Reporte de lectura – Capítulo 7

En este capítulo, el capítulo 7, se nos introduce los conceptos de herencia y de polimorfismo. Nos explica primero la herencia, diciendo que es una de las características más poderosas y fundamentales de la programación orientada a objetos, de forma que con esta se pueden crear clases base que logran encapsular una funcionalidad común. También otras clases pueden derivarse de esta clase base y de este modo pueden heredar las propiedades y métodos de esta clase, además de algunas otras funcionalidades. De esta manera se puede decir que la herencia nos permite una manera de poder compartir código lo cual es crucial dentro de cualquier proyecto, ya que esto permite que podamos ahorrar trabajo y tiempo al querer por ejemplo realizar algún cambio a un sistema, permitiendo que un algoritmo pueda procesar diversas clases de entidades.

De manera que la herencia se podría decir que permite una relación entre dos clases, siendo estas la clase derivada y la clase base, donde la clase derivada consigue la forma de utilizar algunas propiedades o funcionalidades de la clase base. Esto quiere decir que la clase derivada "hereda" varias características que tiene la clase base. Un ejemplo de este podría representarse por ejemplo con una clase vehículo, existen diversos tipos de vehículos como un carro, un camión o hasta un avión, los cuales todos comparten varios aspectos como por ejemplo la velocidad y la aceleración. Si tenemos varias clases, tales como vehículo, carro y avión, las clases carro y avión podrían heredar de la clase vehículo, lo cual significa que estas clases pueden heredar la función aceleración, al igual que la función de velocidad y también de la misma forma los campos y propiedades de esta. Pero que estos puedan heredar algunas funciones de la clase base, esto no significa que los mismos no puedan definir sus propias funciones que no estén relacionadas con la clase base.

La segunda funcionalidad que se nos explica es el polimorfismo, con el cual a la clase base se le permite definir métodos que deben ser implementados por cualquiera de las clases derivadas. Además, se puede decir que este que se pueda tener una función que reciba un parámetro, tal como una clase base y poder pasarle al mismo objeto que sean instancias de clases derivadas de la misma clase base, al igual que si el método recibe como dicho parámetro una interfaz. Así se puede traspasar al método cualquier clase que pueda implementar dicha interfaz. De modo que podemos ver que el polimorfismo es cuando se recibe un parámetro que tiene varios tipos y este permite generalizar algoritmos para que estos puedan funcionar con distintos tipos.

Además de las clases bases y derivadas también nos informan acerca de las clases abstractas que se podría decir que estas vienen siendo una clase que no puede ser instanciada, estas serían de utilidad en casos en los cuales no se quiere que un usuario o varios usuarios instancie la clase base, sino que estas solo puedan instanciar las clases derivadas. Por otro lado, también tenemos las interfaces, las cuales facilitan realizar otro tipo de herencia, una clase base permite la implementación por defectos de algunos métodos, con las interfaces se pueden implementar un conjunto de miembros que las clases que lo implementan deben implementar y al igual que las clases abstractas, las interfaces no pueden ser instanciadas.

Luego de comprender qué son las herencias y polimorfismo, se explica con más detalle, además, acerca de las clases base y las clases derivadas, con relación a la clase base es simple y llanamente una clase que busca encapsular propiedades y métodos que podrán ser utilizadas por las clases derivadas del mismo tipo. Ahora con relación a las clases derivadas, estas no están limitadas a las propiedades y métodos de la clase base, ósea que estas no solo podrán tener las propiedades y métodos de la clase base, ya que estas puedan requerir métodos y propiedades adicionales que sean únicos para cada uno. Un ejemplo de esto puede ser con la clase vehículo, utilizada en el ejemplo anterior, aunque todos en todos los vehículos se miden la velocidad y aceleración, esto no quiere decir que todos tengan las mismas propiedades, como el avión, el cual a diferencia del carro como este va en el aire, una propiedad diferente de este sería la altura de este, que tan lejos se encuentra del suelo.

Ahora, por default todas las clases pueden ser heredadas, esto quiere decir que al momento de crear una clase que puede ser heredada, se debe de tomar en cuenta que estas no estén modificadas de una manera en la cual la clase derivada no pueda funcionar correctamente. Por esto se nos explica sobre otro tipo de clase que nos permite el no tener una cadena de herencias demasiada compleja de forma que resulte demasiado complicado de manejar. A esta clase se le llama de sellado o clase final. Al utilizar esta clase es posible crear nuevas clases, de las cuales el usuario sabe que de estas no se podrá derivar, ya que con esto podemos hacer que una clase sea no heredable y así se pueda evitar una complejidad que se salga de las manos del programador a la hora de alterar o modificar el código de la clase base.

En fin, este capítulo busca introducir dos de las características más importantes y fundamentales de la programación orientada a objetos, permitiendo saber cómo implementar las mismas a la hora de programar.

Referencias.

Clark, D. (2013). Beginning C# Object-Oriented Programming (2nd Ed.). New York: Apress.