



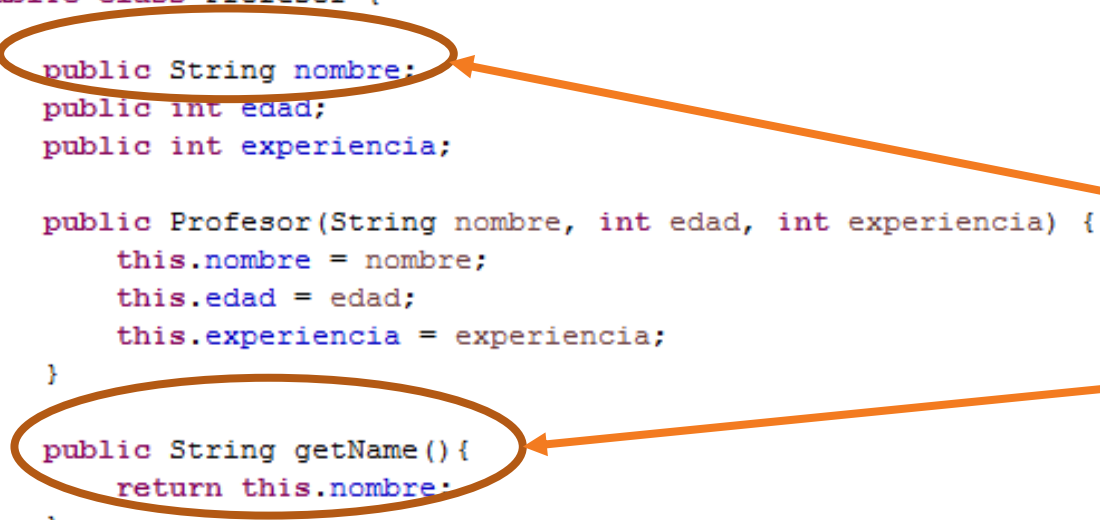
Java™

CONCEPTOS BÁSICOS

Modificadores de acceso – “Public”

- Un objeto, atributo o método **Público** puede ser accedido por cualquier otro objeto o método fuera de su clase. Su acceso no se limita a su scope.

```
1 public class Profesor {  
2  
3     public String nombre;  
4     public int edad;  
5     public int experiencia;  
6  
7     public Profesor(String nombre, int edad, int experiencia) {  
8         this.nombre = nombre;  
9         this.edad = edad;  
10        this.experiencia = experiencia;  
11    }  
12  
13    public String getName() {  
14        return this.nombre;  
15    }  
16 }
```



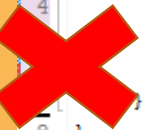
The diagram illustrates the public access of the `Profesor` class. Two orange ovals highlight the `public String nombre;` attribute (line 3) and the `public String getName()` method (line 13). Two orange arrows originate from these ovals and point to the corresponding lines in the `Main` class on the right: one arrow points from line 3 to line 4, and another from line 13 to line 6.

```
1 public class Main {  
2     @SuppressWarnings("unused")  
3     public static void main(String[] args) {  
4         Profesor luis = new Profesor("Luis Orlando Ventre", 30, 20);  
5         //Quiero acceder al nombre completo de Luis  
6         String nombre1 = luis.nombre;  
7         String nombre2 = luis.getName();  
8     }  
9 }
```


Modificadores de acceso – “Private”

- Un objeto, atributo o método **Privado** NO puede ser accedido por cualquier otro objeto o método fuera de su clase. Su acceso se limita a su scope.

```
1 public class Profesor {
2
3     public String nombre;
4     public int edad;
5     public int experiencia;
6     private int salario;
7
8     public Profesor(String nombre, int edad, int experiencia, int salario) {
9         this.nombre = nombre;
10        this.edad = edad;
11        this.experiencia = experiencia;
12        this.salario = salario;
13    }
14
15    public String getName(){
16        return this.nombre;
17    }
18
19    public int getSalario() {
20        return salario;
21    }
22
23    public void setSalario(int salario) {
24        this.salario = salario;
25    }
26 }
```



```
1 public class Main {
2     @SuppressWarnings("unused")
3     public static void main(String[] args) {
4         Profesor luis = new Profesor("Luis Orlando Ventre", 30, 20, 10000);
5         //quiero saber el salario de Luis
6         int salario = luis.salario;
7     }
8 }
```



```
1 public class Main {
2     @SuppressWarnings("unused")
3     public static void main(String[] args) {
4         Profesor luis = new Profesor("Luis Orlando Ventre", 30, 20, 10000);
5         //quiero saber el salario de Luis
6         int salario = luis.getSalario();
7     }
8 }
```

Modificadores de acceso – “Protected”

- Un objeto, atributo o método **Protegido** NO puede ser accedido por cualquier otro objeto o método fuera de su clase, EXCEPTO por sus subclases.

```
1 public class Profesor extends Empleado {  
2  
3     public String nombre;  
4     public int edad;  
5     public int experiencia;  
6  
7     public Profesor(String nombre, int edad, int experiencia, int salario) {  
8         this.nombre = nombre;  
9         this.edad = edad;  
10        this.experiencia = experiencia;  
11        this.salario = salario;  
12        this.proesion = "profesor";  
13    }  
14 }
```

```
1 public class Empleado {  
2  
3     public String profesion;  
4     protected int salario;  
5  
6     public int getSalario() {  
7         return salario;  
8     }  
9  
10    public void setSalario(int salario) {  
11        this.salario = salario;  
12    }  
13 }
```

Modificadores – “Static”

- Un modificador **estático**, hace que un atributo o método, pertenezca a la CLASE y no a la INSTANCIA de la clase.
- Es posible su uso SIN tener una instancia de la clase.

```
1 public class Main {  
2     @SuppressWarnings("unused")  
3     public static void main(String[] args) {  
4         int a = 10, b = 21;  
5         Calculadora calculadora = new Calculadora();  
6         int suma = calculadora.sumaEntera(a, b);  
7     }  
8 }
```

```
1 public class Calculadora {  
2  
3     public Calculadora(){}  
4  
5     public int sumaEntera(int a, int b) {  
6         return a+b;  
7     }  
8 }
```

```
1 public class Main {  
2     @SuppressWarnings("unused")  
3     public static void main(String[] args) {  
4         int a = 10, b = 21;  
5         int suma = Calculadora.sumaEntera(a, b);  
6     }  
7 }  
8
```


```
1 public class Calculadora {  
2  
3     public static int sumaEntera(int a, int b) {  
4         return a+b;  
5     }  
6 }
```


Modificadores – “Final”

- Un modificador **Final**, es inicializado con un valor que NO puede cambiar en tiempo de ejecución.

```
1 public class Main {  
2     @SuppressWarnings("unused")  
3     public static void main(String[] args) {  
4         int diametro = 20;  
5         double perimetro = Calculadora.pi * diametro;  
6     }  
7 }
```

```
1 public class Main {  
2     @SuppressWarnings("unused")  
3     public static void main(String[] args) {  
4         Calculadora calculadora = new Calculadora();  
5         Calculadora.numero = 11;  
6         calculadora.pi = 10;  
7     }  
8 }
```



```
1 public class Calculadora {  
2  
3     public static final double pi = 3.1415;  
4     public static final double e = 2.7182;  
5     public static int numero = 10;  
6  
7     public Calculadora() {}  
8 }
```


Clases y Métodos Abstractos

- Una clase abstracta tiene su razón de ser en la herencia. Una superclase abstracta NO tendrá instancias, sino que sus subclases tendrán instancias, y aplicarán todos los métodos abstractos de la superclase.

Puede
implementar
métodos


```
1 public abstract class Empleado {
2
3     public String nombre;
4     public int edad;
5     public String puesto;
6     protected int salario;
7
8     protected abstract String getPuesto();
9
10    public int getSalario() {
11        return salario;
12    }
13
14    public void setSalario(int salario) {
15        this.salario = salario;
16    }
17 }
```

```
1 public class Profesor extends Empleado {
2
3     public int experiencia;
4     public String materia;
5
6    public Profesor(String materia, String nombre, int edad, int salario) {
7        this.materia = materia;
8        this.nombre = nombre;
9        this.edad = edad;
10       this.experiencia = experiencia;
11       this.salario = salario;
12       this.puesto = "profesor";
13    }
14
15    @Override
16    protected String getPuesto() {
17        return "profesor";
18    }
19 }
```

```
1 public class Director extends Empleado {
2
3    public Director(String nombre, int edad, String puesto, int salario) {
4        this.nombre = nombre;
5        this.edad = edad;
6        this.puesto = "director";
7        this.salario = salario;
8    }
9
10   @Override
11   protected String getPuesto() {
12       return "director";
13   }
14 }
```

Interfaces

- Una **Interface** NO posee implementaciones de métodos. Solo posee las DEFINICIONES que obligatoriamente tienen que implementar las clases que implementen esa interfaz.
- Son utilizadas para unificar clases
- Es una colección de métodos abstractos y propiedades constantes



```
1 public interface Auto {  
2  
3     public void arrancar();  
4     public void acelerar();  
5     public void frenar();  
6     public String getMarca();  
7     public int getModelo();  
8 }
```

```
1 public class Corsa implements Auto {  
2  
3     @Override  
4     public void arrancar() {  
5     }  
6  
7     @Override  
8     public void acelerar() {  
9     }  
10  
11    @Override  
12    public void frenar() {  
13    }  
14  
15    @Override  
16    public String getMarca() {  
17        return "Chevrolet";  
18    }  
19  
20    @Override  
21    public int getModelo() {  
22        return 2017;  
23    }  
24 }
```

```
1 public class Focus implements Auto {  
2  
3     @Override  
4     public void arrancar() {  
5     }  
6  
7     @Override  
8     public void acelerar() {  
9     }  
10  
11    @Override  
12    public void frenar() {  
13    }  
14  
15    @Override  
16    public String getMarca() {  
17        return "Ford";  
18    }  
19  
20    @Override  
21    public int getModelo() {  
22        return 2017;  
23    }  
24 }
```

PREGUNTAS

