

# Métodos abstractos

**DigitalHouse** >  
Coding School



**Certified Tech  
Developer**  
The Ultimate Degree

# Índice

1. [Clases abstractas en Java](#)
2. [Métodos abstractos en Java](#)
3. [Sobreescribir métodos abstractos](#)
4. [Atributos y métodos en clases abstractas](#)

**1**

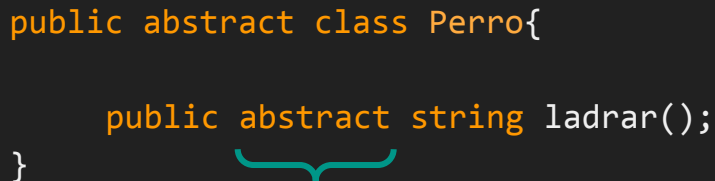
# **Clases abstractas en Java**

# Clases abstractas en Java

Definimos a las clases abstractas y el comportamiento en abstracto con la palabra clave "abstract". Cómo el comportamiento es abstracto (solo decimos qué hacer), los métodos abstractos no tienen código asociado, no tienen "cuerpo". Veamos un ejemplo:

## abstract class

Especifica que la clase Perro es abstracta.



```
public abstract class Perro{  
    public abstract string ladrar();  
}
```

Especifica que el método ladrar() es abstracto.

# Métodos abstractos en Java

# Métodos abstractos en Java

Si **Doberman** quiere **SER UN Perro**, entonces debe respetar el contrato de los Perros: debe implementar un método que se llame **ladrar**, que devuelva un **String** y que no reciba parámetros. En pocas palabras, **debe sobrescribir todos los métodos abstractos definidos en Perro**.

```
public class Doberman extends Perro{  
  
    public string ladrar() {  
        return "ladro como un dóberman GUAU!!!";  
    }  
  
}
```

# Métodos abstractos en Java

Es decir, si Perro dice qué todos los perros deben ladrar(), la hija Doberman, debe “explicar” cómo hacerlo. Llamaremos a esta operación “implementar” el método ladrar().

```
public class Doberman extends Perro{  
  
}
```



En este caso, la clase Doberman arrojará un error de compilación, porque no respeta el contrato de la clase Perro.

**3**

**Sobreescribir  
métodos abstractos**



# Sobreescribir métodos abstractos

Si Caniche quiere SER UN Perro, también debe sobrescribir todos los métodos abstractos definidos en Perro.

```
public class Caniche extends Perro{  
  
    public string ladrar() {  
        return "ladro como un caniche guau...";  
    }  
}
```



Cuando implementamos los métodos, estos dejan de ser abstractos, por eso, en Doberman y Caniche ya no usamos la palabra clave abstract.



Las reglas para la implementación de los métodos abstractos son las de la sobreescritura (de hecho, es lo que estamos haciendo, sobrescribiendo comportamiento abstracto) así que aplican las mismas reglas: respetar tipo, cantidad y orden de los parámetros



Si no lo hacemos, entonces no respetamos el contrato, si no respetamos el contrato, la clase arrojará un error de compilación.

**4**

## **Atributos y métodos en clases abstractas**

# Atributos y métodos en clases abstractas

Una clase abstracta es una clase como cualquier otra y, por tanto, puede tener atributos y puede tener métodos concretos. Aun así, tengamos en cuenta que solo los abstractos serán los que definan el contrato.

```
public abstract class Perro{  
    private String nombre;  
  
    public void setNombre(String nombre){  
        this.nombre = nombre;  
    }  
  
    public String getNombre(){  
        return nombre;  
    }  
  
    public abstract String ladrar();  
}
```

# Atributos y métodos en clases abstractas

¿Por qué tener métodos concretos en una clase que no se puede instanciar? Porque estos métodos son susceptibles de ser reutilizados.

Por otro lado, que una clase abstracta no se pueda instanciar, no significa que no pueda tener constructores, el objetivo es el mismo: podemos definir constructores para reutilizar código cuando heredamos de esa clase abstracta.



# Ejemplo

```
public class Prueba{  
  
    public static void main(String[] args){  
  
        Doberman perro1 = new Doberman();  
        perro1.ladrar();  
  
        Caniche perro2 = new Caniche();  
        perro2.ladrar();  
  
    }  
}
```

# Ejemplo

```
public class Prueba{  
  
    public static void main(String[] args){  
  
        Perro perro1 = new Perro();  
        perro1.ladRAR();  
  
    }  
}
```



Esto dará error al compilar. Recordá que por definición una clase abstracta **NO** puede ser instanciada.

DigitalHouse>  
Coding School