CODES

Capacitación Web – Programación orientada a objetos

codes
CONSULTORIA
Y DESARROLLO DE SISTEMAS

CMMI

ÍNDICE

ÍNDICE

- Es un *Paradigma* de lenguaje de programación
- Se basa en *modelar* el problema. ¿Existe un único modelo? ¿Hay modelos mejores otros? ¿Por qué es importante que sea un buen modelo?
- Se utilizan *objetos* para crear dicho modelo.

unCoche

* Acelera	acelerar()
* Frena	frenar()
* Se estaciona	estacionar()

- Estos objetos se podrán utilizar en los programas, por ejemplo en un programa que gestione un taller de coches se usarán objetos coche.

- Los programas Orientados a objetos utilizan muchos de estos para realizar las acciones que se desean realizar y ellos mismos también son objetos. Es decir, el taller de coches será un objeto que utilizará objetos coche, herramienta, mecánico, recambios, etc.

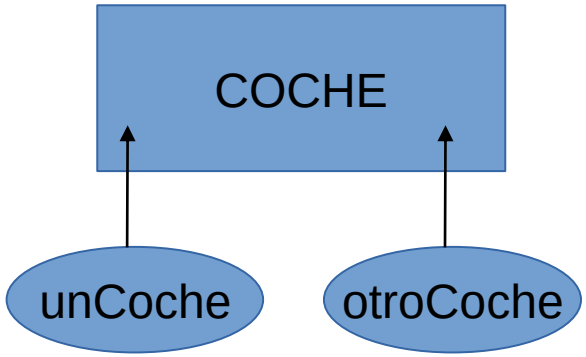
! Objeto: Representación de un ente del problema a resolver

ÍNDICE

- ¿Como creamos objetos? Mediante clases.

- Las clases son declaraciones de objetos, también se podrían definir como abstracciones de objetos. Esto quiere decir que la definición de un objeto es la clase.

- Cuando programamos un objeto y definimos sus características y funcionalidades en realidad lo que estamos haciendo es programar una clase.
En el ejemplo anterior en realidad hablábamos de la clase coche porque sólo estuvimos definiendo, aunque por encima, su forma.



! Clase: Objeto que representa una idea de un ente. Permite definir y crear objetos que representan entes concretos.

ÍNDICE

3. PROPIEDADES EN CLASES

- Las propiedades o atributos son las características de los objetos.

-Cuando definimos una propiedad normalmente especificamos su nombre y su tipo. Nos podemos hacer a la idea de que las propiedades son algo así como variables donde almacenamos datos relacionados con los objetos.

ÍNDICE

- Son las funcionalidades asociadas a los objetos. Cuando estamos programando las clases las llamamos métodos. Los métodos son como funciones que están asociadas a un objeto.

! **Mensaje: Lo que "sabe hacer" un objeto, una funcionalidad.**

! **Método: La implementación de un mensaje.**

! Programa: Objetos que colaboran entre si enviándose mensajes para resolver un problema.

ÍNDICE

- Los objetos son ejemplares de una clase cualquiera. Cuando creamos un ejemplar tenemos que especificar la clase a partir de la cual se creará. Esta acción de crear un objeto a partir de una clase se llama ***instanciar*** (que viene de una mala traducción de la palabra *instance* que en inglés significa ejemplar).

- Para crear un objeto se tiene que escribir una instrucción especial que puede ser distinta dependiendo el lenguaje de programación que se emplee, pero será algo parecido a esto.

```
Coche unCoche = new Coche();
```

- Con la palabra new especificamos que se tiene que crear una instancia de la clase que sigue a continuación. Dentro de los paréntesis podríamos colocar parámetros con los que *inicializar* el objeto de la clase coche.

- *Si estoy modelando una concesionaria de autos. ¿Es válido un coche sin color?*

ÍNDICE

Cuando tenemos un objeto sus propiedades toman valores. Por ejemplo, cuando tenemos un coche la propiedad color tomará un valor en concreto, como por ejemplo rojo o gris metalizado. El valor concreto de una propiedad de un objeto se llama estado.

Para acceder a un estado de un objeto para ver su valor o cambiarlo se utiliza el operador punto.

```
miCoche.patente= "ABC 123"
```

El objeto es **miCoche**, luego colocamos el operador punto y por último el nombre de la propiedad a la que deseamos acceder. En este ejemplo estamos cambiando el valor del estado de la propiedad del objeto a rojo con una simple asignación.

ÍNDICE

Un mensaje en un objeto es la acción de efectuar una llamada a un método. Por ejemplo, cuando le decimos a un objeto coche que se ponga en marcha estamos enviándole el mensaje "ponete en marcha".

Para enviar mensajes a los objetos utilizamos el operador punto, seguido del método que deseamos invocar.

```
miCoche.ponerseEnMarcha()
```

En este ejemplo pasamos el mensaje *ponerseEnMarcha()*. Hay que colocar paréntesis igual que cualquier llamada a una función, dentro irían los parámetros.

ÍNDICE

8. HERENCIA

- Hay ideas mas abstractas que otras, lo mismo pasa con las clases. Por ejemplo Pensemos en los números.

- La herencia es una relación entre clases, permite organizarlas según su grado de generalidad.

- Es la relación entre una clase general y otra clase más especifica.

- La herencia es uno de los mecanismos de la programación orientada a objetos, por medio del cual una clase se deriva de otra, llamada entonces clase base o clase padre, de manera que extiende su funcionalidad.

ÍNDICE

- Si estamos modelando un sistema bancario con sus **clientes**. ¿ Agregarían una propiedad **colorDeOjos** a la representación de cliente?

- Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos y cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción. El proceso de abstracción consiste en seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real.

- La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar.

! Abstracción en modelos: Proceso de eliminar las características no relevantes de los entes del problema a resolver para su representación.

ÍNDICE

- Pensando en el coche ¿ Le indico a cada rueda gire de manera uniforme para así desplazarme? O solo lo acelero?...

```
public class Coche(){

    public Rueda rueda1 { get; set; }
    public Rueda rueda2 { get; set; }
    public Rueda rueda3 { get; set; }
    public Rueda rueda4 { get; set; }

    public void acelerar(){
        this.rueda1.mover();
        this.rueda2.mover();
        this.rueda3.mover();
        this.rueda4.mover();
    }

}
```

```
Coche unCoche = new Coche();
unCoche.rueda1.mover();
unCoche.rueda2.mover();
unCoche.rueda3.mover();
unCoche.rueda4.mover();
```

```
Coche unCoche = new Coche();
unCoche.acelerar();
```

! Encapsulamiento: La idea de no exponer colaboradores internos de un objeto hacia los usuarios del mismo

! Intentamos no romper el encapsulamiento

ÍNDICE

Supongamos que tenemos una patrulla de policía, es similar a un coche pero además permite encender y apagar la sirena… ¿Cómo lo implementaríamos?

! Herencia: Relación entre 2 o mas clases según el grado de generalidad de las mismas

13. EJEMPLO PRÁCTICO – CLASE ANIMAL

```
public abstract class Animal
{
    public string Nombre { get; set; }

    public abstract void Correr();

    public Animal()
    {
        Nombre = "Pepe";
    }

    public Animal(string nombre)
    {
        Nombre = nombre;
    }
}
```

13. EJEMPLO PRÁCTICO – CLASE ANIMAL

```
 //La clase Perro hereda de la clase Animal
public class Perro : Animal
{
    public Perro():base()  { }

    public Perro(string nombre)
      : base(nombre)
    {
    }

    public override void Correr()
    {
        //Hacer algo...
    }

    public void Ladrar()
    {
        System.Console.println("Guau!");
    }
}
```

13. EJEMPLO PRÁCTICO – CLASE ANIMAL

```
public class Gato : Animal
{
    public Gato():base()  { }

    public Gato(string nombre)
      : base(nombre)
    {
    }

    public override void Correr()
    {
        //Hacer algo...
    }

    public void Maullar()
    {
        System.Console.println("Miau!");
    }
}
```

ÍNDICE

Tenemos un requerimiento para dibujar figuras, pueden ser círculos, cuadrados o triángulos, se propone la siguiente solución:

```
public class Cuadrado{

}

public class Triangulo{

}

public class Circulo{

}

public class Dibujador{

    public void dibujarCuadrado(Cuadrado unCuadrado){
        ...
    }

    public void dibujarTriangulo(Triangulo unTriangulo){
        ...
    }

    public void dibujarCirculo(Circulo unCirculo){
        ...
    }

}
```

```
public class Figura{
    public abstract void dibujar(){
      ...
    }
}
```

```
public class Cuadrado : Figura {
    public void dibujar(){
      ...
    }
}

public class Triangulo : Figura {
    public void dibujar(){
      ...
    }
}

public class Circulo : Figura {
    public void dibujar(){
      ...
    }
}
```

```
public class Dibujador{

    public void dibujar(Figura unaFigura){
        unaFigura.dibujar();
    }
}
```

! Polimorfismo: Relación entre dos objetos y un conjunto de mensajes. Decimos que dos objetos son polimórficos entre si con respecto a un conjunto de mensajes si esos objetos pueden responder a todos los mensajes del conjunto.

¿El polimorfismo tiene algo que ver con la herencia?

FIN

¿Dudas?

¿Preguntas?

codes
CONSULTORIA
Y DESARROLLO DE SISTEMAS

CMMI