Lab6: Design Pattern

Tianyi Zhang
Recap of lab last week

• Simple factory (solution 1)
  → Which is not a factory pattern
  → just a programming trick, but heavily used

• Factory method pattern (solution 2)
Solution 1.

```java
Shape makeShape(int type, double width){
    switch (type) {
        case SQUARE:
            return new Square(width);
        case CIRCLE:
            return new Circle(width);
        case EQUILATERAL_TRIANGLE:
            return Triangle(width);
        default:
            System.out.println("Unknown shape type encountered: " + type);
            return null;
    }
}
```
Solution 2.

Solution 1 vs 2, difference?
Rationale: Bonus of Factory Pattern

• Solution 2 prevails Solution 1, why?

• In S1, programmers manually find code locations in the entire project and apply similar editions one by one.

• Such hardcoding involves complex data and control dependencies, more error-prone.

• S2 improves maintainability and readability
Design Patterns Tonight

- Observer Pattern
- Factory Method Pattern
- Abstract Factory Pattern
  - A little different with Factory Method Pattern
• **Definition:**
  • Observer patterns defines a one-to-many dependency between objects so that when the subject changes state, all its dependent (observer) will be notified and updated automatically.
Observer

10 mins to go through the weather monitoring application overview
Observer

<<interface>>
Subject
- registerObserver()
- removeObserver()
- notifyObservers()

WeatherData
- registerObserver()
- removeObserver()
- notifyObservers()
- getTemperature()
- getHumidity()
- getPressure()
- measurementsChanged()

<<interface>>
Observer
- update()
- display() { // display current measurements }

<<interface>>
DisplayElement
- display()

CurrentConditionsDisplay
- update()
- display() { // display current measurements }

StatisticsDisplay
- update()
- display() { // display the average, min and max measurements }

ThirdPartyDisplay
- update()
- display() { // display something else based on measurements }

ForecastDisplay
- update()
- display() { // display the forecast }
Check list (Observer)

- Check out code from svn repo
- Define methods of Subject and Observer Interface
- Add necessary parameters for each method
- Implement WeatherData and Current-ConditionDisplay class.
- Heads UP: Don't need to implement StatisticsDisplay and ForecaseDisplay class in UML diagram
Factory Method Pattern, again

- **Definition:** The Factory Method Pattern defines an interface (Creator) for creating an object (Product), but lets subclasses decide which class to instantiate.

- Factory Method lets a class defer instantiation to subclasses.

- Factory Method isolates client, any code calling factory method, from knowing what kind of concrete product is actually created.
Factory Method Pattern, again
Factory Method Pattern, again

For shape project we did last week
Factory Method Pattern, again

For Pizza Store project tonight
Check list (Factory Method)

• Check out code from svn repo

• Creator and Product abstract class already defined

• Complete missing methods according to class diagram
Abstract Factory Pattern

**Definition:** The Abstract Factory Pattern provides an interface for creating families of related or dependent objects (Products) without specifying their concrete classes.

**Difference:** abstract factory pattern concerns multi-products (product families) which are related.
Abstract Factory Pattern

The AbstractFactory defines the interface that all Concrete factories must implement, which consists of a set of methods for producing products.

This is the product family. Each concrete factory can produce an entire set of products.

The concrete factories implement the different product families. To create a product, the client uses one of these factories, so it never has to instantiate a product object.
Check list (Abstract Factory)

• Skeleton code is in abstractfactory dir in the given svn link above.

• Build PizzaIngredientFactory for each region

• Implement a set of ingredient classes to be used within each factory

• hook all this up by working our new ingredient factories into our old PizzaStore code
When you check out...

- Show me your source code for these 3 applications
- Run each app
Q&A

Thanks !