Strategy & Template Method

4-20-2004

Opening Discussion

- Quiz grades were good again, but this time I know I was too lenient.
- Public inheritance is an is-a relationship. Composition or private inheritance give has-a relationship.
- Downcasting is bad for code maintainability, expandability, and makes runtime errors of things that could be syntax errors.
- Checked exceptions MUST be caught or passed on. Therefore they should only be used for recoverable errors.

Strategy

- With this pattern you define a family of algorithms and abstract them so that they are interchangeable.
- The design for this includes an interface for the strategy, typically named after the algorithm, and then various concrete implementations of it.
- In addition, there is some context object that keeps an instance of the strategy object and uses it. The idea is that the context is doing something that requires the strategy, but the strategy should vary independently of it.

Example

- GoF uses the example of a Composition class which is supposed to lay out some text or other collection of items. Part of the work of the composition class is decided where to put in line breaks. Depending on the exact application, different line break algorithms could be used. Each one gets it's one concrete implementation of a Compositor interface.
- You can imagine similar situations with pieces of code that depend on sorts, layouts, or selections. GeneralBlend in Java code.

Benefits and Drawbacks

- Using this pattern you implicitly create families of algorithms than can lead to greater reuse.
- Provides an alternative to subclassing the entire context.
- It eliminates the conditionals you might otherwise have when picking the algorithm to use.
- Easier to choose between implementations.
- On the down side, the client must be aware of the strategies.

More Drawbacks

- There can be overhead in context calling the strategy, especially if the interface is make more complex to handle more options.
- There are more objects around which can lead to extra overhead. The use of the Flyweight pattern can help relieve some of this.

Template Method

- This pattern defines the skeleton of an algorithm where some of the details are left for subclasses to fill in or redefine.
- Basically a template method defines an algorithm based on some abstract operations.
- When this is done dynamically, the template method will call virtual methods that are given concrete implementations in subclasses.

Example

- GoF uses the example of an application that can open and create documents that we have talked about before. In this case, the method OpenDocument is a template method. It calls abstract methods like createDocument and read that are defined by specific subclasses.
- The template method basically fixes the order of calling other methods that will be defined later.

Benefits and Drawbacks

- You have to be careful with the documentation if you provided inherited operations in the superclass that should be overridden, but don't have to be.
- In C++ you might often to this with templates instead of inheritance.
- In Java you do this with inheritance, but you have to take very seriously the suggestion to either prevent inheritance or document for it.

Progress Presentations

- Tom said that he wanted to present today. I'd like to give a brief presentation on the collision handling in the Java code. If anyone else wants to show something quick to supplement an earlier presentation feel free.
- I'm sending around sign-up sheets. I want to meet with each of you for 30 minutes. You should bring UML and printouts of your code or at least bring UML and be able to quickly tell me what files to print to see most of your work.