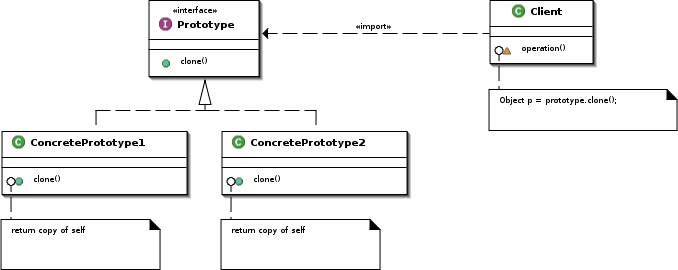
**Prototype**

**Objetivo:**

Evitar el costo inherente a la creación de un objeto nuevo mediante el operador new cuando esto demasiado costoso para la aplicación.

**Aplicación:**

Para implementar este patrón, se declara una clase base abstracta que tiene un método *clone()*. Cualquier clase que necesite un constructor deriva de la clase abstracta e implementa el método *clone()*.  
  
El cliente, en vez de escribir código que hace uso del operador *new* sobre una clase específica, llama al método *clone()* de la clase prototipo.

**Diseño:**

**Observer**

**Objetivo:**

Este patrón de diseño permite reaccionar a ciertas clases llamadas observadores sobre un evento determinado.

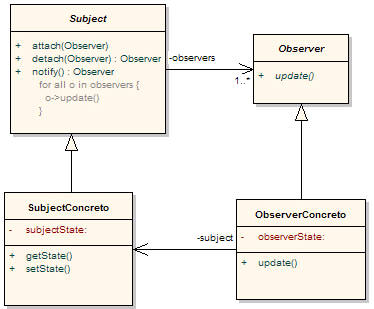
Es usado en programación para monitorear el estado de un objeto en un programa. Está relacionado con el principio de invocación implícita. La motivación principal de este patrón es su utilización como un sistema de detección de eventos en tiempo de ejecución. Es una característica muy interesante en términos del desarrollo de aplicaciones en tiempo real.

**Aplicación:**

Debe utilizarse cuando:

* Un objeto necesita notificar a otros objetos cuando cambia su estado. La idea es encapsular estos aspectos en objetos diferentes permite variarlos y reutilizarlos independientemente.
* Cuando existe una relación de dependencia de uno a muchos que puede requerir que un objeto notifique a múltiples objetos que dependen de él cuando cambia su estado.

**Diseño:**

****

**Subject**: Conoce a sus observadores y ofrece la posibilidad de añadir y eliminar observadores. Posee un método llamado attach() y otro detach() que sirven para agregar o remover observadores en tiempo de ejecución.

**Observer**: Define la interfaz que sirve para notificar a los observadores los cambios realizados en el Subject.

**SubjectConcreto**: Almacena el estado que es objeto de interés de los observadores y envía un mensaje a sus observadores cuando su estado cambia.

**ObserverConcreto**: Mantiene una referencia a un SubjectConcreto. Almacena el estado del Subject que le resulta de interés. Implementa la interfaz de actualización de Observer para mantener la consistencia entre los dos estados.