para ser prestados.

1. Modificar en el diagrama la entidad Libro que modeló anteriormente.

2. Modificar la implementación contemplando los nuevos cambios.

# Parte F

Un socio de una biblioteca además de tener nombre, apellido, número de identificación, posee una lista de **ejemplares retirados** y una **cantidad máxima** de libros que puede retirar. Si es un socio clásico, puede llevarse hasta 1 libro, pero si es un socio VIP puede llevarse hasta 3 libros.

1. Modificar el diagrama para que las entidades Socio y SocioVIP contemplen los nuevos cambios.

2. Modificar la implementación contemplando los nuevos cambios.

# Parte G

1. La clase Libro debe implementar el siguiente comportamiento:
   a. Agregar un nuevo ejemplar a la lista de ejemplares.
   b. Consultar si el libro tiene ejemplares disponibles. Este método devuelve true si tiene disponible ejemplares o false en caso contrario.
   c. Prestar un ejemplar del libro. El método tiene que eliminar de la lista de ejemplares el primer ejemplar y retornar dicho ejemplar.
   d. Registrar el reingreso de un ejemplar que fue prestado. Este método debe agregar a la lista de ejemplares, el ejemplar que recibe por parámetro.

# Parte H

1. La clase Socio en cambio, debe implementar el siguiente comportamiento:
    a. Consultar si un socio tiene cupo disponible para llevarse un libro. Este método devuelve true si tiene cupo o false si no tiene cupo.

       **Aclaración**: recordar que a un socio clásico sólo se le presta 1 ejemplar mientras que a un socio VIP se le prestan hasta 3 libros.

    b. Pedir prestado un ejemplar. Es decir, el método deberá agregar un ejemplar a la lista de ejemplares prestados del socio.

    c. Devolver un ejemplar. Es decir, el método deberá eliminar de la lista de ejemplares prestados al socio, el ejemplar prestado, ya que el socio hizo la devolución.

# Parte I

Se requiere modelar la representación de un objeto **Préstamo**. Esta entidad representa el préstamo de un ejemplar a un socio. En principio, posee un Ejemplar, un Socio y una Fecha de préstamo. Los préstamos siempre vencen a los 5 días (por lo que no es necesario registrar la fecha de finalización del préstamo).

1. Agregar la clase Préstamo en el diagrama de clases.

2. Implementar la clase creando los atributos necesarios.

3. Crear un constructor que tome como parámetro al socio y al ejemplar. El método constructor debe generar un préstamo con la fecha del día actual.

# Parte J

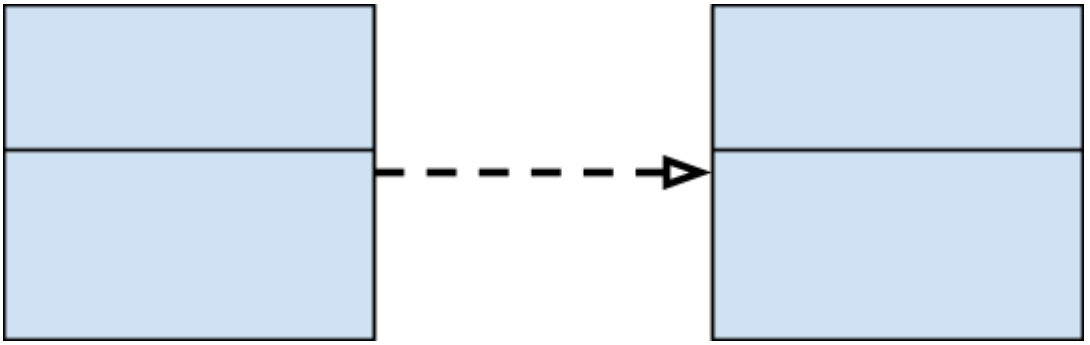
A este punto ya deberías tener un **diagrama de clases del dominio** y su implementación en **en código C#**. Te sugerimos que revises el diagrama para chequear que no te falta modelar ninguna relación, atributo o método. Además, si desarrollaste una aplicación de consola, será necesario que implementes 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 K

## Responder las siguientes preguntas:

Por favor, especifica tu nombre completo: _____

1. En C#, ¿para qué sirve una propiedad?.
2. ¿Cuándo utilizaría acceso protegido en los miembros de una clase?.
3. En UML, ¿qué tipo de relación es la siguiente?



Explique de qué trata dicha relación.

4. Explique con sus palabras qué implica una relación de **Dependencia** entre dos clases.
5. Indique **V** o **F** según corresponda. Si es **F**, fundamente su respuesta:
    a. Un constructor es un método que se invoca de forma automática cuando se instancia el objeto de la clase.
    b. Un constructor debe tener siempre el mismo nombre de la clase.
    c. Un constructor puede retornar un valor.
    d. Un constructor puede ser privado.
    e. Una clase sólo puede tener declarado un único constructor.

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