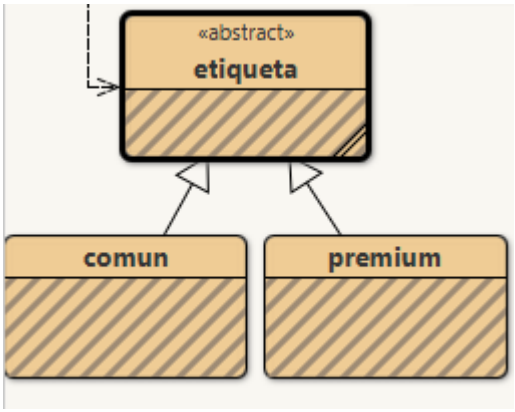
Nombre de los alumnos: Antoniow Agustín, Alegre Ariel Santiago.
Nombre del profesor: Mónica Inés Ringa
Grupo Laboratorio: Grupo 1 – lunes 15hs.
Trabajo Práctico: 5
DNI: 46.380.591, 44.467.862.
Fecha de entrega: 20/10/25

En este trabajo practico, dimos el uso de herencia, abstracción y polimorfismo, el concepto de súper clase o clase padre y de cómo sus clases hija heredan todos sus métodos y atributos, la herencia se la representa con las dos flechas finas y que tienen la punta de flecha vacía, tanto común como Premium heredan de la clase etiqueta




```
public class premium extends etiqueta
{
```

Con la palabra reservada extends, indicamos que una clase hereda de una superclase sus distintos métodos y atributos, en el constructor de la clase hija debe tener los atributos de la clase padre, no es necesario definir sus get y set porque de eso se encarga la superclase, simplemente se usa el método super(dentro los atributos que se inicializaran en el constructor de la superclase).

```
public premium(int p_colores, double p_costo){
    super(p_costo);// el cual se visualiza entre
    this.setColores(p_colores);
}
```

Dimos también los mecanismos de abstracción que se representa con la etiqueta de abstract en la firma de la clase esto indicica que es una clase de tipo abstracta(no puede instanciarse directamente) y que puede tener parte o la totalidad de sus métodos abstractos(si o si debe tener al menos un método abstracto), lo que significa que estos son redefinidos en clases de menor jerarquía(de manera obligatoria).
Un ejemplo es el método precio de la clase etiqueta el cual no está definido en su propia clase, sino que se redefine de manera diferente en sus clases hijos en base a ciertas condiciones o restricciones que tengan

```
// el método precio de tipo double tamb
public abstract double precio(int q);
```

Esta la redefinición de dicho método, en la clase Premium

```
    public double precio(int q){
    return super.getCosto() + adicional();
```

Esta estrategia permite aplicar polimorfismo, ya que el mensaje precio() puede enviarse a cualquier instancia de Etiqueta, y cada una de las clases responderá con su propia lógica.
Método redefinido en la clase Común

```
/**
 * método para obtener el precio de la prenda.
 */
@Override
public double precio(int q){
    return (super.getCosto() * q) + this.getPlus();
}
```

Ambas clases responden al mismo mensaje, pero con comportamientos distintos.

En resumen, lo que se aplicó en este TP fue encapsulamiento respetando el principio de ocultamiento de datos, accediendo a los atributos mediante sus métodos. La reutilización con la herencia podemos evitar el uso de código duplicado y centralizando los atributos y métodos en una superclase. Extensibilidad debido a que el uso de clases abstractas permite la incorporación de nuevos tipos de clases que tengan similitudes comunes con la superclase.