CSCI 253

Object Oriented Design:
Template Method Pattern
George Blankenship

Overview

Creational Patterns
- Singleton
- Abstract factory
- Factory Method
- Prototype
- Builder

Structural Patterns
- Composite
- Façade
- Proxy
- Flyweight
- Adapter
- Bridge
- Decorator

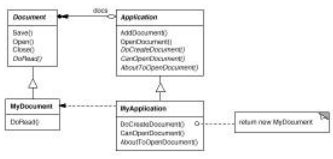
Behavioral Patterns
- Chain of Respons.
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Template Method
- Visitor

The Elements of a Design Pattern

- A pattern name
- The problem that the pattern solves
  - Including conditions for the pattern to be applicable
- The solution to the problem brought by the pattern
  - The elements (classes-objects) involved, their roles, responsibilities, relationships and collaborations
  - Not a particular concrete design or implementation
- The consequences of applying the pattern
  - Time and space trade off
  - Language and implementation issues
  - Effects on flexibility, extensibility, portability

The Template Method Pattern: The Problem

Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.



Sometimes you want to specify the order of operations that a method uses, but allow subclasses to provide their own implementations of some of these operations

Template Method Pattern George Blankenship 4

OpenDocument() Method

- **public void OpenDocument (String name) {**
  if (!CanOpenDocument(name)) { return; }
  Document doc = DoCreateDocument();
  if (doc != null) {
  docs.AddDocument(doc);
  AboutToOpenDocument(doc);
  doc.Open();
  doc.DoRead();
  }
  }
- The OpenDocument() method is a *Template Method*
- The template method fixes the order of operations, but allows Application subclasses to vary those steps as needed

Template Method Pattern George Blankenship 5

TextDocument Class

- **public abstract class TextDocument {**
  ...
  public final void printPage (Page page) {
  printTextHeader();
  printTextBody(page);
  printTextFooter();
  }
  public abstract void printTextHeader();
  public final void printTextBody(Page page) {
  System.out.println(page.body());
  }
  public abstract void printTextFooter();
  ...
  }
- The class includes a printPage() template method in a superclass that allows for specific document classes to provide unique implementations of the abstract methods to print the header and footer

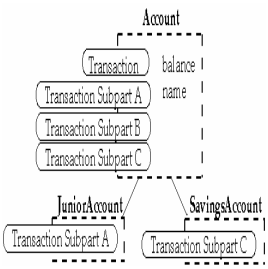
Template Method Pattern George Blankenship 6

The Template Method Pattern: Applicability

- Use the Template Method pattern:
  - To implement the invariant parts of an algorithm once and leave it up to subclasses to implement the behavior that can vary
  - To localize common behavior among subclasses and place it in a common class (in this case, a superclass) to avoid code duplication. This is a classic example of ”code refactoring.”
  - To control how subclasses extend superclass operations. You can define a template method that calls "hook" operations at specific points, thereby permitting extensions only at those points.
- *The Template Method is a fundamental technique for code reuse*.

Template Method Pattern George Blankenship 7

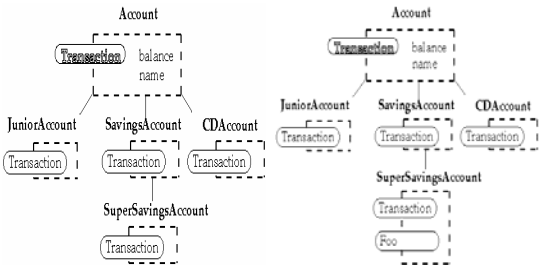
Template Method




```
class Account {
      public: void Transaction(float amount);
      void virtual TransactionSubpartA();
      void virtual TransactionSubpartB();
      void virtual TransactionSubpartC();
      }
void Account::Transaction(float amount) {
      TransactionSubpartA(); TransactionSubpartB();
      TransactionSubpartC(); // EvenMoreCode;
      }
class JuniorAccount : public Account {
      public: void virtual TransactionSubpartA();
      }
class SavingsAccount : public Account {
      public: void virtual TransactionSubpartC();
      }
Account* customer;
customer = createNewAccount();
customer-&gtTransaction(amount);
```

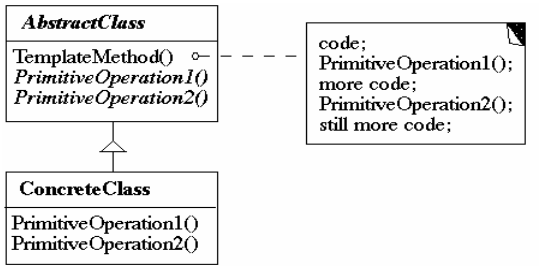
Template Method Pattern George Blankenship 8

Good Polymorphism/Bad Polymorphism



Template Method Pattern George Blankenship 9

The Template Method Pattern: Structure



Template Method Pattern George Blankenship 10

The Template Method Pattern: Participants

- Template methods tend to call:
  - concrete operations
  - concrete AbstractClass operations
  - primitive operations - must be overridden
  - factory methods
  - hook operations - can be overridden
- It is important to denote:
  - which methods must overridden
  - which methods can be overridden
  - which methods can not be overridden

Template Method Pattern George Blankenship 11

The Template Method Pattern: Collaboration

- Strategy is like Template Method except in its granularity. [Coplien, C++ Report, Mar 96, p88]
- Template Method uses inheritance to vary part of an algorithm. Strategy uses delegation to vary the entire algorithm. [GOF, p330]

Template Method Pattern George Blankenship 12

The Template Method Pattern: Consequences

- Method overloading and method overriding are good examples of template method pattern.
  - Coercion polymorphism -- refers to a single operation serving several types through implicit type conversion.
  - Overloading polymorphism -- refers to using a single identifier for different operations.
  - Parametric polymorphism -- refers to a class declaration that allows the same field names and method signatures to associate with a different type in each instance of that class.
- Note that the side effect of using coercion polymorphism is casting in and casting out if you need specific type to do the work. If you forget to do so, you may have unexpected result and it is hard to debug.
- Without using template method pattern, you may write more lines of code. The good thing is that you don't have any side effect by using specific designed method and you don't need to cast in or out.

Template Method Pattern George Blankenship 13

The Template Method Pattern: Implementation

- Operations which must be overridden by a subclass should be made abstract
- If the template method itself should not be overridden by a subclass, it should be made final
- To allow a subclass to insert code at a specific spot in the operation of the algorithm, insert “hook” operations into the template method. These hook operations may do nothing by default.
- Try to minimize the number of operations that a subclass must override, otherwise using the template method becomes tedious for the developer
- In a template method, the parent class calls the operations of a subclass and not the other way around. This is an inverted control structure that's sometimes referred to as "the Hollywood principle," as in, "Don't call us, we'll call you".

Template Method Pattern George Blankenship 14