Apuntes Prototype:

* La copia de objetos manual genera acoplamiento entre clases, porque debemos conocer la clase del objeto del que se genera la copia en sí.
* Si se trata de aplicaciones o dependencia de terceros, no tenemos acceso a la clase en si, a veces conocemos la interfaz que sigue un objeto, pero no la clase concreta del objeto.
* En este caso, ¿en métodos que reciben parámetros genéricos, como se supone que debemos duplicar el objeto?, si no tenemos idea de que clase concreta está implementando?
* En estos escenarios se debe utilizar el patron prototype.
* El prototype en un patrón de diseño creacional que delega el proceso de clonación a los objetos reales que se están clonando, sin acoplar la creación del objeto a su clase.
* EL método de clonación permite desacoplar la lógica de clonación de la clase cliente y permite acceso a los campos privados
* El método de clonación y los campos privados pertenecen a la misma clase, ahora el cliente puede hacer una copia de cualquier objeto que implemente la interfaz
* Cada objeto que admita la clonación se llama prototype