Two More GRASP Patterns: Indirection and Protected Variations

CMPT 371
Fall 2004
J.W. Benham
Indirection

- **Problem:** Where can one assign responsibility to avoid direct coupling between two or more components?

- **Solution:** Assign the responsibility to an intermediate object to mediate between other components or services so that they are not directly coupled.
Indirection Example: Factories

- **Problem:** Who should be responsible for creating the precedence table?
  - By Information Expert, one should assign this responsibility to the class SSSS
  - By Creator, one should assign to either SSSS or TranslatorFrontEnd
  - Either assignment significantly reduces cohesion (and increases coupling)

- **Solution:** Create a new class FrontEndTablesFactory to serve as intermediary.
Indirection Example 2: Persistent Storage

• **Problem:** Who should be responsible for storing TranslationEnvironment objects?
  – By Information Expert, we could assign to TranslationEnvironment.
  – This reduces cohesion, because the TranslationEnvironment now needs to know about the structure of persistent storage

• **Solution:** Introduce a PersistentStorage class to handle this responsibility.
Adapter Pattern (GoF)

• **Problem:** How can we convert the interface of a class to the interface clients expect?
  – Handling legacy code
  – Handling interface with differing external systems

• **Solution:** Introduce an Adapter class that provides the expected client interface and adapts it for server’s interface.
Adapter Class Diagram (Generic)
Adapter Example

• **Problem**: We want SST to work with different database software
  – MySQL, Oracle, Access, SyBase, etc.

• **Solution**: Introduce a DatabaseAdapter
  – Abstract class (or interface) provides expected interface to clients within system
  – Concrete subclass for each database system converts requests from clients to appropriate form for database system and converts results (if any) from database system to conform to client interface.
Database Adapters
Protected Variations

• **Problem:** How can one design objects, subsystems, and systems so that variations or instability in those elements does not have an undesirable impact on other elements?

• **Solution:** Identify points of predicted variation or instability and assign responsibilities to create a stable interface around them.
Protected Variations Example

**Problem:** Required search-string form for target search engines will change over time. New search engines will be introduced.

**Solution:**
- Create an abstract class `TranslatorBackEnd` with abstract method `generateTargetString(st: SyntaxTree)`
- For each supported search engine, create a concrete subclass of `TranslatorBackEnd` that generates a search string for the that search engine.
Back Ends Class Diagram

TranslatorBackEnd

GoogleBackEnd

TeomaBackEnd

generateTargetString()
Mechanisms Motivated by PV

• Core mechanisms
  – Data encapsulation
  – Interfaces
  – Polymorphism
  – Indirection

• Data-driven designs
  – Example: Translation Environments

• Interpreter-driven designs
  – Example: Develop a scripting language for back ends
Liskov’s Substitution Principle

• “Whenever an instance of a class is expected, one can always substitute an instance of any of its subclasses.”

• Consequences of LSP
  – A subclass may strengthen an invariant, but cannot weaken it
  – A subclass may weaken a pre-condition for an operation override in the subclass, but cannot strengthen it
  – A subclass may strengthen a post-condition in an operation override, but cannot weaken it
LSP Example

• BankAccount.
  – invariant: balance >= 0
  – withdraw(amount)
    • pre-condition: balance >= amount
    • post-condition: balance = balance@pre – amount

• MinimumBalanceAccount:
  – invariants:
    • balance >= minimumBalance
    • minimumBalance > 0
  – withdraw(amount)
    • pre-condition: balance >= amount + minimumBalance
    • post-condition: balance = balance@pre – amount
Another LSP Example: Is a Square a Rectangle?

- **Rectangle invariants**
  - area = length * width
  - perimeter = 2*(length + width)

- **Square invariants**
  - area = length * width
  - perimeter = 2*(length + width)
  - length = width
LSP Example (continued)

• Rectangle.setLength(newLength: double)
  – Pre-conditions: -- none
  – Post-conditions:
    • length = newLength
    • Is anything missing??

• Square.setLength(newLength: double)
  – Pre-conditions: -- none
  – Post-conditions:
    • length = newLength
    • width = newLength
Law of Demeter

• “Don’t talk to strangers.”

• Send messages only to following objects (called “familiars”)
  – self (receiver)
  – A parameter of method
  – An attribute of self
  – An element of a collection that is an attribute of self
  – An object created within the method

• All other objects are “strangers” – reached only indirectly from receiver (no direct visibility)
Law of Demeter Example

• In a method of TranslatorFrontEnd, suppose we had the following call
  – lexicalAnalyzer
    .keywordTable
    .getEntry(aLexeme)
    .getTokenType()

• TranslatorFrontEnd needs knowledge that LexicalAnalyzer has attribute keywordTable, knowledge of keywordTable method, and knowledge of KeywordTableEntry method getTokenType.
  – Reduces cohesion