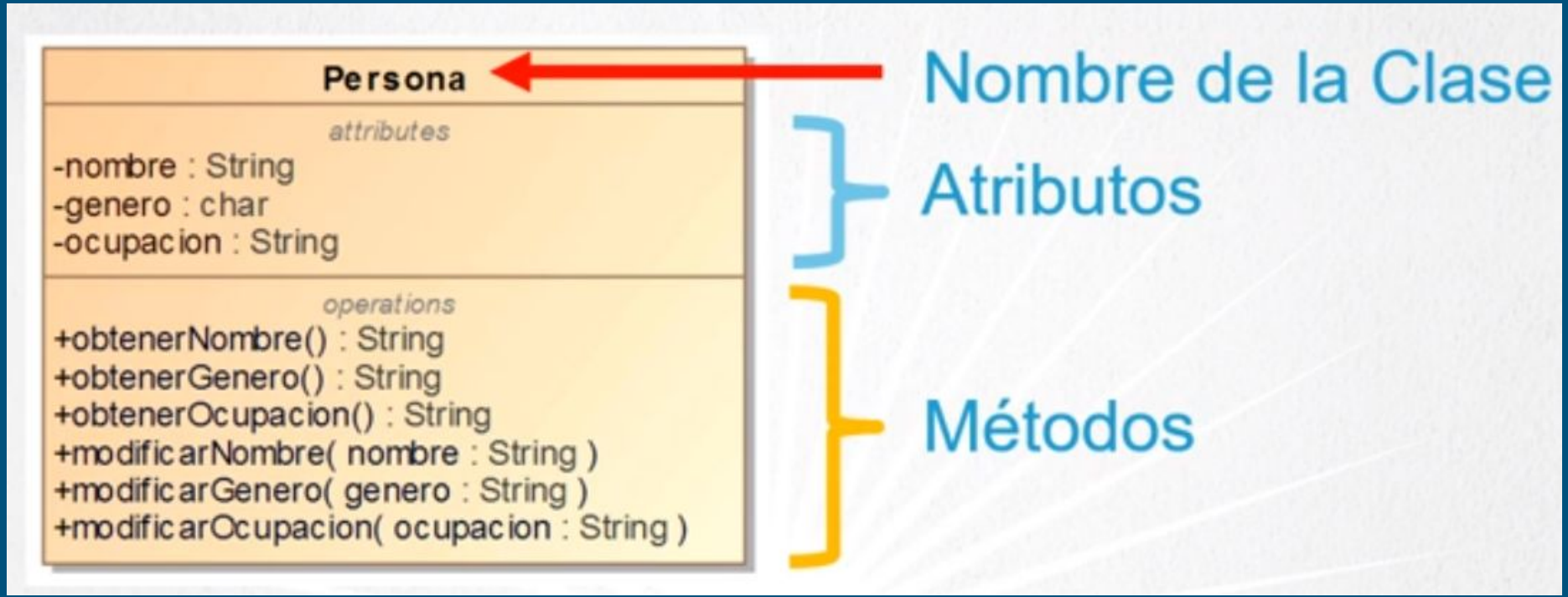
Fundamentos de Java

Módulo 1 - Clase #04
Java Programmer SE 8

Abstracción

Resaltar la parte más representativa de algo, ignorando detalles para centrarse en lo principal.

Diagrama General de una Clase



Encapsulamiento

Es un mecanismo que consiste en organizar datos y métodos de una estructura, conciliando el modo en que el objeto se implementa, es decir, evitando el acceso a datos por cualquier otro medio distinto a los especificados.

Modificadores de Acceso

Modificador	Descripción
public	Los elementos pueden ser accedidos desde cualquier clase o instancia sin importar el paquete o procedencia de ésta.
private	Los elementos pueden ser accedidos únicamente por la misma clase.
protected	Permite acceso a los elementos desde la misma clase, clases del mismo paquete y clases que hereden de ella (incluso en diferentes paquetes).
default	Permite que tanto la propia clase como las clases del mismo paquete accedan a dichos componentes.

Ejemplo

```java
public class PruebaEncapsulamiento{//Clase1

    public static void main(String[] args) {
        Persona p1 = new Persona("Juan");
        //Marca error,
        //no se puede acceder directamente un atributo privado desde otra clase
        p1.nombre = "Pedro";
        //Si es posible acceder a un método o atributo publico desde otra clase
        p1.obtenerNombre();
    }
}
```

```java
class Persona {//Clase2

    private String nombre; //Uso de private en un atributo

    public Persona(String nombre) { //Uso de public en un método
        this.nombre = nombre;
    }

    public String obtenerNombre() { //Uso de public en un método
        return nombre;
    }
}
```

Setters / Getters

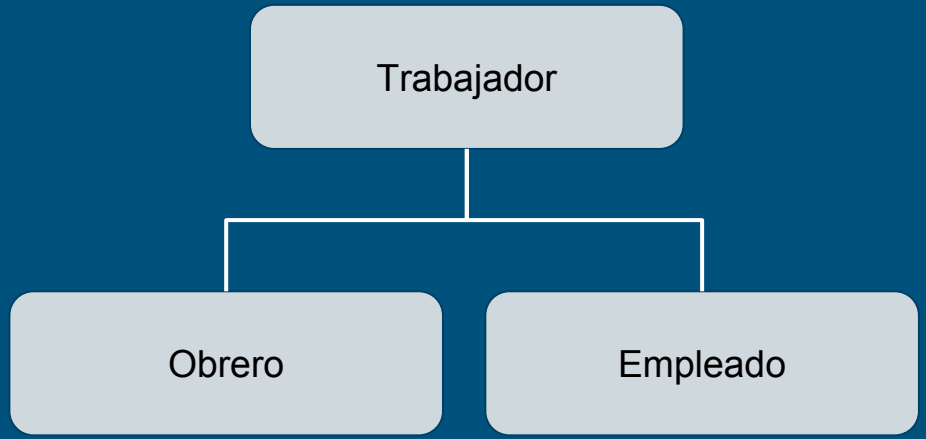
```java
public class PruebaEncapsulamiento {

    public static void main(String[] args) {
        //Creamos el objeto
        Persona p1 = new Persona();
        //Modificamos el atributo nombre
        p1.setNombre("Juan");
        //Accedemos al atributo nombre
        System.out.println("Nombre:" + p1.getNombre());
    }
}
```

```java
class Persona {
    //Atributo privado
    private String nombre;
    //Método publico para acceder al atributo nombre
    public String getNombre(){
        return nombre;
    }
    //Método publico para modificar al atributo nombre
    public void setNombre(String nombre){
        this.nombre = nombre;
    }
}
```

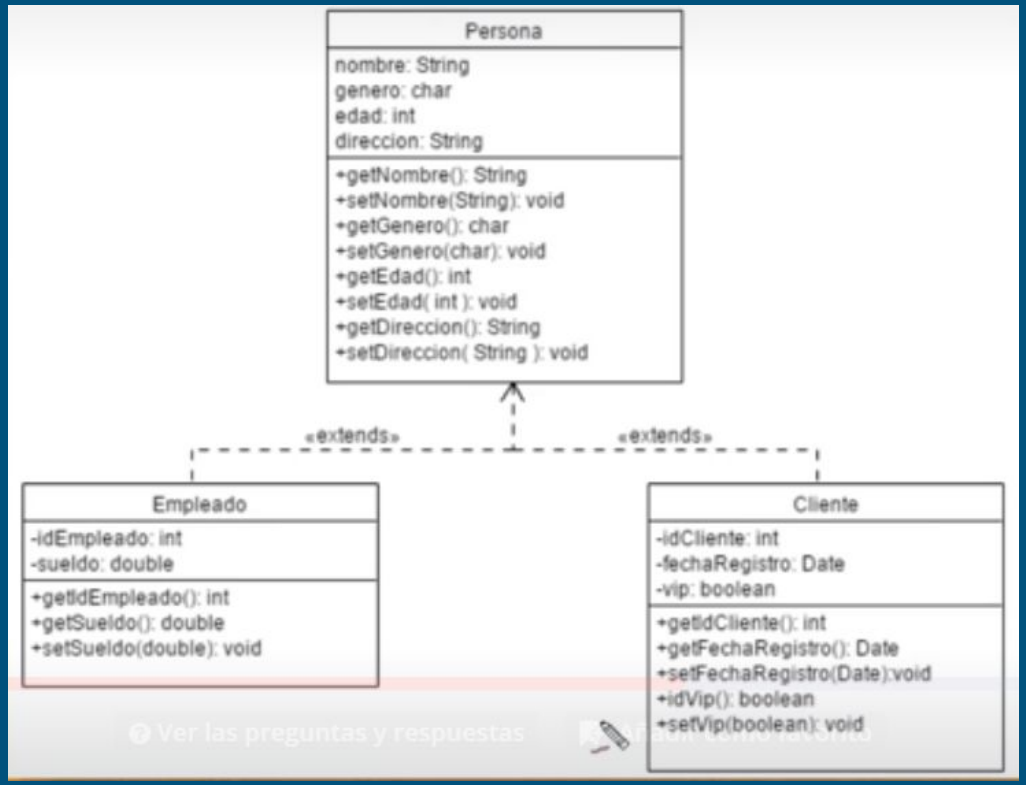
Diagrama de Objetos

Herencia

- Utilizar la estructura o modelo de una clase padre.
- La clase hija se puede acceder a los métodos y atributos de la clase padre.
- Se usa la palabra reservada “extends”, seguido del nombre de la clase de la cual se quiere heredar.
- Representar comportamiento común.
- Evitar duplicación de código.
- Es un característica de la POO.
- Se utiliza la palabra ”super” para acceder a los componentes del padre.



Ejemplo



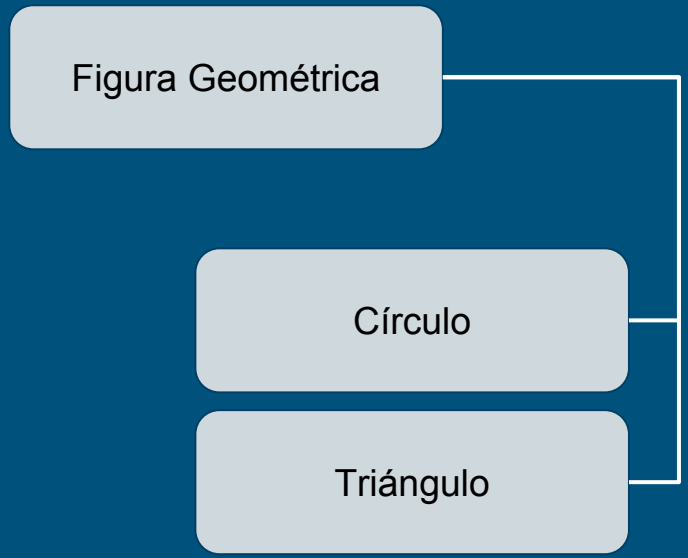
Clase Persona
+ getNombre()
+ setNombre()
+ getGenero(), etc

Clase Empleado
+getIdEmpleado()
+getSueldo(), etc

Polimorfismo

Es la capacidad que tienen los objetos de comportarse de múltiples formas. Eso significa que para aplicarlo se debe programar de manera general en vez de hacerlo de manera específica.

El objetivo de aplicar polimorfismo es el de poder crear un árbol de objetos en un proyecto, de tal manera que cada instancia de objeto que secree pueda pertenecer a un mismo tipo (siempre y cuando el análisis del desarrollo de los objetos lo necesite).



Extras

final

- Variable tipo constante.
- No admite cambios después de su declaración y asignación de valor.
- Determina que un atributo no puede ser sobreescrito o redefinido.
- Toda constante declarada con final ha de ser inicializada en el mismo momento de ser declarada.
- Se usa como palabra clave en otro contexto: una clase final (final) es aquella que no puede tener clases que la hereden.

static

- Los atributos miembros de una clase pueden ser atributos de clase o atributos de instancia.
- Ocupa un único lugar en memoria.
- Si no se usa static, el sistema crea un lugar nuevo para esa variable con cada instancia (la variable es diferente para cada objeto).
- Cuando usamos “static final” se dice que creamos una constante de clase, un atributo común a todos los objetos de esa clase.

Ejemplo