# Patrón Observador

El patrón Observador es útil para diseñar un modelo de comunicación coherente entre un conjunto de objetos dependientes y un objeto del que dependen. Esto permite que los objetos dependientes tengan su estado sincronizado con el objeto del que dependen. El conjunto de objetos dependientes se refiere a los **observadores** “*observers*” y el objeto del que dependen se denomina **sujeto** “*subject*”. Para lograr esto, el patrón Observador sugiere un modelo editor-suscriptor “*publisher-subscriber*” que conduce a un límite claro entre el conjunto de objetos Observador y el Sujeto objeto.

* Un observador típico es un objeto con interés o dependencia en el estado del sujeto.
* Un sujeto puede tener más de uno de esos observadores.

Cada uno de estos observadores necesita saber cuándo el sujeto experimenta un cambio en su estado. El sujeto no puede mantener una lista estática de tales observadores, ya que la lista de observadores para un sujeto dado podría cambiar dinámicamente. Por lo tanto, cualquier objeto con interés en el estado del sujeto debe registrarse explícitamente como un observador con el sujeto. Cada vez que el sujeto sufre un cambio en su estado, notifica a todos sus observadores registrados. Al recibir la notificación del sujeto, cada uno de los observadores consulta al sujeto para sincronizar su estado con el del sujeto. Por lo tanto, un sujeto se comporta como un editor al publicar mensajes a todos los observadores que se suscriben.

* El escenario contiene una relación de uno a muchos entre un sujeto y el conjunto de sus observadores.
* Cada vez que la instancia del sujeto sufre un cambio de estado, todos sus observadores dependientes son notificados y pueden actualizarse.
* Cada uno de los objetos del observador tiene que registrarse con el sujeto para recibir una notificación cuando haya un cambio en el estado del sujeto.
* Un observador puede registrarse o suscribirse con múltiples sujetos.
* Cuando un observador desea no volver a ser notificado, se da de baja con el sujeto.

## Funcionamiento:

* El tema debe proporcionar una interfaz para registrar y cancelar el registro de notificaciones de cambio.
* Uno de los siguientes dos debe ser cierto:
  + En el modelo de extracción *“In the pull model”*: el sujeto debe proporcionar una interfaz que permita a los observadores consultarle al sujeto la información de estado requerida para actualizar su propio estado.
  + En el modelo de inserción *“In the push model”*: el sujeto debe enviar la información del estado en la que los observadores pueden estar interesados.
* Los observadores deben proporcionar una interfaz para recibir notificaciones del sujeto.

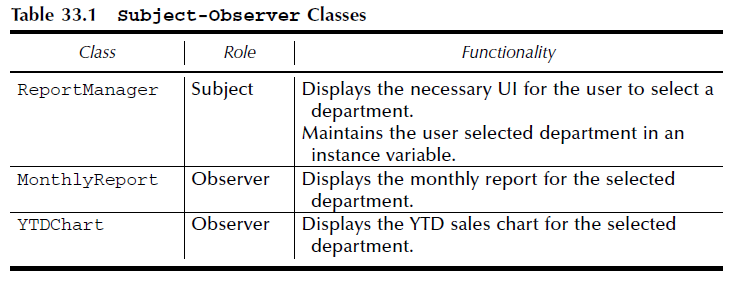
El diagrama de clases en la Figura 33.1 describe la estructura de diferentes clases y su asociación, atendiendo a la lista de requisitos anterior. De este diagrama de clase se puede ver que:

* Se espera que todos los sujetos proporcionen implementación para una interfaz similar a la interfaz Observable.
* Se espera que todos los observadores tengan una interfaz similar a la interfaz del Observador.

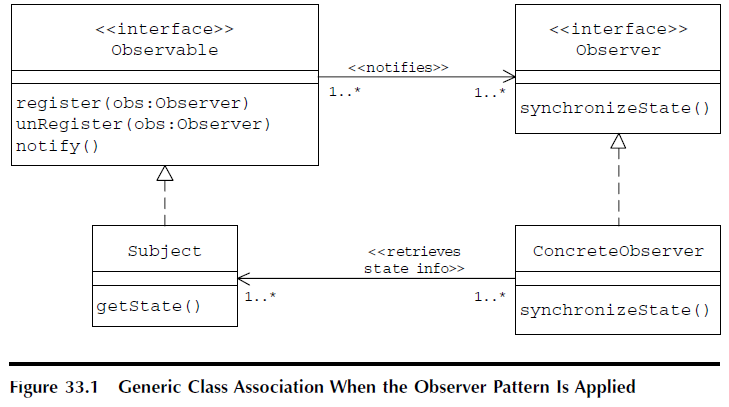
Se pueden pensar en varias variaciones al aplicar el patrón de Observador, lo que lleva a diferentes tipos de observadores de sujetos, como observadores que están interesados solo en tipos específicos de cambios en el sujeto.

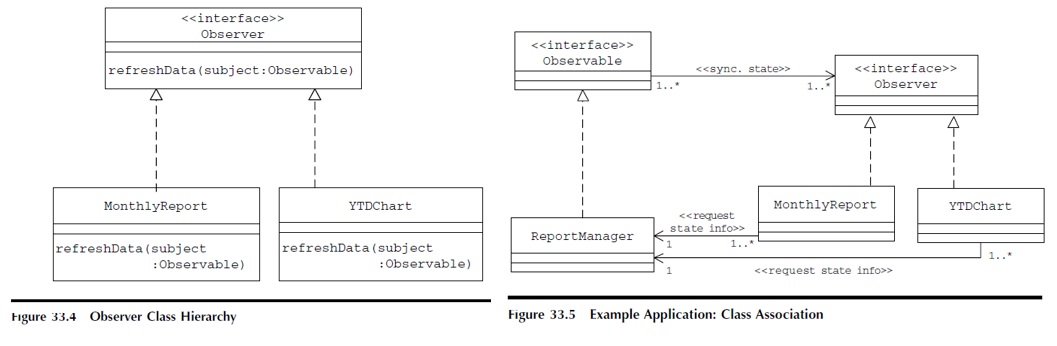
## Añadir nuevos observadores

Después de aplicar el patrón Observador, se pueden agregar dinámicamente diferentes observadores sin requerir ningún cambio en la clase **sujeto**. Del mismo modo, los observadores no se ven afectados cuando cambia la lógica de cambio de estado del sujeto.



La clase ReportManager (Listado 33.1) proporciona implementación para métodos declarados en la interfaz Observable. Los dos objetos de informe dependientes pueden usar estos métodos para especificar como observadores. ReportManagers almacena cada uno de estos observadores registrados en el vector observersList. El departamento seleccionado actualmente constituye el estado del objeto ReportManager y se mantiene en forma de una variable de instancia seleccionada departamento. Cada vez que se establece un nuevo valor para la variable de departamento (esto constituye un cambio en el estado), se invoca el método para notificar a los observadores. Como parte del método thenotifyObservers, ReportManager invoca el método de actualización de datos (Observable) en cada uno de sus observadores registrados actualmente. Además, proporcionar una implementación para los métodos de interfaz Observable, ReportManager muestra la interfaz de usuario necesaria como en la Figura 33.3 para permitir que un usuario seleccione un departamento de interés específico. También definido como interfaz Observador para el mar implementado por las clases Informe mensual y YTDChart (Figura 33.4 y Listado 33.2):





## Asociación sujeto-observador

Por lo general, un cliente primero crearía una instancia del asunto (ReportManager). Cada vez que se crea un objeto Observer (por ejemplo, MonthlyReport, YTDChart), le pasa la referencia de instancia de Sujeto como parte de su llamada al método Constructor. El objeto Observador se registra con este ejemplar del Sujeto.

