Архитектура программных систем

Лабораторная работа № 2

Выполнили:

Е Хэн

P33111

Преподаватель:

Задание

Из списка шаблонов проектирования GoF и GRASP выбрать 3-4 шаблона и для каждого из них придумать 2-3 сценария, для решения которых могу применены выбранные шаблоны.

Сделать предположение о возможных ограничениях, к которым можем привести использование шаблона в каждом описанном случае. Обязательно выбрать шаблоны из обоих списков.

Adapter, GOF

Introduce:

Adaptation between different objects is achieved by converting the original interface into the target interface.

shortcoming:

An adapter can only adapt to one target requirement. Extensive use of adapters may lead to more and more layers of code in the project.

Scenarios:

When you travel from the China to Russia for the first time, you may get a surprise when trying to charge your laptop. The power plug and sockets standards are different in different countries. That’s why your US plug won’t fit a German socket. The problem can be solved by using a power plug adapter that has the American-style socket and the European-style plug.

Builder, GOF

introduce:

Separate the construction process and representation of complex objects so that the same construction process can be reused.

shortcoming:

Each type of product requires a constructor, which increases the amount of code.

Scenarios:

We have a complex object that requires laborious, step-by-step initialization of many fields and nested objects. Such initialization code is usually buried inside a monstrous constructor with lots of parameters. Or they are scattered all over the client code. Like we want to create a House object. To build a simple house, we should construct four walls and a floor, install a door, fit a pair of windows, and build a roof. But it can't work if we want a bigger house, with a backyard and other goodies (like a heating system, plumbing, and electrical wiring). The Builder pattern let us extract the object construction code out of its own class and move it to separate objects called builders. The pattern divides object construction into steps (buildWalls, buildDoor, etc.). A series of these steps are performed on a builder object to create an object. The important thing to remember is that you do not have to call all of the steps. Only the steps required to produce a specific configuration of an object can be called.

Factory Method, GoF

introduce:

Compared with the simple factory pattern, if a factory is currently needed to construct specific objects, but the creation process of the factory is not yet known, we need to consider using the factory method pattern for implementation. The factory method pattern leaves the specific factory creation behavior to subclasses.

shortcoming:

The Factory Method pattern further aggregates a set of constructed objects and a set of factories that build them. When new objects need to be created, new factories also need to be defined.

Scenarios:

We're creating a logistics management application. The first version can only handle transportation by trucks and the bulk of code lives inside the Truck class. At present, most of your code is coupled to the Truck class. Adding Ships into the app would require making changes to the entire codebase. Moreover, if later we decide to add another type of transportation to the app, you will probably need to make all of these changes again.The code will be riddled with conditionals that switch the app's behavior depending on the class of transportation objects. We can replace direct object construction calls (using the new operator) with calls to a special factory method.

Pure fabrication, GRASP

introduce:

Factory Method is a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.

Weak point:

The code may become more complicated since you need to introduce a lot of new subclasses to implement the pattern. The best case scenario is when you’re introducing the pattern into an existing hierarchy of creator classes.

Scenarios:

1. If we want to implement the Data Transfer Object pattern used in the backend, we should transform "raw" data from the client into a usable model. Pure fabrication is the best solution.

Conclusion

GRASP is the premise of the GoF design pattern. The GoF design pattern is an object-oriented design pattern that meets the requirements of the GRASP pattern principle. GRASP is a route planning scheme, and GoF is a concrete and implementable route formulated according to the plan.