TP Nro 13

1. La Herencia, es la acción de crear una nueva clase partiendo de otra existente, utilizando sus atributos y métodos. La herencia de clases es muy importante para la programación orientada a objetos, ya que nos permite organizar mejor los objetos que queremos crear, y optimizar el código de programas grandes al no tener que crear una clase entera nueva cuando ya hay otra relacionada.
2. ¿Cómo afecta a esto el encapsulamiento?, el encapsulamiento define qué atributos y métodos podremos acceder directamente desde la clase hija (heredada). Una clase heredada solo puede heredar los atributos protected y public, no los private.
3. Public: Este tipo de visibilidad puede ser heredado por una clase hija, ya que los atributos y métodos public se pueden acceder desde cualquier lado.  
   Protected: Este tipo de visibilidad es exclusivo para la herencia de clases. Los métodos y atributos protected pueden ser heredados, y solo se pueden acceder desde clases heredadas.  
   Private: Este tipo de visibilidad no puede ser heredado, y solo se pueden acceder a los atributos y métodos private con métodos de la misma clase que los contiene. Esto es incluso para las clases heredadas.
4. Una especialización es cuando una clase heredada, se vuelve más específica que su clase padre, que es más general. Un ejemplo sería que si yo tengo una clase padre Humano, mi clase hija especializada podría ser Empleado.
5. La herencia múltiple, nos permite optimizar nuestro código heredando múltiples atributos y métodos de diferentes clases al mismo tiempo, ahorrándonos tiempo. El principal problema de esto, es que dificulta la lectura del código y puede provocar confusiones. Es complicado rastrear el origen de los atributos y métodos a través del código, cuando hay múltiples posibilidades y nombres repetidos.