UNIVERSITY of GREENWICH

Alliance with FPT Education

ADVANCED PROGRAMMING

Group 10

Group 10

Insert the title of your subtitle Here

Tran Quang Huy

GCD18457

Nguyen Van Hieu

GCD17175

INTRODUCTION

OOP

UNIFIED
MODELING
LANGUAGE

Agenda

1. Document about OOP
2. Design Pattern
3. Relationship between OOP and Design Pattern
4. Improvement

Document about OOP

OOP Characteristics

Insert the title of your subtitle Here

01

Encapsulation

is defined as “The process of packaging one or more items inside a logical or physical package". Encapsulation, in OOP method, prevents access to implementation details of Implementation.

02

Inheritance

Inheritance allows us to define a class in terms of another class, which makes it easier to create and maintain an application. This also provides an opportunity to reuse code features and faster execution times.

03

Polymorphism

The word polymorphism means many forms. In object-oriented programming, polymorphism is often expressed as "one interface, many functions".

04

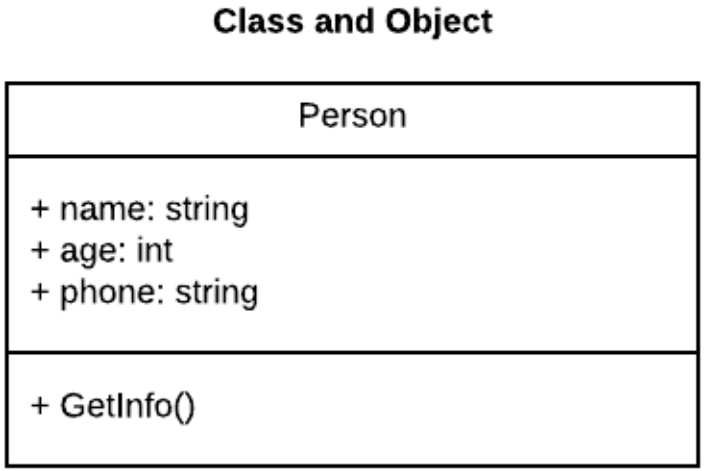
Abstraction

Abstraction is the process of producing only the basic information of an object in the real world and hiding all details of an object. It is based on interface splits and interface implementations.

Class and Object

Definition

a class describes the contents of the objects that belong to it: it describes an aggregate of data fields (called instance variables), and defines the operations (called methods). object: an object is an element (or instance) of a class; objects have the behaviors of their class.

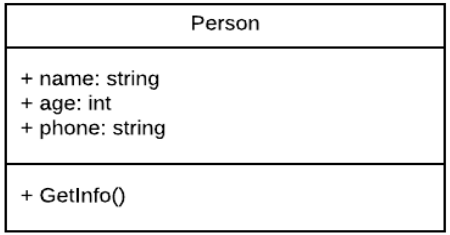


Encapsulation

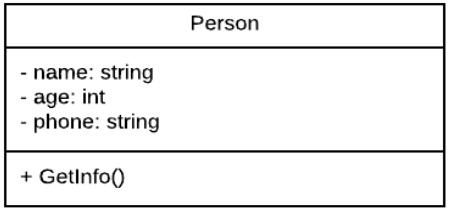
Define

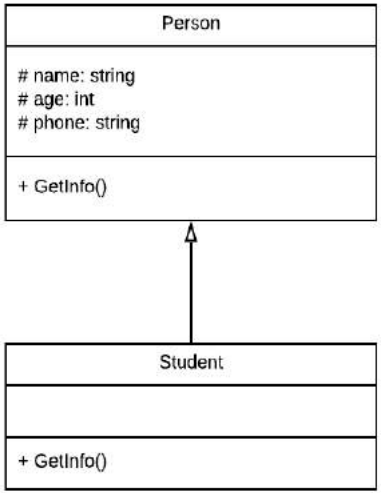
In c#, Encapsulation is a proces s of binding the data members and member functions into a sin gle unit.
Generally, in c# the encapsulati on is used to prevent an alterati on of code (data) accidentally fr om the outside of functions. In c #, by defining the class fields wi th properties we can protect the data from accidental corruption.

Class and Object



Encapsulation

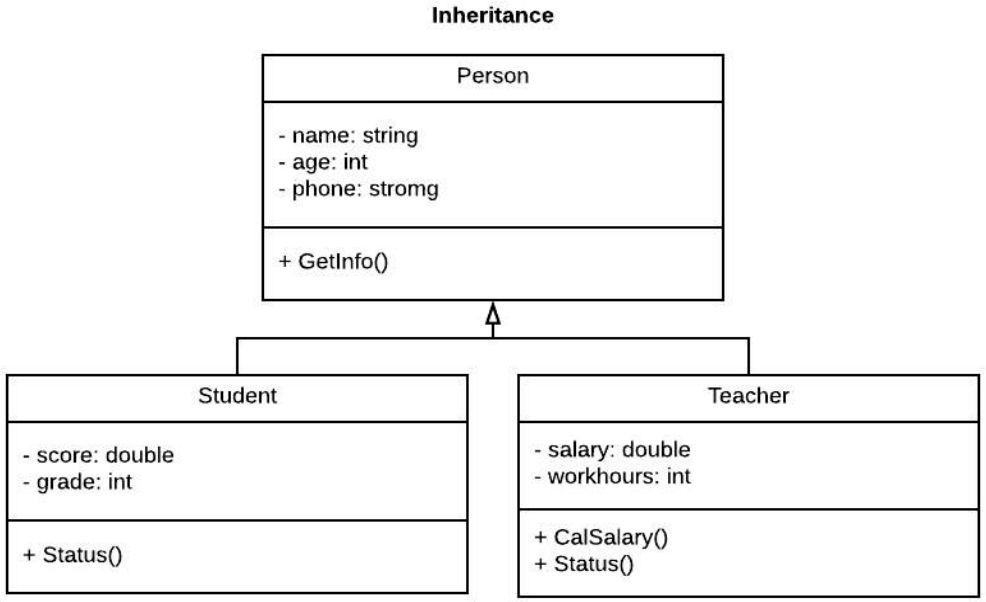




Inheritance

Define

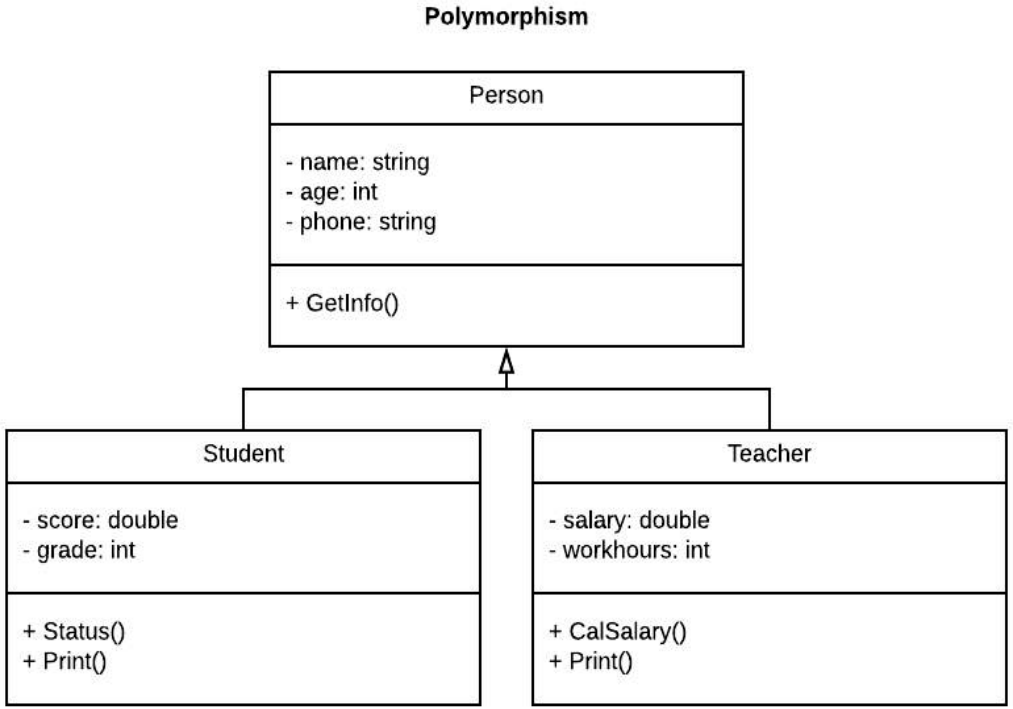
Inheritance is an important concept in C#. Inheritance is a concept in which you define parent classes and child classes. The child classes inherit methods and properties of the parent class, but at the same time, they can also modify the behavior of the methods if required



Polymorphism

Define

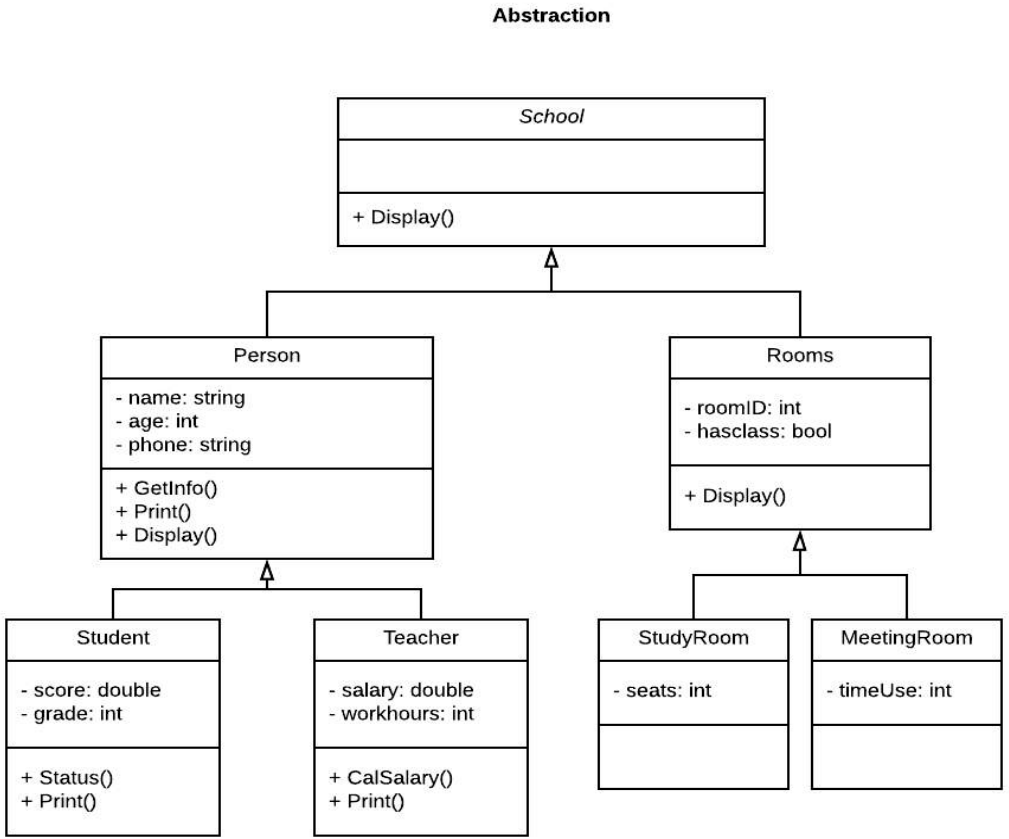
Polymorphism, in C#, is the ability of objects of different types to provide a unique interface for different implementations of methods. It is usually used in the context of late binding, where the behavior of an object to respond to a call to its method members is determined based on object type at run time.



Abstraction

Define

The process of defining a class by providing the necessary and essential details of an object to the outside world and hiding the unnecessary things is called abstraction in C#. It means we need to display what is necessary and compulsory and need to hide the unnecessary things to the outside world.

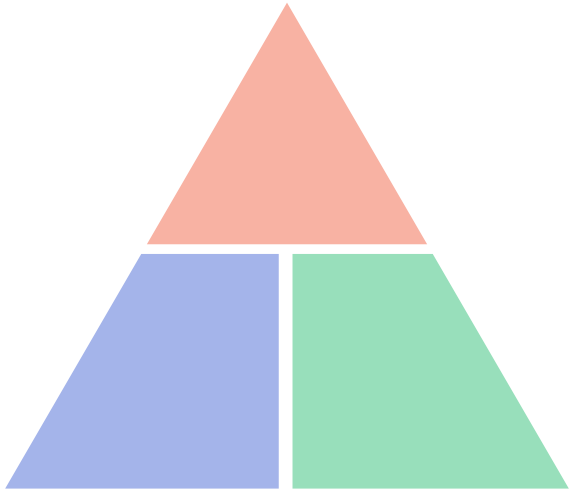


Design Pattern

Main Design Pattern

Creational Pattern

Creational design patterns are design patterns that deal with object creation mechanisms, trying to create objects in a manner suitable to the situation. The basic form of object creation could result in design problems or in added complexity to the design.



Structural Pattern

Structural design patterns are design patterns that ease the design by identifying a simple way to realize relationships among entities. ... Retrofit Interface Pattern: An adapter used as a new interface for multiple classes at the same time.

Behavioral Pattern

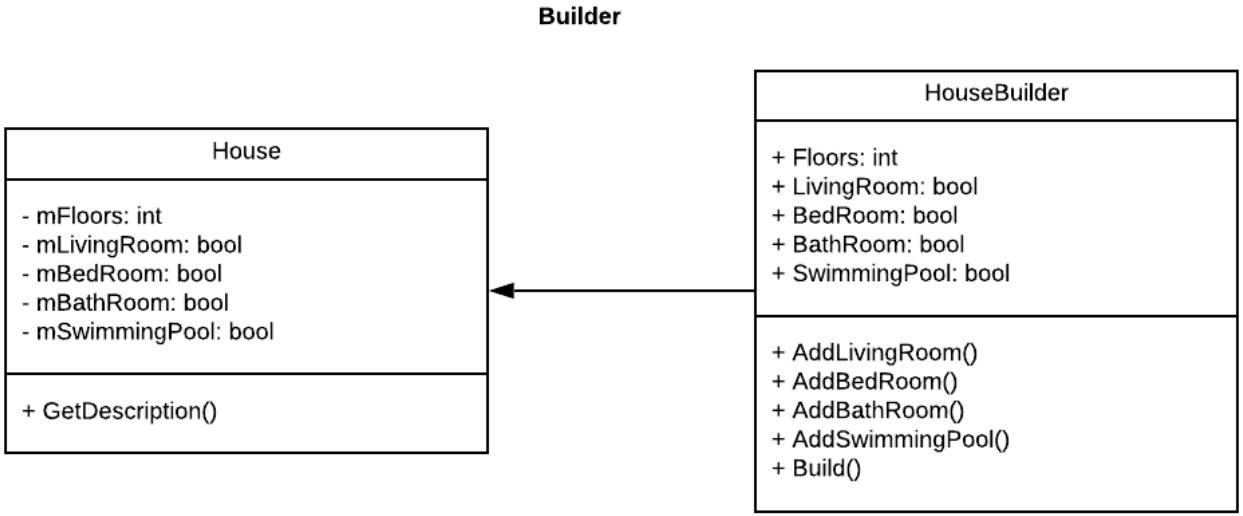
Behavioral design patterns are design patterns that identify common communication patterns among objects and realize these patterns. By doing so, these patterns increase flexibility in carrying out this communication.

Creational – Builder Pattern

The Builder is a design pattern designed to provide a flexible solution to various object creation problems in object-oriented programming. The intent of the Builder design pattern is to separate the construction of a complex object from its representation. It is one of the Gang of Four design patterns.

- Separate construction of a complex object from its representation so that the same construction process can create different representations.
- Parse a complex representation, create one of several targets.
- Difference between Builder and Abstract Factory
- Builder focuses on constructing a complex object step by step. Abstract Factory emphasizes a family of product objects - simple or complex.
- Builder returns the product as a final step, but as far as the Abstract Factory is concerned, the product gets returned immediately.

Class Diagram of Builder

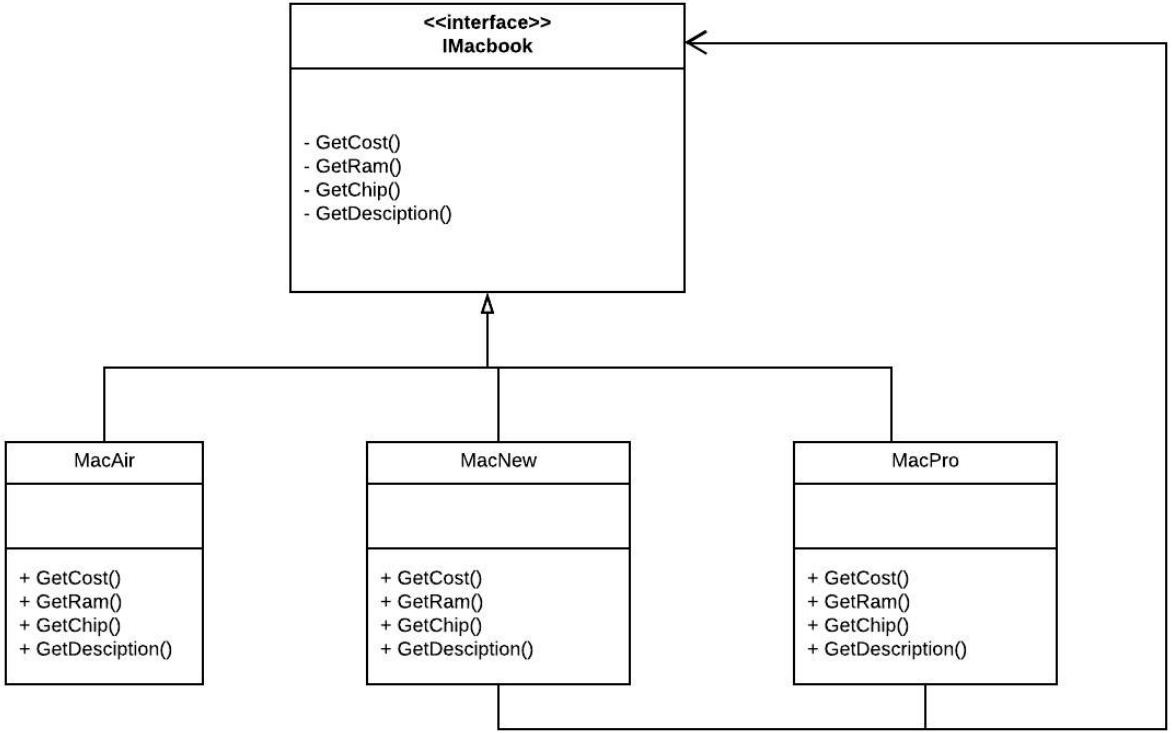


Structural – Decorator Pattern

In object-oriented programming, the decorator pattern is a design pattern that allows behavior to be added to an individual object, dynamically, without affecting the behavior of other objects from the same class. The decorator pattern is often useful for adhering to the Single Responsibility Principle, as it allows functionality to be divided between class es with unique areas of concern. The decorator pattern is structurally nearly identical to the chain of responsibility pattern, the difference being that in a chain of responsibility, exactly one of the classes handles the request, while for the decorator, all classes handle the request.

- Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to sub classing for extending functionality.
- Client-specified embellishment of a core object by recursively wrapping it.
- Wrapping a gift, putting it in a box, and wrapping the box
- You want to add behavior or state] to individual objects at run-time. Inheritance is not feasible because it is static and applies to an entire class.

Class Diagram of Decorator



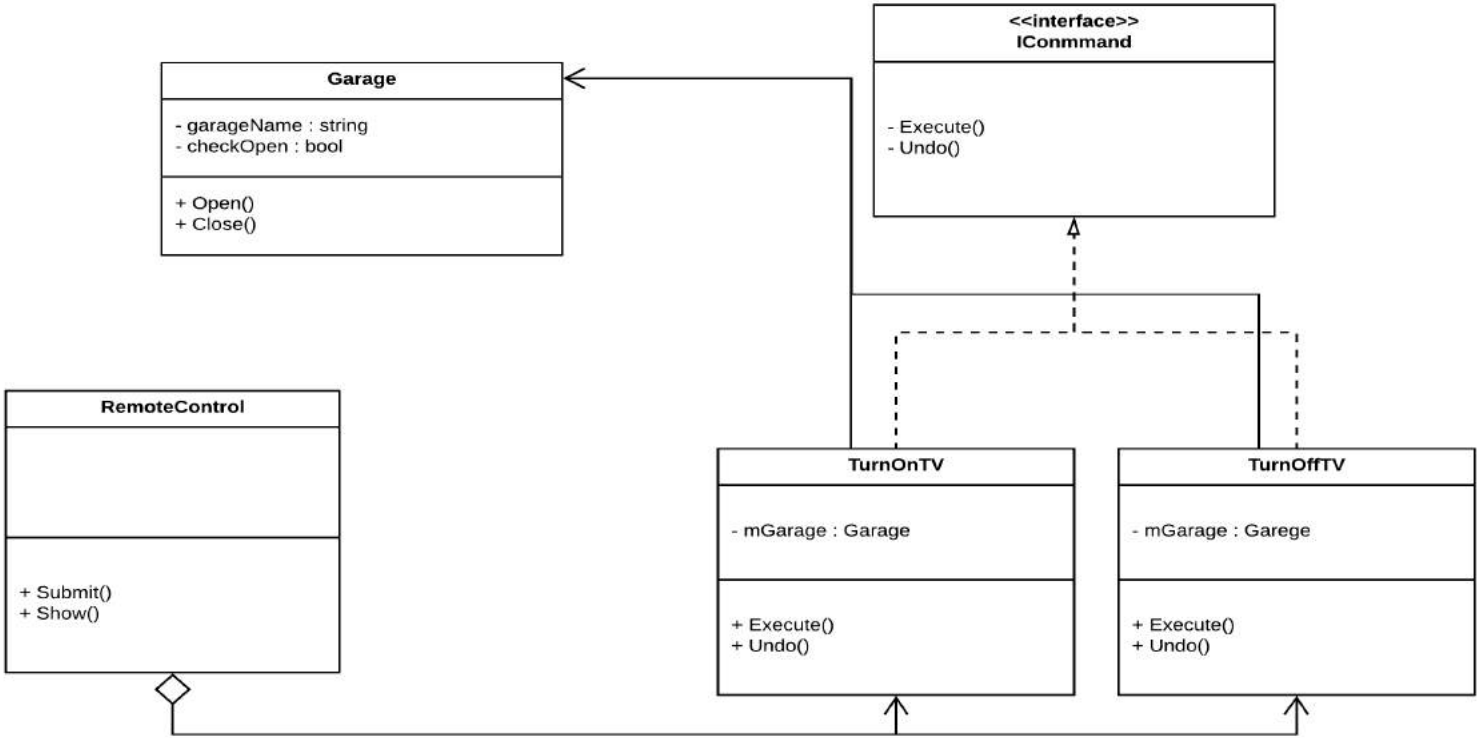
Behavioral – Command Pattern

In object-oriented programming, the command pattern is a behavioral design pattern in which an object is used to encapsulate all information needed to perform an action or trigger an event at a later time. This information includes the method name, the object that owns the method and values for the method parameters.

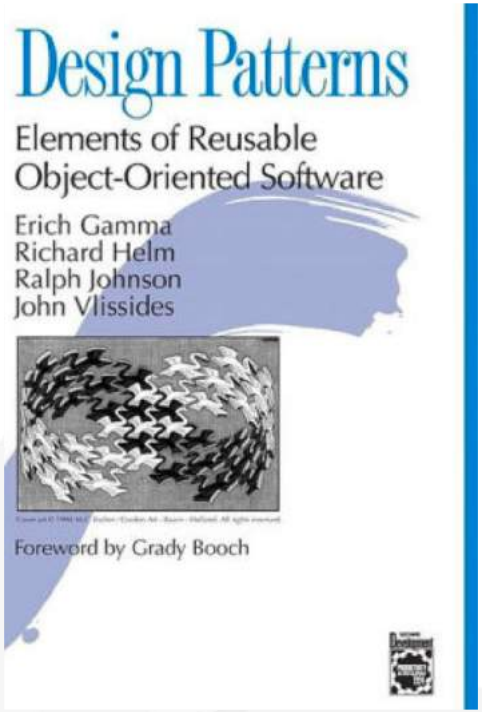
Command means command. The commanding officer is called a commander, who does not do it but orders others to do it. As such, there must be a person who receives and exe cutes the order.

Command - provides a class that encapsulates commands

Class Diagram of Command



Relationship between OOP and Design Pattern



Software that object-oriented software is hard, and designing reusable object-oriented software is even harder

Yet experienced object-oriented designers do make good designs

Why Design Pattern born because OOP problem

These patterns solve specific design problems and make object-oriented designs more flexible, elegant, and ultimately reusable

Design patterns help a designer get a design "right" faster

Design Pattern Solve Problem

Finding Appropriate Objects

The Strategy pattern
The State pattern

Determining Object Granularity

The Facade pattern
Abstract Factory and Builder
Visitor and Command

Specifying Object Interfaces

The Memento pattern
Decorator and Proxy pattern
The Visitor pattern

Class versus Interface Inheritance

The Composite pattern
The Command, Observer, State, and Strategy patterns

Programming to an Interface, not an Implementation

Abstract Factory, Builder, Factory Method, Prototype and Singleton

Why are design patterns and OOP treated separately?

Because they are different subjects. In general, you learn to program, then you learn how to think about programming.

39

days.

Learning to program is not hard for many people. It's like playing with Legos: there's a handful of different pieces you get to snap together however you want. Sometimes you make something cool, but most of the time you make crap =). Usually, the longer you play, the better you get.

Studying Design Patterns is learning *good ways to build your programs*. You're essentially reading advice from people that have been building things for decades. They've distilled their most common solutions into simple, digestible tidbits of knowledge with memorable names. It's like an apprenticeship for the digital-age: your elders are giving you their best advice. You can take it and be ahead of the game, or ignore it and repeat all of their mistakes.

Why are design patterns and OOP treated separately? Because they are different subjects. In general, you learn to program, then you learn how to *think about programming*. I wish it were the other way around, but I'm not holding my breath.

Is someone that knows Design Patterns necessarily an OOP expert? No.

share improve this answer

answered Jan 26 '09 at 3:46

https://stackoverflow.com/questions/478773/how-is-oop-and-design-patterns-related

Why are design patterns and OOP treated separately?

Design patterns describe object-oriented designs, they are based on practical solutions that have been implemented in mainstream object-oriented programming languages. Each design pattern focuses on a particular object-oriented design problem or issue. It describes when it applies, whether it can be applied in view of other design constraints, and the consequences and trade-offs of its use.

https://www.bookdepository.com/Design-Patterns-Erich-Gamma/9780201633610

A Developer can programming in OOP without using Design Patterns, but if Developer want to be good at building flexible, reusable or maintainable system. They have to find out what is Design Patterns

Design Patterns do not need to be Object-Oriented

- Gang of Four was a team of four members: Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides.

In my opinion OOP is not a design pattern. OOP is basic. Design Patterns are approaches to implement OOP concept

Improvement

Welcome to HighLands Coffee

Insert the title of your subtitle Here

The Highlands Coffee has made for itself as the fastest growing coffee shop around. If you've seen one on your local corner, look across the street; you'll see another one.

Because they've grown so quickly, they're scrambling to update their ordering systems to match their beverage offerings.

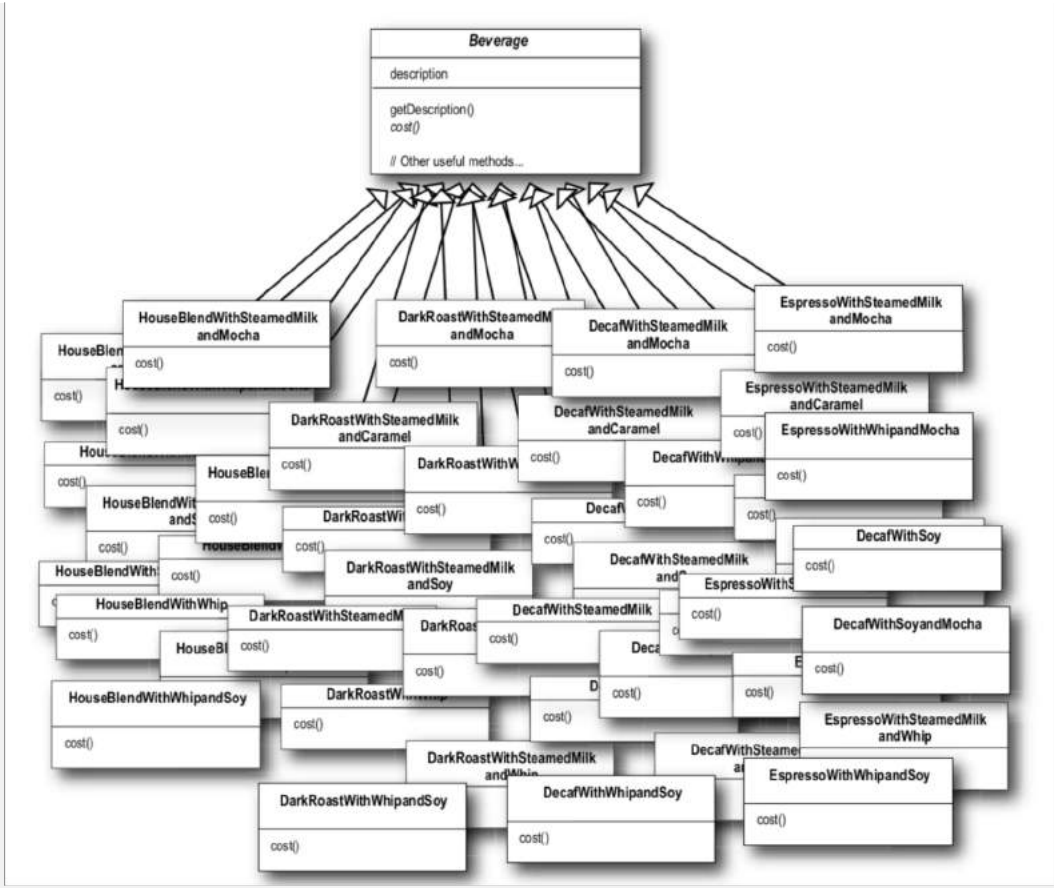
The manager of Highlands coffee has defined several requirements that need to be fulfilled the order system:

- The coffee shop serves some kinds of coffee (House Blend, Dark Roast, Decaf, Espresso).
- The coffee can be combined with additional Condiments like steamed milk, soy, mocha (Chocolate) and have it all topped off with whipped milk.
- Highlands charges a bit for each of these, so they really need to get them built into their order system.

The Coffee shop has a menu card include coffee and condiments like table below:

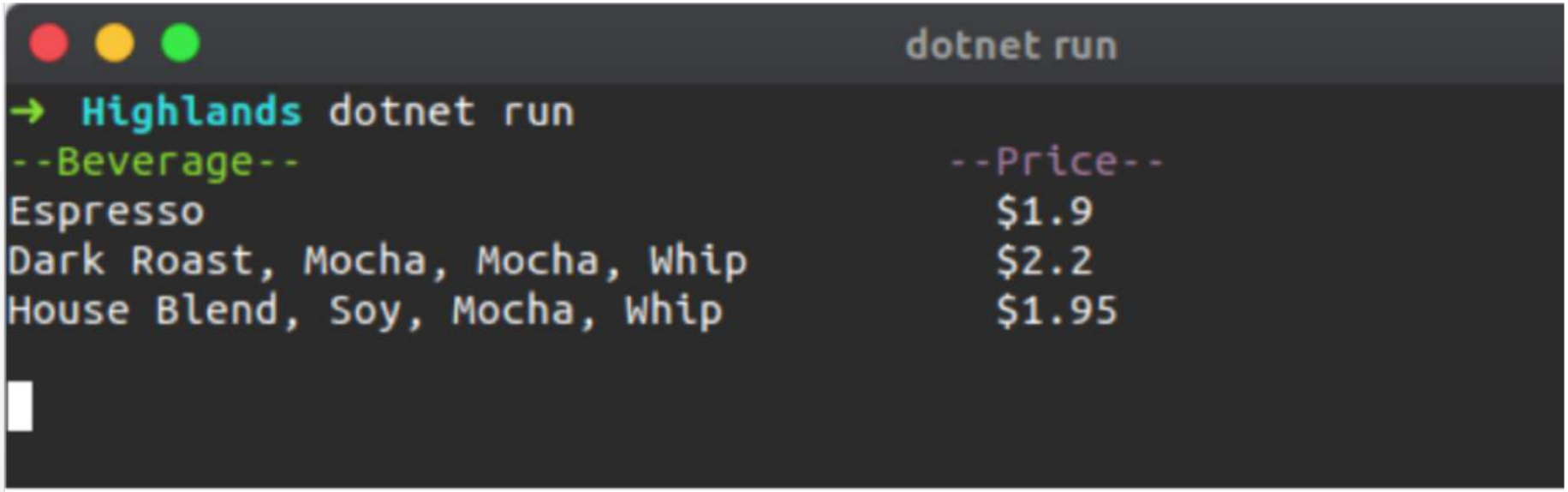
Highlands Coffee	
Coffees	
House Blend	1.2
Dark Roast	1.4
Decaf	1.7
Espresso	1.9
Condiments	
Steamed milk	.20
Mocha	.30
Soy	.25
Whip	.20

Before using Decorator Pattern



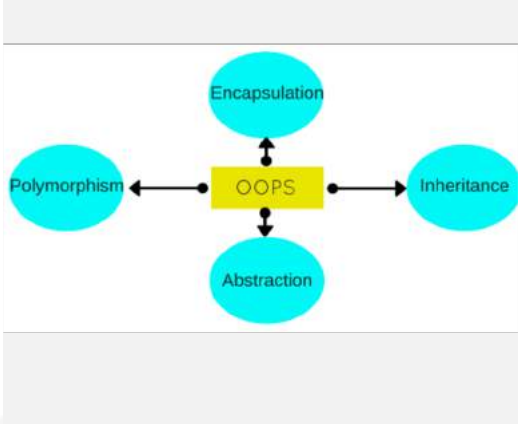
After using Decorator Pattern

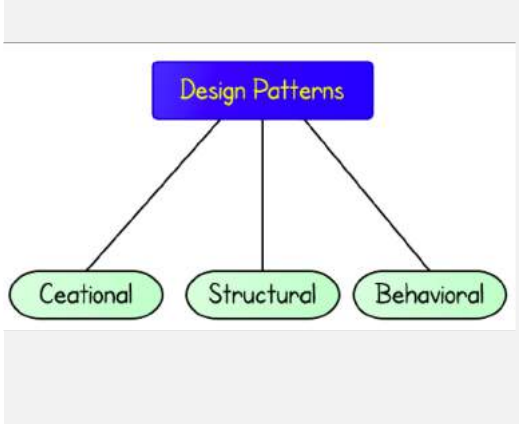
Result of using Decorator Pattern

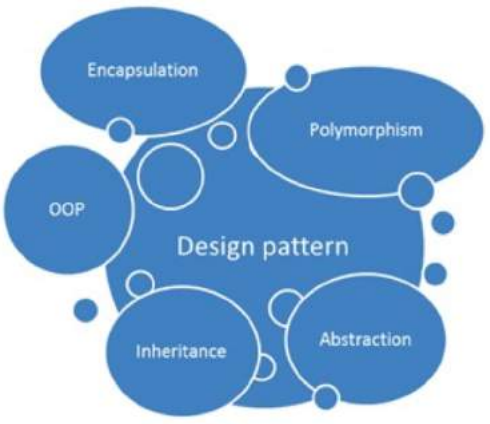


Conclusion

Insert the title of your subtitle Here







01

4 characteristic of OOP

02

Design Pattern

03

Relationship between OOP and Design Pattern

Thank you

Group 10