

Definiciones

Clases y objetos

La programación orientada a objetos es un paradigma de programación que se basa, como su nombre indica, en la **utilización de objetos**. Estos objetos también se suelen llamar instancias.

Un **objeto** en términos de POO no se diferencia mucho de lo que conocemos como un objeto en la vida real. Pensemos por ejemplo en un coche. Nuestro coche sería un coche más de tantos que hay. Todos esos coches son objetos concretos que podemos ver y tocar.

Tanto mi coche como el coche del vecino tienen algo en común, ambos son coches. En este caso mi coche y el coche del vecino serían instancias (objetos) y Coche (a secas) sería una clase. La palabra **Coche** define algo genérico, es una abstracción, no es un coche

elementos que tienen una serie de propiedades; este conjunto de propiedades se denominan

de cualidades, por ejemplo Coche.

reción de una clase, por ejemplo mi coche.

en los objetos de una clase, por ejemplo para la , modelo, color y número de plazas.

ben con la primera letra en mayúscula mientras izan con una letra en minúscula. Por ejemplo, la e el objeto "mi coche" se podría escribir como

Definiremos cada clase en un fichero con el mismo nombre más la extensión .java, por tanto, la definición de la clase Coche debe estar contenida en un fichero con nombre Coche.java



- **OBJETO** = INSTANCIA (minúsculas)
- **Plantilla** = en mayúscula (la primera)
- Propiedades = **atributos o variables de instancia**
- No tiene main.
- El main es otra clase distinta
- New class.java

Clase

Concepto abstracto que denota una serie de cualidades, por ejemplo Coche.

Instancia

Objeto palpable, que se deriva de la concreción de una clase, por ejemplo mi coche.

Atributos

Conjunto de características que comparten los objetos de una clase, por ejemplo para la clase coche tendríamos matrícula, marca, modelo, color y número de plazas.

En Java, los nombres de las clases se escriben con la primera letra en mayúscula mientras que los nombres de las instancias comienzan con una letra en minúscula. Por ejemplo, la clase coche se escribe en Java como Coche y el objeto "mi coche" se podría escribir como miCoche.

En Java, los nombres de las clases se escriben con la primera letra en mayúscula mientras que los nombres de las instancias comienzan con una letra en minúscula. Por ejemplo, la clase coche se escribe en Java como Coche y el objeto "mi coche" se podría escribir como miCoche.

Definiremos cada clase en un fichero con el mismo nombre más la extensión .java, por tanto, la definición de la clase Coche debe estar contenida en un fichero con nombre Coche.java



EJEMPLO

Vamos a definir a continuación la clase Libro con los atributos isbn, autor, titulo y numeroPaginas.

```
/*Definición de la clase Libro*/
public class Libro {
    // atributos
    String isbn;
    String titulo;
    String autor;
    int numeroDePaginas;
}
```

A continuación creamos varios objetos de esta clase.

```
/*Programa que prueba la clase Libro*/
public class PruebaLibro {
    public static void main(String[] args) {
        Libro lib = new Libro();
        Libro miLibrito = new Libro();
        Libro quijote = new Libro();
    }
}
```

Hemos creado tres instancias de la clase libro: lib, miLibrito y quijote.

Las variables de instancia (atributos) determinan las cualidades de los objetos.

Salvador Sánchez Fernández

(protegido) y private (privado).

En la siguiente tabla se muestra desde dónde es visible/accesible un elemento (atributo o método) según el modificador que lleve.

	En la misma clase	En el mismo paquete	En una subclase	Fuera del paquete
private	✓	✗	✗	✗
protected	✓	✓	✓	✗
public	✓	✓	✓	✓
sin especificar	✓	✓	✗	✗

Figura Ámbito de los elementos de una clase.

Por ejemplo, un atributo private solamente es accesible desde la misma clase, sin embargo, a un método protected se podrá acceder desde la misma clase donde esté definido, desde otro fichero dentro del mismo paquete o desde una subclase.

Como regla general, se suelen declarar private los atributos o variables de instancia y public los métodos.

```
/*Definición de la clase Animal */
public class Animal {

    private String tipo; //doméstico,salvaje
    private String sexo; //macho,hembra

    public Animal (String tipo, String sexo) {
        this.tipo = tipo;
        this.sexo = sexo;
    }

    public Animal () {
        this.tipo = "salvaje";
        this.sexo = "macho";
    }
}
```

Salvador Sánchez Fernández

Ámbito/visibilidad de los elementos de una clase	12
Herencia	14
Sobrecarga de métodos.....	17
Atributos y métodos de clase (static)	19
Método equals	21
Arrays de objetos	22
Composición de otras clases	24
Clases internas	26
<i>Ejemplo: Huevo, Yema y Clara</i>	26
<i>Ejemplo: Automóvil</i>	27
<i>Ejemplo: Lista de objetos de distinto tipo</i>	28
Tipo enumerado.....	31
Wrappers.....	32
Clase abstracta (abstract)	33
Interfaces	35
Final en herencia.....	39

Documento activo

Salvador Sánchez Fernández

