## **S06.s1 - Clases abstractas. Creación y ampliación de clases abstractas**

### **1. ¿Qué es una clase abstracta en Java y cuál es su propósito principal?**

* Una **clase abstracta** es una clase que **no puede instanciarse directamente** (no se pueden crear objetos de ella).
* Se utiliza como una **plantilla o modelo** para que otras clases la hereden.
* Puede contener:  
  + **Métodos abstractos** (declarados, pero sin implementación).
  + **Métodos regulares** (con implementación).
  + Atributos y constructores.
* **Propósito principal**: Definir una estructura común para un conjunto de clases relacionadas, obligando a las subclases a implementar ciertos comportamientos.

### **2. ¿Cómo se crea una subclase que extiende una clase abstracta en Java?**

* Se utiliza la palabra clave **extends**.
* La subclase **debe implementar todos los métodos abstractos** de la clase padre, salvo que también sea abstracta.

### **3. ¿Qué es un método abstracto y cómo se diferencia de un método regular en una clase abstracta?**

* Un **método abstracto**:  
  + Se declara con la palabra clave **abstract**.
  + **No tiene cuerpo** (solo la firma).
  + Obliga a que las subclases lo implementen.

Una **clase abstracta** en Java es una clase que **no se puede instanciar directamente**, es decir, **no puedes crear objetos a partir de ella**.  
 Su propósito es servir como **plantilla o modelo** para que otras clases la extiendan (**hereden**) y completen su comportamiento.

## **Características principales:**

Se declara con la palabra clave abstract.  
  
 abstract class Figura {

abstract double area(); // método abstracto

}

1. Puede contener:  
   * **Métodos abstractos** (sin cuerpo, solo la firma).
   * **Métodos normales (concretos)** ya implementados.
   * Atributos y constructores.
2. Las **subclases concretas** que la heredan están obligadas a implementar sus métodos abstractos.
3. Se usa para representar conceptos generales que luego se especializan.

## **¿Cuál es su propósito?**

* Evitar código duplicado, centralizando lo común en la clase abstracta.
* Forzar a que las subclases implementen ciertos métodos (garantizando que todas cumplan con un contrato mínimo).
* Facilitar la **herencia y el polimorfismo**.