1. Identificar las clases que heredan, ¿qué atributos y métodos heredan?

 **Clases que heredan**:

* Triángulo hereda de FiguraGeometrica.
* SuperficiePlana podría ser una superclase de FiguraGeometrica si estuviera implementada como tal (dependiendo de tu diseño).

 **Atributos y métodos heredados**:

* Triángulo hereda los atributos y métodos definidos en FiguraGeometrica.
* Los atributos podrían incluir características generales de cualquier figura geométrica, como nombre o color (si los hubiera).
* Los métodos heredados podrían incluir métodos como getNombre(), getArea(), o cualquier otro método general que defina el comportamiento común a todas las figuras geométricas.

1. Identificar las clases que conforman una composición.

**Clases que conforman una composición**:

* Triángulo tiene una composición con la clase Punto, ya que un Triángulo está compuesto por tres instancias de la clase Punto.
* Esto significa que Triángulo no puede existir sin sus puntos, lo cual es un fuerte indicador de una relación de composición.

1. ¿Qué es una superclase y una subclase?

 **Superclase**: Es una clase de la cual otra clase hereda. Es más general y define atributos y métodos comunes que pueden ser utilizados por las subclases. En tu caso, FiguraGeometrica es una superclase de Triángulo.

 **Subclase**: Es una clase que hereda de otra clase (la superclase). Las subclases pueden añadir atributos y métodos específicos además de los heredados. Triángulo es una subclase de FiguraGeometrica.

1. ¿Por qué usamos *abstract*? ¿Se puede dejar de heredar un método de una clase abstracta?

 **Uso de abstract**: Usamos abstract para declarar clases o métodos que no tienen una implementación completa en la clase base y que deben ser implementados por las subclases.

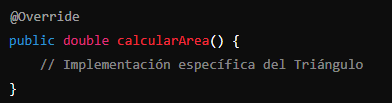
* Una clase abstracta no se puede instanciar directamente.
* Un método abstracto no tiene cuerpo en la superclase y debe ser implementado en cada subclase concreta.

 **Heredar métodos de una clase abstracta**:

* No se puede evitar heredar un método abstracto, pero es obligatorio implementarlo en la subclase, a menos que la subclase también sea abstracta.
* Si no quieres que una subclase implemente un método abstracto, deberías hacer que esa subclase también sea abstracta.

1. ¿Qué anotación utilizo para sobreescribir métodos?

La anotación que se utiliza para sobreescribir métodos en Java es @Override. Esta anotación se coloca encima del método que está siendo sobreescrito en la subclase para indicar que se está proporcionando una nueva implementación del método en la superclase.



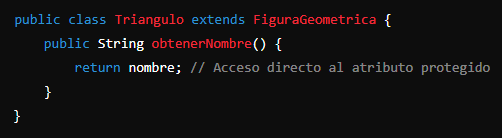
1. Los atributos de la clase Figura Geométrica conviértalas en protected. ¿En qué condición convierte a los atributos? ¿Es posible acceder a los atributos protegidos sin utilizar una invocación a super() o sin método get?.

 **Condición de atributos protected**:

* Cuando los atributos son protected, están disponibles para las subclases y para otras clases en el mismo paquete. Esto es más accesible que private, pero más restrictivo que public.

 **Acceso a atributos protegidos**:

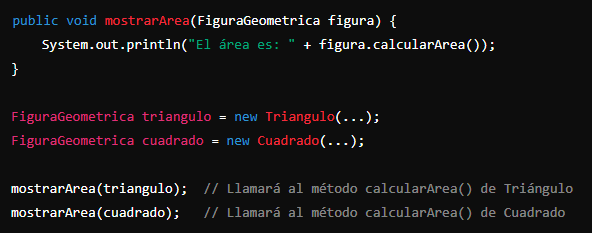
* Es posible acceder directamente a los atributos protected desde una subclase sin necesidad de invocar a super() o utilizar un método get.
* Por ejemplo, si nombre es un atributo protected en FiguraGeometrica, Triángulo puede acceder a nombre directamente:



1. ¿Cómo aplicarías polimorfismo?.

**Polimorfismo** se refiere a la capacidad de una variable de referencia de un tipo base (como FiguraGeometrica) de apuntar a objetos de subclases diferentes (Triángulo, Cuadrado, etc.) y ejecutar el método correspondiente de la subclase.

* Por ejemplo, si tienes un método que acepta una FiguraGeometrica como parámetro, puedes pasarle cualquier objeto de una subclase (Triángulo, Cuadrado, etc.):



En este caso, el método mostrarArea() es polimórfico porque puede trabajar con cualquier subclase de FiguraGeometrica, llamando al método calcularArea() correspondiente dependiendo del tipo de objeto real al que se refiere.