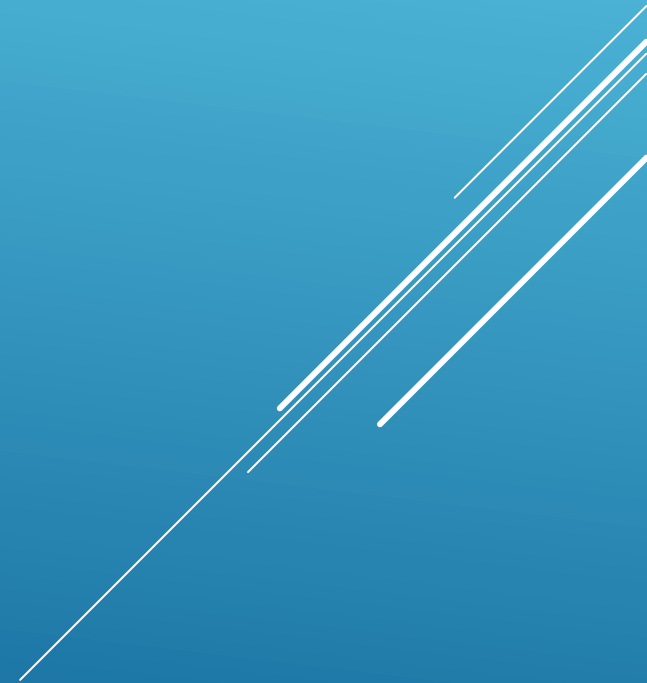



# PROGRAMACION ORIENTADA A OBJETOS **HERENCIA**

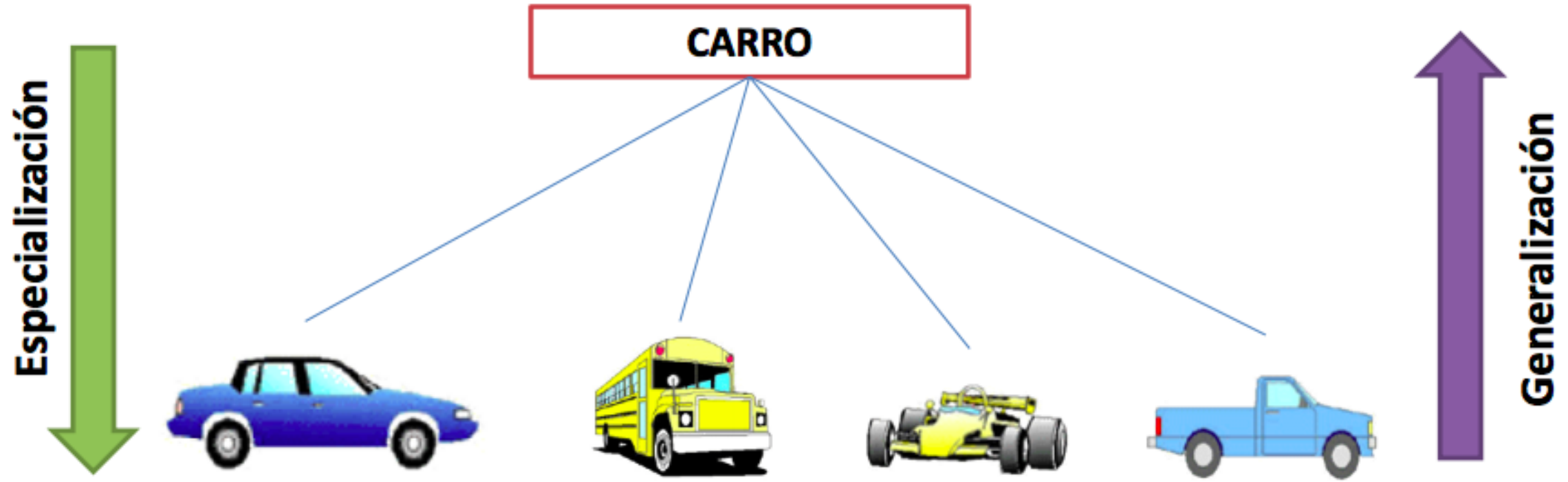


- *La herencia es un mecanismo que permite la definición de una clase a partir de la definición de otra ya existente. La herencia permite compartir automáticamente métodos y datos entre clases, subclases y objetos.*

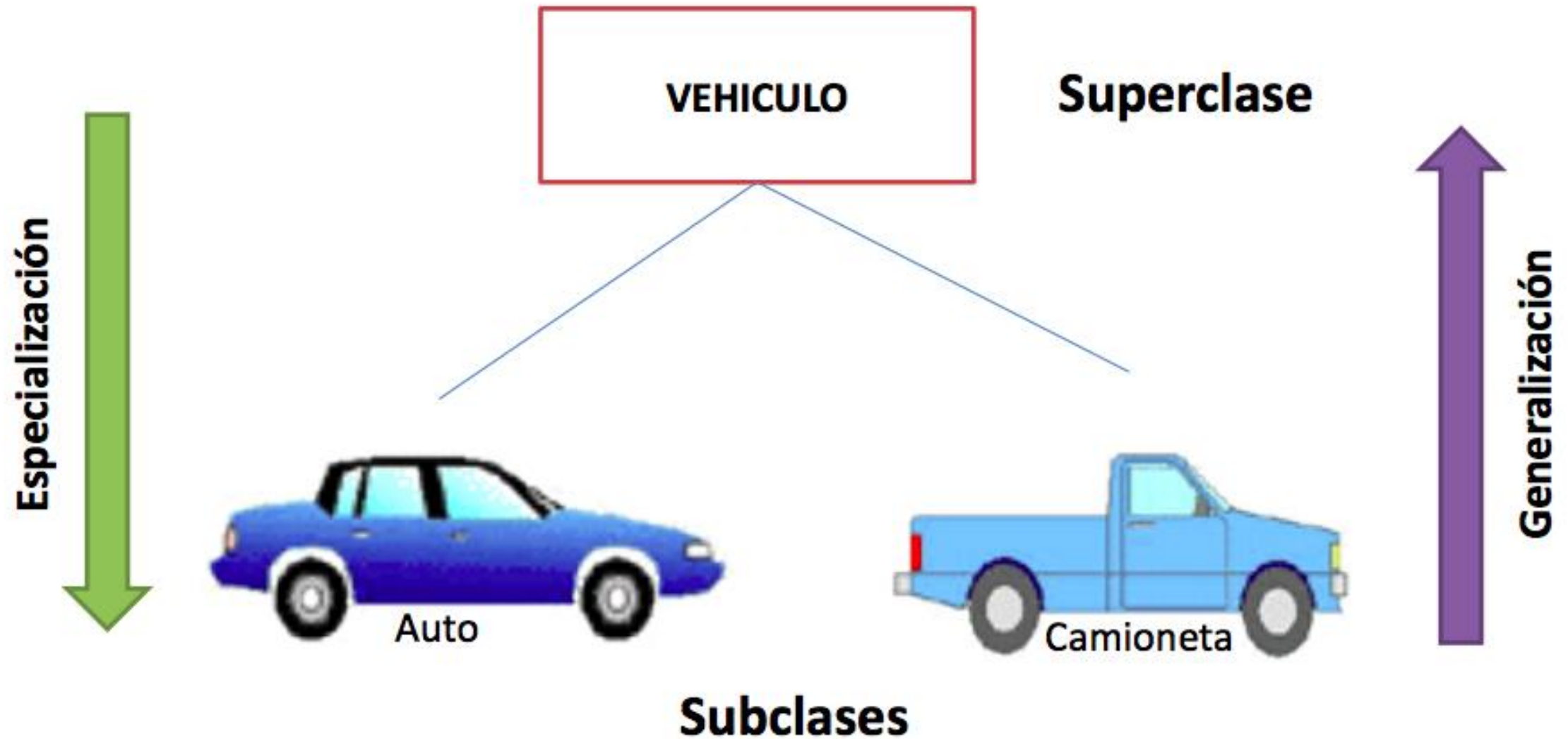


- ▶ Tipo especial de relación entre clases
  - ▶ Es uno de los aspectos que distinguen el Paradigma de Orientación a Objetos frente a otros paradigmas
  - ▶ Mecanismo que, bien utilizado, facilita la modificabilidad y reutilización de los diseños y el código
- 
- A series of white diagonal lines of varying lengths and thicknesses, located in the bottom right corner of the slide, creating a modern, abstract graphic element.

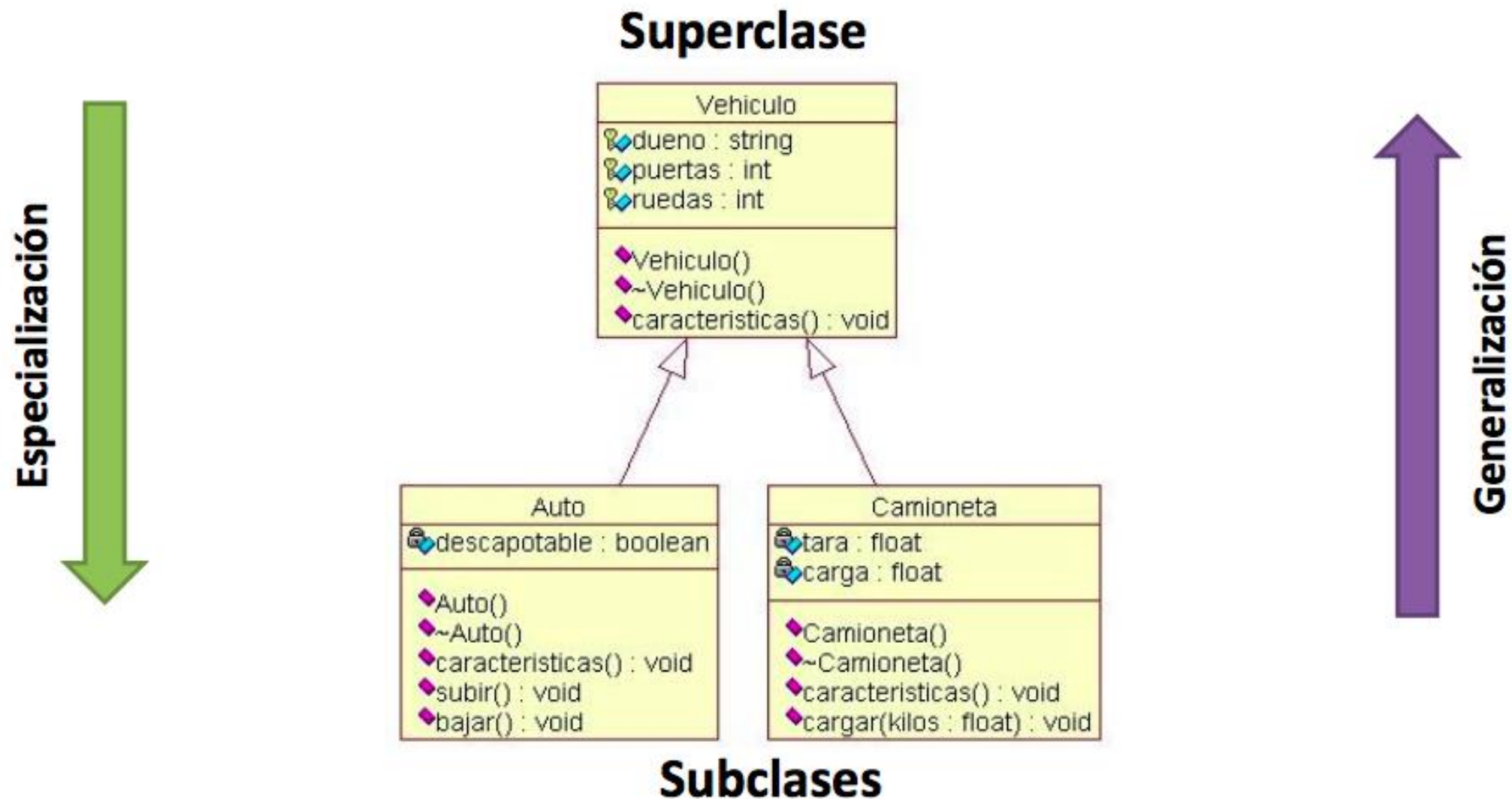
Se conoce como HERENCIA, las jerarquías de estructura de clase (**generalización/especialización**) en la cual se define una relación entre clases, en donde una clase comparte la estructura o comportamiento definido en una o más clases.



**Herencia Simple:** Cada subclase o clase derivada tiene una y solo una superclase o clase base directa.



**Herencia Simple:** Cada subclase o clase derivada tiene una y solo una superclase o clase base directa.





## Modificadores de Acceso



```
graph TD; A[Modificadores de Acceso] --> B[Private]; A --> C[Protected]; A --> D[Public];
```

### Private

Los miembros private de una clase son accesibles sólo dentro de la misma clase. Los miembros private de una superclase no son heredados por sus subclases

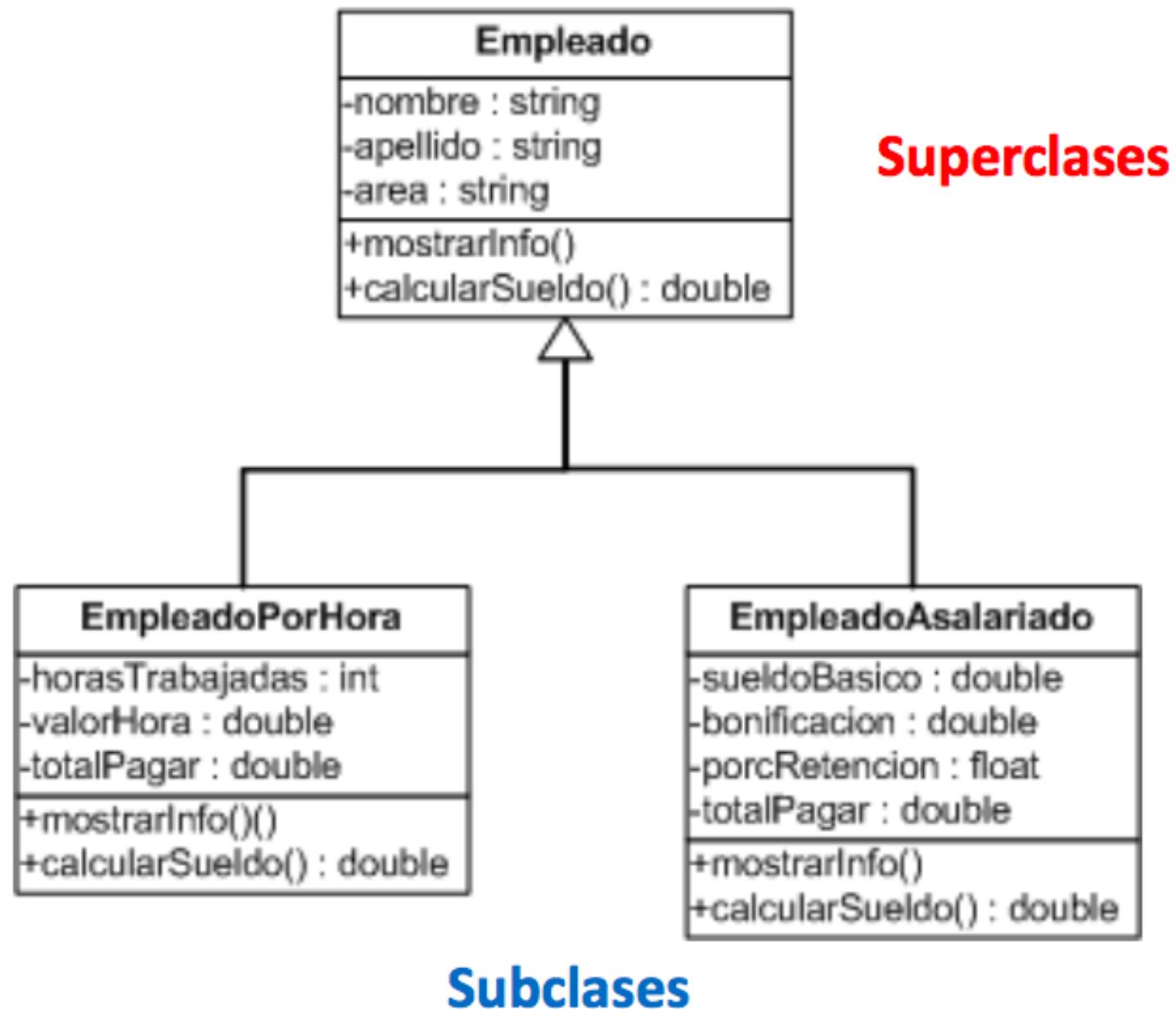
### Protected

Los miembros protected de una superclase pueden ser utilizados por los miembros de esa superclase, por los miembros de sus subclases y por los miembros de otras clases en el mismo paquete

### Public

Los miembros public de una clase son accesibles en cualquier parte en donde el programa tenga una referencia a un objeto de esa clase, o una de sus subclases

# HERENCIA – USO DE **extends**, **super**, **final**





# HERENCIA – USO DE **extend**, **super**, **final**

```
1 package ejercicioherencia;
2
3 public class Empleado {
4
5     private String nombre;
6     private String apellido;
7     private String area;
8
9     public Empleado(String nombre, String apellido, String area) {
10         this.nombre = nombre;
11         this.apellido = apellido;
12         this.area = area;
13     }
14
15     public String getNombre() { return nombre; }
16     public void setNombre(String nombre) {this.nombre = nombre;}
17     public String getApellido() { return apellido;}
18     public void setApellido(String apellido) {this.apellido = apellido;}
19     public String getArea() {return area;}
20     public void setArea(String area) {this.area = area;}
21
22     public void mostrarInfo() {
23         System.out.println("[ INFORMACION DE EMPLEADO ]");
24         System.out.println("-----");
25         System.out.println("Nombre: "+this.nombre);
26         System.out.println("Apellido: "+this.apellido);
27         System.out.println("Area: "+this.area);
28         System.out.println("Sueldo: "+this.calcularSueldo());
29     }
30     public double calcularSueldo() {
31         return 0;
32     }
33 }
```

**Superclase**  
**Empleado**

```

1 package ejercicioherencia;
2
3 public class EmpleadoPorHora extends Empleado { // Hereda de Empleado
4     private int horasTrabajadas;
5     private double valorHora;
6     private double totalPagar;
7
8     public EmpleadoPorHora(int horasTrabajadas, double valorHora, String nombre, String apellido, String area) {
9         super(nombre, apellido, area); // Ejecuta constructor de superclase Empleado
10        this.horasTrabajadas = horasTrabajadas;
11        this.valorHora = valorHora;
12    }
13
14    @Override // Sobreescribe Metodo
15    public void mostrarInfo() {
16        System.out.println("[ INFORMACION DE EMPLEADO POR HORA]");
17        System.out.println("-----");
18        System.out.println("Nombre: "+super.getNombre());
19        System.out.println("Apellido: "+super.getApellido());
20        System.out.println("Area: "+super.getArea());
21        System.out.println("Sueldo: "+this.calcularSueldo());
22    }
23
24    @Override // Sobreescribe Metodo
25    public double calcularSueldo() {
26        this.totalPagar= this.horasTrabajadas*this.valorHora;
27        return this.totalPagar;
28    }
29 }
30

```

## Subclase EmpleadoPorHora

```

1 package ejercicioherencia;
2
3 public class EmpleadoAsalariado extends Empleado {
4     private double sueldoBasico;
5     private double bonificacion;
6     private float porcRetencion;
7     private double totalPagar;
8
9     public EmpleadoAsalariado(double sueldoBasico, double bonificacion, float porcRetencion,
10                                String nombre, String apellido, String area) {
11
12         super(nombre, apellido, area);
13         this.sueldoBasico = sueldoBasico;
14         this.bonificacion = bonificacion;
15         this.porcRetencion = porcRetencion;
16     }
17
18     @Override // Sobreescribe Metodo
19     public void mostrarInfo() {
20         System.out.println("[ INFORMACION DE EMPLEADO ASALARIADO ]");
21         System.out.println("-----");
22         System.out.println("Nombre: "+super.getNombre());
23         System.out.println("Apellido: "+super.getApellido());
24         System.out.println("Area: "+super.getArea());
25         System.out.println("Sueldo: "+this.calcularSueldo());
26     }
27
28     @Override // Sobreescribe Metodo
29     public double calcularSueldo() {
30         this.totalPagar= (this.sueldoBasico+this.bonificacion);
31         this.totalPagar-= this.totalPagar*this.porcRetencion/100;
32         return this.totalPagar;
33     }
34 }

```

## Subclase EmpleadoAsalariado

```

5  package ejercicioherencia;
6
7  /**
8   *
9   * @author JAIRO F
10  */
11  public class EjercicioHerencia {
12
13      /**
14       * @param args the command line arguments
15       */
16      public static void main(String[] args) {
17          // TODO code application logic here
18          Empleado e1, e2, e3;
19
20          e1 = new Empleado("JAIRO FRANCISCO", "SEOANES LEON", "ADMINISTRATIVA");
21          e2 = new EmpleadoPorHora("LUIS", "SARATE", "OPERATIVA", 240, 15000);
22          e3 = new EmpleadoAsalariado("SERGIO", "SANCHEZ", "SERVICIO GENERAL", 580000, 60000, 16);
23
24          e1.mostrarInfo();
25          System.out.println("");
26          e2.mostrarInfo();
27          System.out.println("");
28          e3.mostrarInfo();
29
30      }
31  }

```

Output - ejercicioHerencia (run)

```

[ INFORMACION DE EMPLEADO ]
-----
Nombre: JAIRO FRANCISCO
Apellido: SEOANES LEON
Area: ADMINISTRATIVA
Sueldo: 0.0

[ INFORMACION DE EMPLEADO POR HORA]
-----
Nombre: LUIS
Apellido: SARATE
Area: OPERATIVA
Sueldo: 3600000.0

[ INFORMACION DE EMPLEADO ASALARIADO ]
-----
Nombre: SERGIO
Apellido: SANCHEZ
Area: SERVICIO GENERAL
Sueldo: 537600.0

```



# EJERCICIO

Una editorial de libros y discos desea crear fichas que almacenen el título y el precio (de tipo float) de cada publicación, Crear la correspondiente clase (denominada Publicación) que implemente los datos anteriores.

A partir de esta clase, diseñar dos clases derivadas: Libro, con el número de páginas (tipo int), año de publicación (tipo int) y precio (tipo float); y disco, con duración en minutos (tipo float) y precio (tipo int) Cada una de las tres clases tendrá una función mostrar ( ), para visualizar sus datos.

Escribir un programa que cree instancias de las clases Libro y disco, solicite datos del usuario y a continuación los visualice.

Así mismo se desea crear una nueva clase base llamada ventas que contenga un arrayList con las publicaciones que sean vendidas. Esta clase un método adicionar Ventas el cual agregara al arrayList una nueva publicación que sea vendida , sea Libro o disco. Así mismo debe tener un método mostrar ( ) que obtenga y visualice todas las publicaciones vendidas.