Tomislav S. Mitic
DePaul University Winter 2013
SE 430
Final Exam

The Observer pattern allows one or more objects, known as the observers, to watch another object known as the subject. The pattern allows the subject and observers to form a publish-subscribe relationship. Observers register with the subject to receive updates on events the subject deams noteworthy. When the subject needs to inform its observers of an event, it sends the event to each of the registered observers.

The benefit for using the Observer pattern is that it decouples the observer from the subject. The subject doesn't need to know anything special about its observers. And the only thing the subject allows is for observers to subscribe. When a subject detects an event it simply passes it on to each of its observers.

The Observer pattern is a behavioral design pattern. It provides a way for classes to be loosely coupled and for one class, or many, to be notified when another class is updated. When something has happened in one place, anyone who is observing and is interested in that event is notified of it.

The Observer pattern supports abstraction since it focuses on the essential aspects that subjects and observers possess while ignoring and/or concealing less important or non-essential aspects of those objects. The subject provides the capability to register observers to receive notifications, to remove observers from receiving notifications, and to notify the observers that some event of interest has happened. While observers only have the capability of being updated that some event has happened. There is no other known functionality between the subject and its observers.

The Observer pattern support separation since interfaces are used to specify the behavior of the subject and of the observers. The interface is viewed as the visible, external aspect of the software that must be understood in order to use the software. The implementation of the respective behaviors of the subject and observers is left as the responsibility of the implementor of those objects. The only thing that matters is that an implementation satisfies an interface by implementing all of the behaviors within that interface.

The Observer pattern supports encapsulation by the fact that the subject knows how to call the update method of the observers that it is keeping track of, no other details or functionality of the observer is exposed to it. The subject doesn't even know how the update method is implemented in the observers. Essentially the observers are like a black box to the subject. In the same way the subject is a black box to the observers, the observers only know how to register and unregister for updates from the subject. The observers don't even know how they are registered/unregistered within subject, nor do they have any access to any other details about

the subject. Encapsulation is a language facility, it bundles the behavior that are of most importance to subject and observers together within interfaces which the subject and observers then use to reference each other.

The Observer pattern supports information hiding by the fact that on the subject only the register and unregister events are exposed to the observers all other details about that object have been hidden to the observers. Observers are only concerned with how to register/unregister from the subject. Likewise the subject only knows how to update the observers of an important event that has happened, no other details about the observers is known to the subject. Should the implementation details of the subject change it will not matter to the observers since they will still know that they are able to register/unregister from the subject; should the implementation details of the observers change it will not matter to the subject since it will still know how to update the observers of important events that are happening. This also implies that the observer and subjects are not allowed to pass references of key information inside of themselves between each other. The subject is allowed to pass primitive data types to the observers or a DTO of bundled information, while the observers are able to register/unregister themselves from the subject but are unable to pass any key piece of data that is a reference within themselves to the subject. Information hiding is a design facility, it ensures that the implementation details of the subject and observers are hidden and only the necessary methods between them are exposed so that they can communicate with each other.