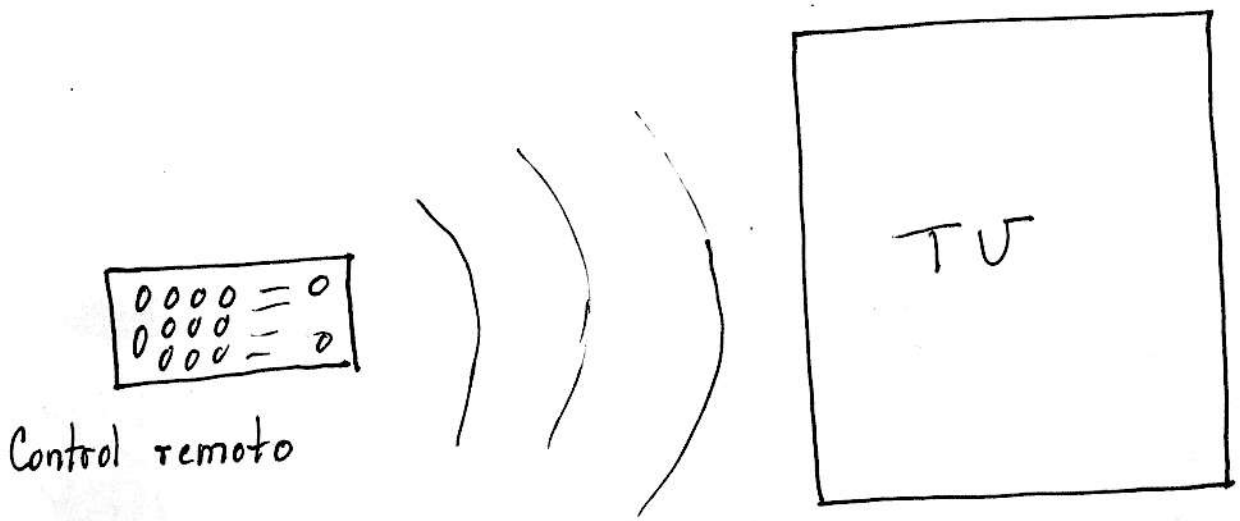
Encapsulamiento y ocultamiento de la información

Es una de las propiedades de la POO que permite ocultar los detalles de un objeto. Esto evita que el objeto sea manipulado a través de operaciones distintas a las definidas. Se puede ver el encapsulamiento como una caja negra de la cual solo conocemos su interfaz pero desconocemos cómo funciona.

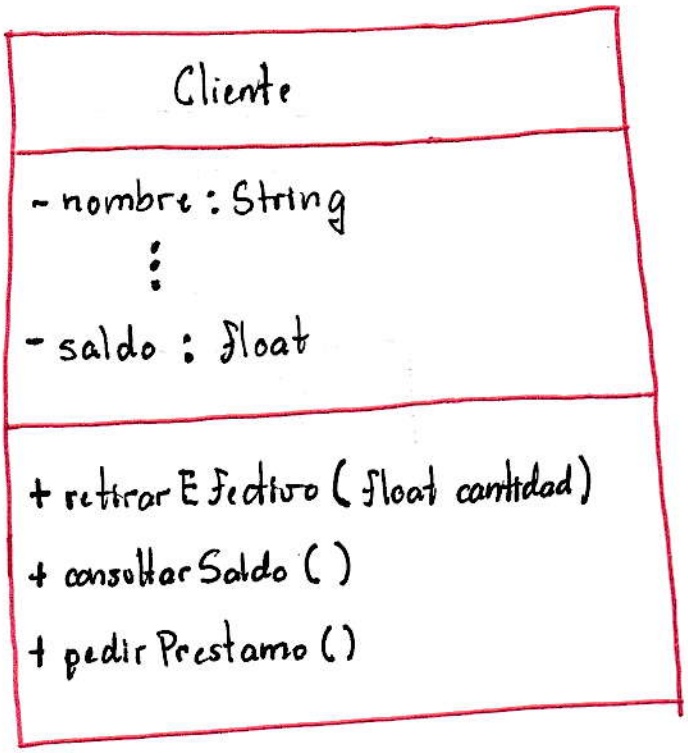


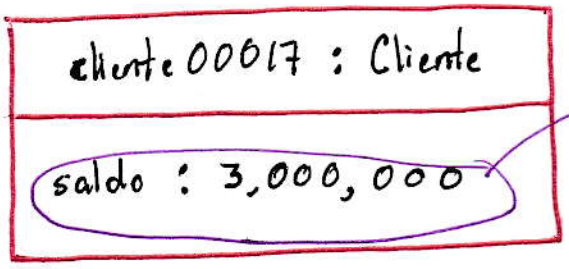
Podemos utilizar la interface del control remoto (botones) para que interactúe con la televisión pero desconocemos como funciona. A su vez también desconocemos cómo funciona la tv por dentro.

Otros ejemplos: - un smartphone. ¿Qué pasa si conocemos su funcionamiento interno?

- Un cliente de un banco.

Clase 18-sept-2017





¿Qué pasa si cualquier objeto no autorizado modifica directamente el atributo saldo?

Visibilidad

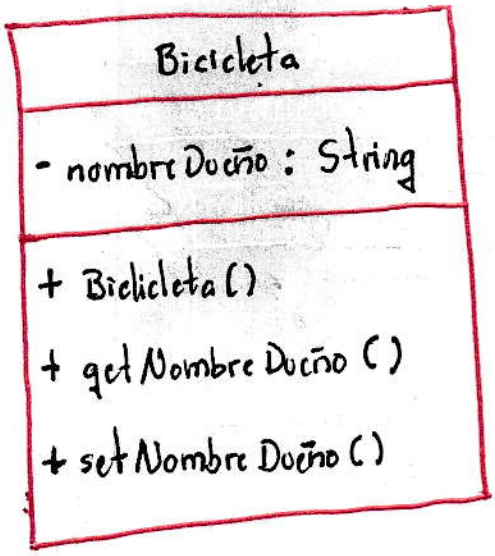
+ : pública, se puede acceder a ese miembro de la clase desde fuera.

- : privada, solo se puede acceder desde la propia clase.

: protegido, solo se puede acceder desde la propia clase y desde las clases que heredan de ella.

Ejemplo : Clases en UML y Java

UML



Java

```
public class Bicicleta {

    private String nombreDueño;

①   // Constructor
    public Bicicleta() {
        this.nombreDueño = "desconocido";
    }

②   // "getter" y "setter"
    public String getNombreDueño() {
        return this.nombreDueño;
    }
                ③
    public void setNombreDueño(String
      nombreDueño) {
        this.nombreDueño = this.nombreDueño;
    }

}
```

① Constructores

Los constructores son un tipo especial de métodos que son utilizados para inicializar el estado de un objeto de la clase correspondiente.

Dentro del main podemos inicializar un objeto de tipo Bicicleta de una sola forma.

```
public static void main (String[] args) {

    Bicicleta bici = new Bicicleta();

}
```

Nombre de la clase

Nombre del objeto (instancia) de tipo Bicicleta

Constructor

Al ejecutar el programa, el objeto 'bici' de tipo Bicicleta estará en memoria. Conforme el atributo y su valor dado por el constructor el atributo nombreDueño del objeto 'bici' será 'desconocido'.

Una clase puede tener más de un constructor. Así que podemos añadir el siguiente.

```
public Bicicleta (String nombreDueño) {
        this.nombreDueño = nombreDueño;
}
```

De esta forma podremos inicializar un objeto (instanciar) de dos formas.

```
Bicicleta bici1 = new Bicicleta();
Bicicleta bici2 = new Bicicleta ("Juan Pérez");
```

② Los llamados métodos 'getter' y 'setter' son métodos que permiten obtener y asignar respectivamente un valor a un atributo de la clase. Esto permite mantener el encapsulamiento por que sirven de interface entre los atributos de un objeto y otros objetos. En un control remoto el botón 'info' sirve como getter del contenido de algún canal, y el botón 'favorite' permite asignar canales a nuestros favoritos, este actúa como un setter.

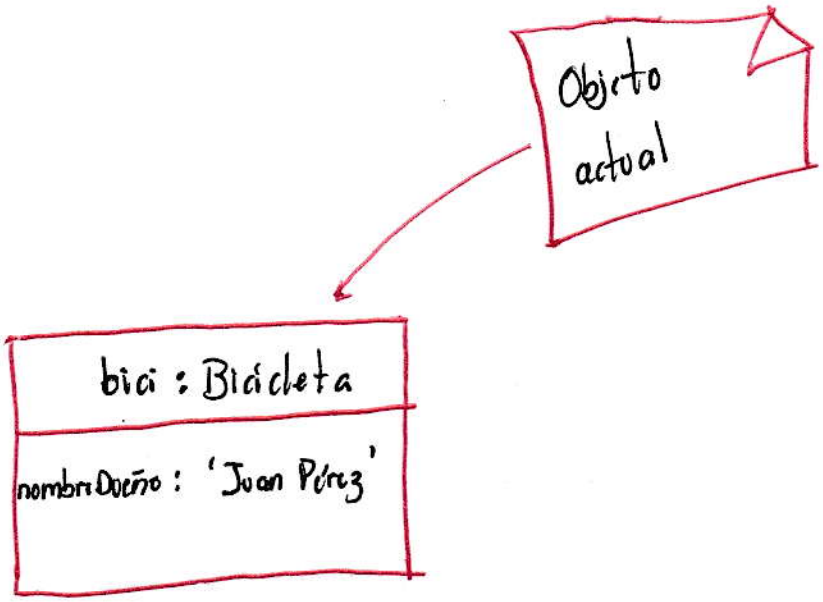
```
setNombreVariable(          ) {
    return
}

getNombreVariable (         ) {

}
```

③ Palabra reservada 'this'. Hace referencia al objeto actual.

Si no la utilizamos dentro de la definición de la clase, las referencias a sus atributos estarán implícitas. Al usar "this" estamos haciendo más claras dichas referencias.



-Si invocamos bici.getNombreDueño(); entonces nos devolverá el valor de this.nombreDueño. Es decir, el valor del atributo nombreDueño del objeto en memoria 'bici'.

- Si invocamos bici.setNombreDueño("Pepita Jiménez"); entonces el valor del atributo nombreDueño del objeto actual 'bici' será "Pepita Jiménez".