Japanese Design Patterns with Groovy

Using design patterns with Java is a well-established topic. Design patterns also apply to Groovy:

- some patterns carry over directly (and can make use of normal Groovy syntax improvements for greater readability)
- some patterns are no longer required because they are built right into the language or because Groovy supports a better way of achieving the intent of the pattern
- some patterns that have to be expressed at the design level in other languages can be implemented directly in Groovy (due to the way Groovy can blur the distinction between design and implementation)

Patterns

- Japanese Singleton Pattern

References

1. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley. ISBN 0-201-63361-2.
    - *The canonical reference of design patterns.*
2. Martin Fowler (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley. ISBN 0-201-48567-2.
3. Joshua Kerievsky (2004). *Refactoring To Patterns*. Addison-Wesley. ISBN 0-321-21335-1.
4. Eric Freeman, Elisabeth Freeman, Kathy Sierra, Bert Bates (2004). *Head First Design Patterns*. O'Reilly. ISBN 0-596-00712-4.
    - *A great book to read, informative as well as amusing.*
5. Dierk Koenig with Andrew Glover, Paul King, Guillaume Laforge and Jon Skeet (2007). *Groovy in Action*. Manning. ISBN 1-932394-84-2.
    - *Discusses Visitor, Builder and other Patterns.*
6. Brad Appleton (1999). Pizza Inversion - a Pattern for Efficient Resource Consumption.
    - *One of the most frequently used patterns by many software engineers!*
7. *Design Patterns in Dynamic Languages* by Neil Ford. Houston Java User's Group. Examples in Groovy and Ruby. http://www.hjug.org/present/Neal_Ford-Design_Patterns_in_Dynamic_Languages-slides.pdf

See also: Refactoring with Groovy