Módulo 4: Clases y Objetos

1. Programación Orientada a Objetos

Conceptos de Programación Orientada a Objetos

Empecemos con un ejemplo:

Supón que estas en un estacionamiento caminando hacia tu auto, entonces tomas el control remoto y

oprimes un botón, en ese momento se desactiva la alarma de tu automóvil.

El control remoto es un objeto físico que tiene propiedades: peso, tamaño y también puede hacer cosas:

puede enviar mensajes al automóvil. No está claro como hace esto, pero no necesitas saberlo para poder usarlo; solo necesitas

saber qué botón oprimir para que esto ocurra.

Los botones son la **interface** del control remoto. Si conoces la interface de un objeto, lo puedes usar para realizar acciones sin

entender cómo trabaja.

El automóvil también es un objeto físico, tiene propiedades y realiza acciones, una de ellas es recibir un mensaje de un control

remoto para desactivar la alarma.

Gracias a la **abstracción** eres capáz de hacer que dos objetos interactúen entre sí, sin necesidad de entender cómo funcionan.

Esto también permite que uses otros controles remotos y otros automóviles de otras marcas y otros modelos.

La clave para desarrollar un sistema utilizando los conceptos de Orientación a Objetos es definir qué objetos forman parte del

sistema, crear abstracciones apropiadas para ellos y separar la implementación interna de su comportamiento externo.

Desde el punto de vista de la programación orientada a objetos, la abstracción consiste en ocultar los detalles irrelevantes para

que el objeto pueda ser comprendido y utilizado por el usuario. Por un lado, esto disminuye la complejidad permitiendo que el

usuario utilice los datos del objeto sin necesidad de conocer detalles, y por otro lado protege los datos evitando que el usuario

pueda llegar a ellos y modificarlos sin utilizar las funciones diseñadas para ello, afectando con esto la confiabilidad del sistema.

Pero, ¿Qué es un objeto?

Un **objeto** puede ser alguna cosa física, como una pluma, un escritorio, un gato; pero también hay cosas abstractas que son

objetos, como un número o una cuenta bancaria.

Sabemos que algo es un objeto si tiene nombre, se puede describir en base a sus características o propiedades y es capaz de

hacer algo o comportarse de cierta manera.

Por ejemplo:

Pensemos en una cuenta bancaria sencilla, podemos describirla como un objeto de la siguiente manera:

- Clase: Cuenta Bancaria
- Propiedades: Nombre del dueño, saldo actual, tipo de cuenta.
- Comportamientos: depositar, retirar, consultar el saldo.

Los lenguajes orientados a objetos nos permiten crear abstracciones encapsulando las propiedades y los comportamientos en un

solo concepto llamado **clase**. Las propiedades (también llamadas atributos o datos miembro) que pertenecen a una clase se

definen como variables y los comportamientos (también llamados métodos o funciones miembro) se definen como funciones, y

éstos tienen la característica de tener acceso a los atributos privados de la clase.

La diferencia entre clase y objeto es sutil pero muy importante. La clase es el concepto abstracto, es una especie de molde para

crear objetos, define qué atributos y métodos tiene la clase. Por otro lado el objeto es la entidad concreta. A partir de una clase,

se pueden crear instancias, es decir objetos y cada objeto tiene valores distintos para sus atributos.

Por ejemplo:

Continuando con nuestro ejemplo de la clase cuenta bancaria,

La clase es Cuenta Bancaria.

Los objetos podrían ser

Cuenta 193	Cuenta 281
Nombre del dueño: Jorge López	Nombre del dueño: Laura Treviño
saldo actual: 5,248.00	saldo actual: 3,915.00
tipo de cuenta: ahorros	tipo de cuenta: chequera

Un objeto puede recibir **mensajes**, los cuales son para para pedirle que haga algo, o bien, que cambie alguna de sus propiedades.

Esto es equivalente a llamar a una función, solo que al llamar al método se tiene que hacer referencia al objeto específico en el

que se ejecutará dicho método.

Por ejemplo, se puede pedir a la cuenta bancaria 193 ejecutar el método depositar con el valor 100, lo cual se reflejará con un

cambio en el valor del atributo saldo actual, el nuevo valor para dicho atributo será 5,348.00. Pero nota que el valor del atributo

saldo actual de la cuenta 281 permanece sin cambio.

Funciones para Establecer y Consultar el valor de los atributos

Es muy común que las clases incluyan métodos para establecer (o escribir) y consultar (o leer) los valores de atributos privados de

la clase; a estos métodos se les llama en inglés "mutators" y "accesors" y se acostumbra poner los nombres "set" seguido del

nombre de la variable para métodos "mutators" y "get" seguido del nombre de la variable para los métodos "accesors"; así por

ejemplo deberíamos tener los siguientes métodos:

```
void setSaldoActual(double valor);      // método para establecer el valor de la variable saldo Actual.
double getSaldoActual();                // método para consultar el valor de la variable saldo Actual.
```

Tener estos métodos puede parecer equivalente a haber declarado el atributo como público;

sin embargo hay una pequeña

diferencia que es muy imortante. Si hacemos el atributo público cualquier programa cliente puede modificar su valor sin verificar

que el nuevo valor que se le asigne sea un valor apropiado; por otro lado, un método para establecer el valor de la variable puede

realizar la validación del dato que va a asignar al atributo y si no es un dato válido puede rechazarlo.

Ejercicio

A partir del siguiente texto, identifica los atributos y métodos para la clase Rectángulo.

En un paquete de dibujo uno de los objetos que permite dibujar en la pantalla es un rectángulo, en este paquete se puede

establecer el tamaño del rectángulo con el movimiento del mouse, se puede establecer el grosor de las líneas del rectángulo, así

como su color, se puede establecer si tiene o no un color de relleno y escoger dicho color y se puede posicionar en cualquier

coordenada de la pantalla. ver solución

Ligas sugeridas

http://www.cplusplus.com/doc/tutorial/

http://www.cs.wustl.edu/~schmidt/C++/

Regresar

Siguiente módulo