Introduction To Design Patterns

You will learn about design techniques that have been successfully applied to different scenarios.

What Is A Design Pattern?

- A general and reusable solution to a commonly occurring problem in the design of software.
- IT IS NOT a finished algorithm that can be directly translated into program code.
- IT IS a template for how to solve a problem that has been used in many different situations.
- Object-Oriented design patterns show interactions between classes and objects without the specific the program code that implements the pattern.

Origin Of Design Patterns

- The foundation for design patterns come from the original patterns specified in the book "*Design Patterns: Elements of Reusable Object-Oriented Software*"
- Authors: "The gang of four" (Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides).
- Although examples of the patterns were provided in C++ and SmallTalk the patterns can be applied to any Object-Oriented language.

Monitor (Synchronization) Pattern

- In terms of concurrent programming a monitor is an object that can be used by more than one thread.
- Synchronization occurs by allowing only one thread to access the monitor (and therefore the methods of the object) at a time.
- The mutually exclusive nature of access to the monitor greatly simplifies the logic required.

The Model-View-Controller Pattern[1]

- Sometimes the same data may have to be accessed under different contexts e.g., powerful desktop, web, mobile device.
- Each context may require a different interface (e.g., web page on a mobile device, software on a computer).
- Even within an interface there may be a desire to see different views of the data e.g., financial analysts may want to see details (spreadsheet and/or financial statement) whereas the shareholders or management may focus on overview views (graphs)

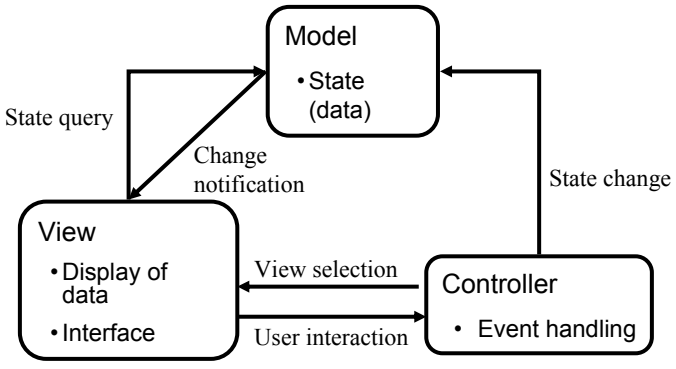
1 Some additional sources that describe the model-view controller pattern:

I. Sun Microsystems: http://java.sun.com/blueprints/patterns/MVC-detailed.html

II. Microsoft: http://msdn.microsoft.com/en-us/library/ms978748.aspx

The Model-View-Controller Pattern[1]

- With this pattern the logic required to maintain the data is separated (database, text file) from how the data is viewed (graph, numerical) vs. how the data can be interacted with (GUI, command line).



Model-View Controller Pattern (2)

- With many client applications the view and the controller may be viewed as one entity.
- With web-based applications the view and the controller may be very well defined:
  - –View: client browser program
  - –Controller: the server side applications that handle the web requests

Model-View-Controller Pattern (3)

- Implementing different parts that are decoupled (minimized dependencies) provides many benefits:
  - –One part may be changed independent of the other parts e.g., updates to the interface can have minimal impact on the data.
  - –It's seldom that one person will have a deep understanding of all parts (e.g., knowledge of Accounting to create the financial statements vs. knowledge of web design to create the web interface). Different people with different areas of expertise can work on the different parts.
  - –One version of the data can be created and maintained and as needed different ways of interacting and viewing data can be developed.

You Should Now Know

- What is a design pattern
- How two example design patterns work