Decorator Pattern

Idaho State University | Computer Science

Isaac Griffith

CS 2263
Department of Informatics and Computer Science
Idaho State University

Outcomes

After today's lecture you will be able to:

- Understand the use of the Decorator Design Pattern
- Use and implement the Decorator Pattern
- Describe and use the Open-Closed Principle

Inspiration

“An API that isn’t comprehensible isn’t usable.” – James Gosling

Decorator Pattern

- The Decorator Pattern provides a powerful mechanism for adding new behaviors to an object at run-time.
  - The mechanism is based on the notion of “wrapping” which is just a fancy way of saying “delegation” but with the added twist that the delegator and the delegate both implement the same interface
    - You start with object A that implements abstract type X
    - You then create object B that also implements abstract type X
    - You pass A into B’s constructor and then pass B to A’s client
    - The client thinks its talking to A but its actually talking to B B’s methods augment A’s methods to provide new behavior

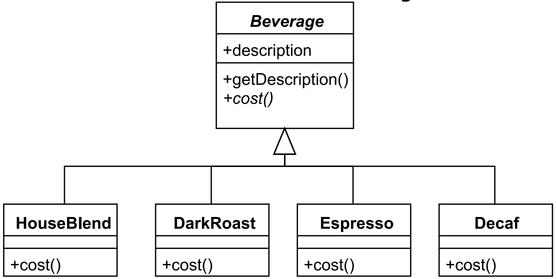
Why? Open-Closed Principle

- The decorator pattern provides yet another way in which a class's runtime behavior can be extended without requiring modification to the class
- This supports the goal of the open-closed principle:
  - Classes should be open for extension but closed to modification
    - Inheritance is one way to do this, but composition and delegation are more flexible (and Decorator takes advantage of delegation)
- The "Starbuzz Coffee" example clearly demonstrated why inheritance can get you into trouble and why delegation/composition provides greater run-time flexibility

Starbuzz Coffee

- Under pressure to update their “point of sale” system to keep up with their expanding set of beverage products
  - Started with a Beverage abstract base class and four implementations: `HouseBlend`, `DarkRoast`, `Decaf`, and `Espresso`
    - Each beverage can provide a description and compute its cost.
  - But they also offer a range of condiments including: steamed milk, soy, and mocha
    - These condiments **alter** a beverage’s description and cost.
    - “Alter” is a key word here since it provides a hint that we might be able to use the Decorator pattern

Initial Starbuzz System

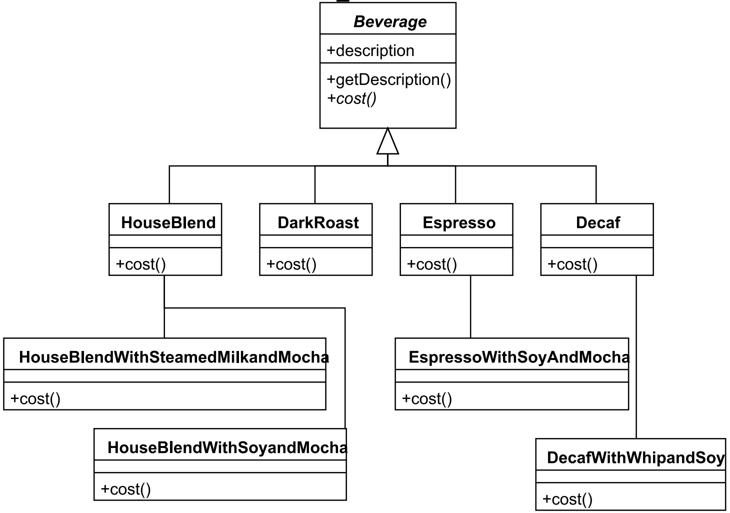


With inheritance on your brain, you may add condiments to this design in one of two ways:

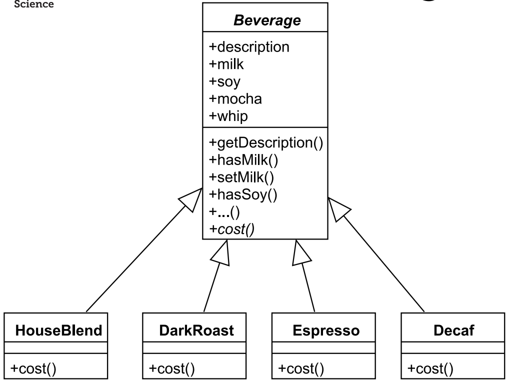
❶ One subclass per combination of condiment

❷ Add condiment handling to the Beverage superclass

One Subclass per Combination



Let Beverage Handle Condiments

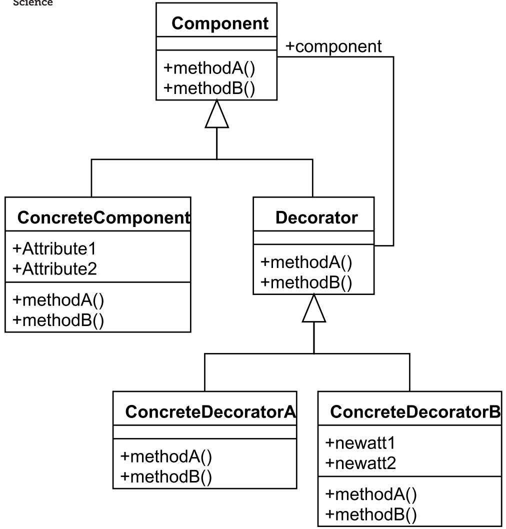


Houston, we have a problem...

1. This assumes that all concrete `Beverage` classes need these condiments
2. `Condiments` may vary (old ones go, new ones are added, price changes, etc.), shouldn't they be encapsulated some how?
3. How do you handle "double soy" drinks with Boolean variables?

Idaho State University Computer Science

Decorator Pattern: Definition and Structure

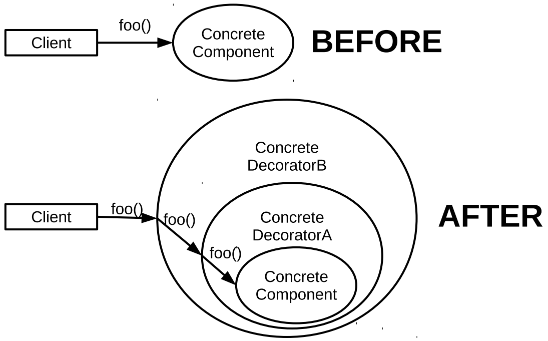


Inheritance is used to make sure that components and decorators **share** the same interface: namely the **public interface of Component** which is either an **abstract class** or an **interface**

At run-time, concrete decorators **wrap** concrete components and/or other concrete decorators the object to be wrapped is typically passed in **via the constructor**

ROAR

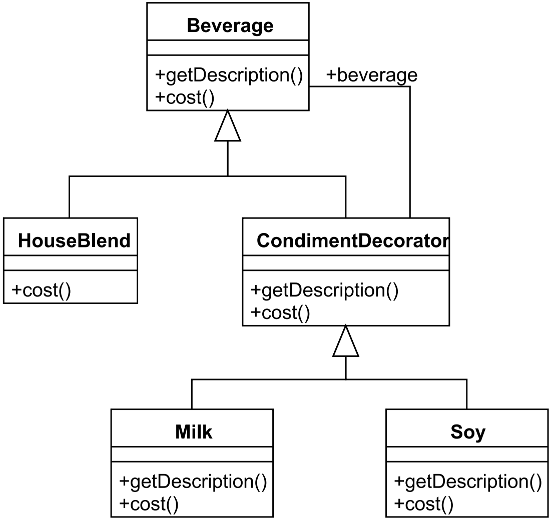
Client Perspective



In both situations `Client` thinks its talking to a `Component`. It shouldn't know about the concrete subclasses. Why?

Idaho State University Computer Science

StarBuzz Using Decorators (Incomplete)



ROAR

Expanding Starbuzz

Individual Exercise

❶ Download the code from Moodle for the Decorator pattern.

❷ Expand the program to allow for the following Beverage Combinations (add prices and descriptions as necessary)

- Irish Coffee: Coffee + Whiskey. Starts your morning off with hair of the dog.
- Tea with Honey
- Tea with Milk
- Tea with Honey and Milk
- Hot Toddy: Whiskey + Lemon + Cinnamon Stick (optional)

❸ Update the Driver to showcase your new drinks

If you finish early, work with your neighbors to help them complete it.

Are there any questions?