Design Pattern Decision Document

This document describes why and how I implemented each design pattern

Assignment 2 code  
Kieran Abelen

Contents

[Factory pattern 2](#_Toc23426826)

Design Pattern Implementation 1

# Beginning Class Diagram



# Factory pattern

The issue with this program is that two drawers and they cannot both be initialized at the same time since they will override each other.

c = config.read().splitlines()  
**if** c[0] == **'DrawerKieran'**:  
 **from** DrawerKieran **import** Drawer  
**elif** c[0] == **'DrawerTurtleJack'**:  
 **from** DrawerTurtleJack **import** Drawer  
config.close()

main = Interface(Parser(Drawer()))  
main.go()

To fix this we need a drawer and an abstract product for both real drawers.

The Main can initialize with the fake drawers and the interface when trying to use drawer methods will select between the two real drawers.

This aligns perfectly with the design pattern book

“Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclass. “ (Erich Gamma, 1994) pg121

Use the Factory Method pattern when

* A class can't anticipate the class of objects it must create. (exactly why I need it)
* A class wants its subclasses to specify the objects it creates.
* classes delegate responsibility to one of several helper subclasses, and you want to localize the knowledge of which helper subclass is the delegate. (Erich Gamma, 1994) pg122

Since I can’t predict what subclasses I need, a factory needs to produce the desired outcome when it is needed.

# References

Erich Gamma, J. V. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software.* United States: Addison-Wesley. Retrieved from www.uml.org.cn: http://www.uml.org.cn/c++/pdf/designpatterns.pdf