

**Universidad Abierta y a Distancia**

**de México**

**División de Ciencias Exactas, Ingeniería y Tecnología**

**Desarrollo de software**

**Semestre:** Sexto Semestre

**Asignatura:** Programación net II

**Unidad de aprendizaje: Desarrollo de objetos en la plataforma .NET II**

**Actividad: Encapsulación de datos mediante C#**

**Nombre del estudiante:** Francisco González Antonio

**Matrícula:** ES1821009465

**Grupo: DS-DPRN2-2202-B2-003**

**Docente: José Francisco Rico Gallegos**

**Fecha de entrega:**

**Ciudad de México, 12 de Octubre de 2022**

Diseño: DL-CPL

**ÍNDICE**

[**INTRODUCCIÓN** 5](#_Toc80333573)

[**DESARROLLO DE LA ACTIVIDAD** 6](#_Toc80333574)

[**CONCLUSIONES** 7](#_Toc80333575)

[**REFERENCIAS BIBLIOGRÁFICAS** 8](#_Toc80333576)

# **INTRODUCCIÓN**

En esta actividad se realizará la abstracción de los personajes proporcionados. Los datos a abstraer son el nombre real, el nombre del personaje, la primera aparición en New 52, los poderes que son un arreglo de cadenas de textos, el creador del personaje y el artista que lo diseñó. Adicional a ello se muestran la información que se requiere del ejercicio para hacer que el personaje simula una pelea.

Cada unos de los datos abstraídos tienen una especificación de acceso privada por lo cual es necesario implementar su respectivo set y get para el caso en el que se consuma la información.

Se agregan ciertos métodos para mostrar la información de la ficha técnica y del estado actual de la pelea del personaje. También se agregan los métodos solicitados para interactuar con la información de combate y peleas del personaje. Dentro de la función principal se simula la creación del personaje de Wonder Woman mostrando todos sus métodos e imprimiendo los valores después de simular un combate y cómo los atributos físicos de dicho personaje se ven disminuidos según los combates que vaya teniendo.

# **DESARROLLO DE LA ACTIVIDAD**

Como parte de la visión del problema esta actividad nos puede introducir a una acción que realizaremos en la vida real y es manejar la programación orientada a Objetos, en este caso los Personajes son vistos como un objeto y haciendo uso de la abstracción (que es un pilar de la programación orientada a objetos) es posible crear y diseñar una clase Personaje que cuenta con los atributos más generales y propios del personaje. Más aún, al tratarse de C# el problema de abstraer personajes puede ser consistente con el desarrollo de un videojuego, ya que Unity utiliza C# para la codificación. Las herramientas a utilizar son .NET Framework y una herramienta de diagramado para documentar y permitir visualizar el cómo se va a implementar la abstracción de la clase Personaje.

En las instrucciones de esta actividad se ha solicitado que se abstraigan ciertos personajes de cómics, para ello es importante considerar los atributos que se encuentran en común en todos los personajes y más aún, aquellos atributos que son inherentes a cualquier personaje como por ejemplo el nombre del personaje, el nombre real o si es o no humano. En este sentido es posible considerar las instrucciones como requerimientos funcionales para la clase personaje.

A continuación como primer acercamiento a la solución, el planteamiento del problema y el desarrollo del código he colocado información sobre los atributos que todos los personajes tienen en común y que se pueden colocar en una clase Personaje para hacer uso de la herencia. La siguiente tabla muestra el punto de partida de los requerimientos funcionales del diseño pues se muestran los datos y cuáles se pueden abstraer porque todos los personajes cuentan con ellos.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Personaje** | | | |
| **Atributo** | **Wonder Woman** | **Lobo** | **Reverse Flash** | **Catwoman** |
| **Nombre real** | x | x | x | x |
| **Primera aparición histórica** | x | x |  | x |
| **Primera aparición New S2** | x | x | x | x |
| **Poderes** | x | x | x | x |
| **Creado por** | x | x | x | x |
| **Ficha técnica** | x | x | x | x |
| **Información adicional** |  |  |  |  |

Aunque la “información adicional” se encuentra disponible para todos los personajes considero que es demasiada como para colocarla dentro de los atributos esenciales de un personaje y más aún como parte de una abstracción. A continuación se muestra un diagrama de clase para el diseño de Personaje.

Table

Description automatically generated with low confidence

**Seis propiedades que permiten modificar los atributos abstraídos**

En la actividad se solicitan seis atributos y además algunos métodos para estos atributos. El total de atributos abstraídos por mí con base en la información son siete, los cuales uno está declarado en el constructor y corresponde al nombre del Personaje. Adicional hay cinco atributos solicitados como parte de la actividad. A continuación muestro los atributos segmentados en los que abstraje y en los que son parte de la actividad y adicional se muestra el constructor con un único atributo que es el nombre del personaje.

Text

Description automatically generated

Los atributos son privados ya que cada uno de los que se necesiten mostrar tienen su método get y set respectivamente.

**A screenshot of a computer screen

Description automatically generated with medium confidence**

Ahora bien he creado dos métodos que muestran la información del personaje, el primero corresponde a la información que es posible obtener en la ficha técnica de la actividad y el otro que permite verificar el estado actual del personaje con base en la información indicada en la actividad:

Text

Description automatically generated

Text

Description automatically generated

**EsHumano y sus métodos**

****

**Graphical user interface, text, application

Description automatically generated**

**Puntos y sus métodos**

****

****

El método que modifica los puntos de experiencia y el cansancio está englobado dentro del método pelea pues estos valores cambian si y sólo si hay una pelea:

Text

Description automatically generated

**NivelFisico y sus métodos**

****

También nivel físico cambia cuando hay una pelea, por lo cual dentro del método pelea se modifican estos valores:

Text

Description automatically generated

**Pelea y sus métodos**

Aquí hay una explicación un poco más detallada del método pelea, el cual se ejecuta tantas veces se llame, pero mientras el número de peleas no sea mayor a 10, en la décima pelea y en las subsecuentes arrojará un mensaje diciendo que el personaje ya no puede seguir peleando.

****

**Text

Description automatically generated**

**Ataque y sus métodos**

Para implementar ataque fue necesario que los poderes estuvieran en un arreglo de datos y con base en dicho arreglo se seleccionara un poder y se calculara el nivel de ataque, entonces para el ataque fue necesario diseñar un método auxiliar que regresara un poder aleatorio, aunque el seleccionar un ataque aleatorio se pudo insertar dentro de la función Ataque decidí separarlo para que fuera más claro y además para practicar el cómo interactúan los métodos dentro de una clase.

Método que regresa un poder aleatorio:

Text

Description automatically generated

El método Ataque que regresa el daño real y el poder que utilizó:

Text

Description automatically generated

**Interacción con el programa**

Se ingresan los datos del personaje Wonder Woman utilizando sus métodos Set y Get:

Text

Description automatically generated

Text

Description automatically generated

Se imprime la ficha técnica:

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Ahora se imprime el estado general sin ninguna pelea:

Text

Description automatically generated

Text

Description automatically generated

Ahora se realiza una pelea y se determina el ataque:

Text

Description automatically generated



Ahora imprimimos de nuevo el estado general para ver qué valores aumentaron y cuales disminuyeron:



Text

Description automatically generated

De nuevo una pelea y un ataque:

Text

Description automatically generated



De nuevo el estado general de la pelea:



Text

Description automatically generated

Esta secuencia ha sido dejada en el código con el fin de mostrar la evidencia de estas capturas y del correcto funcionamiento del programa.

A continuación muestro la captura general de la información para visualizar la fecha y hora del sistema:

Text

Description automatically generated

Adicional y con la única intención de hacer una comprobación del desarrollo de la actividad le proporciono la liga de mi github con la parte del código correspondiente a la actividad y donde es posible corroborar la hora de subida que fue algunos minutos después de la del sistema:

<https://github.com/JanoBourian/csharp_topics/tree/main/programacion_net_II/U1_A1/SuperHeroesApp>

# **CONCLUSIONES**

En esta actividad pude aprender a implementar métodos set y get de una manera rápida y sencilla para así evitar que los datos de la clase esté expuestos y siempre manipular por medio de un controlador de acceso. Aunque en este caso los métodos fueron muy simples es cierto que se pueden crear set y gets más complejos con validaciones y con sentencias try/catch para realizar validaciones más complejas. Por otro lado aprendí a utilizar constructores con la información básica y necesaria, en este caso consideré que el nombre del personaje era lo mínimo necesario para implementar la clase.

Por último, como mencioné al inicio de la actividad esto podría ser una oportunidad para diseñar personaje de un juego, por lo que la abstracción se vuelve de suma importancia y la implementación correcta de los métodos es relevante ya que de ellos es posible heredar a nuevas clases con nuevas funcionalidades, pero de momento la clase Personaje puede ser considerada el inicio o la base de futuros desarrollos, ya que como sabemos lo ideal es reutilizar código, pero dicho código debe ser creado con base a patrones y estar probado.

En la actividad no tuve la oportunidad de realizar alguna sobre carga de operadores, sin embargo según la temática y el material proporcionado me ha sido posible aprender más al respecto sobre la sobrecarga, para qué utilizarla y cómo implementarla.

# **REFERENCIAS BIBLIOGRÁFICAS**

C. Wille; C#, Prentice Hall. 2001. ISBN 0-672-32037-1.

Deitel (2004). “Instructor Manual for Simply C#”. Pearson Education

Tanenbaum, A. (2009). “Sistemas Operativos Modernos”. 3 ed. México. Pearson Education

UnADM (2021) “Unidad 1. Introducción a .net”. Recuperado el día 27 de Septiembre de 2022 de https://campus.unadmexico.mx/contenidos/DCEIT/BLOQUE1/DS/05/DPRN1/U1/descargables/DPRN1\_U1\_Contenido.pdf