Behavioral pattern

From Wikipedia, the free encyclopedia

In software engineering, **behavioral design patterns** are design patterns that identify common communication patterns between objects and realize these patterns. By doing so, these patterns increase flexibility in carrying out this communication.

Examples of this type of design pattern include:

- Chain of responsibility pattern: Command objects are handled or passed on to other objects by logic-containing processing objects
- Command pattern: Command objects encapsulate an action and its parameters
- "Externalize the Stack": Turn a recursive function into an iterative one that uses a stack[1]
- Interpreter pattern: Implement a specialized computer language to rapidly solve a specific set of problems
- Iterator pattern: Iterators are used to access the elements of an aggregate object sequentially without exposing its underlying representation
- Mediator pattern: Provides a unified interface to a set of interfaces in a subsystem
- Memento pattern: Provides the ability to restore an object to its previous state (rollback)
- Null Object pattern: Designed to act as a default value of an object
- Observer pattern: aka Publish/Subscribe or Event Listener. Objects register to observe an event that may be raised by another object
  - Weak reference pattern: De-couple an observer from an observable[2]
- Protocol stack: Communications are handled by multiple layers, which form an encapsulation hierarchy[3]
- Scheduled-task pattern: A task is scheduled to be performed at a particular interval or clock time (used in real-time computing)
- Single-serving visitor pattern: Optimise the implementation of a visitor that is allocated, used only once, and then deleted
- Specification pattern: Recombinable business logic in a boolean fashion
- State pattern: A clean way for an object to partially change its type at runtime
- Strategy pattern: Algorithms can be selected on the fly
- Template method pattern: Describes the program skeleton of a program
- Visitor pattern: A way to separate an algorithm from an object

See also

- Concurrency pattern
- Creational pattern
- Structural pattern

References

1. "Externalize The Stack". c2.com. 2010-01-19. Archived from the original on 2010-01-19. Retrieved 2012-05-21.
2. Nakashian, Ashod (2004-04-11). "Weak Reference Pattern". c2.com. Archived from the original on 2004-04-11. Retrieved 2012-05-21.
3. "Protocol Stack". c2.com. 2006-09-05. Archived from the original on 2006-09-05. Retrieved 2012-05-21.