DESIGN PATTERN PRINCIPLES

* DESIGN PRINCIPLES
  + The first of many design principles
  + 1. Identify the aspects of your application that vary and separate them from what stays the same
    - Take what varies and “encapsulate” it so it won’t affect the rest of your code.
    - The result? Fewer unintended consequences from code changes and more flexibility in your systems!
    - Take the parts that vary and encapsulate them, so that later you can alter or extend the parts that vary without affecting those that don’t
    - As simple as this concept is, it forms the basis for almost every design pattern. All patterns provide a way to let some part of a system vary independently of all other parts.
  + 2. Program to an interface, not an implementation
    - PROGRAMMING TO AN IMPLEMENTATION WOULD BE :

Dog d = new Dog()

d.bark();

NOTE

Declaring the variable “d” as type Dog( a concrete implementation of Animal) forces us to code to a concrete implementation.

* + - BUT PROGRAMMING TO AND INTERFACE/SUPERTYPE WOULD BE:

Animal animal = new Dog();

Animal.makesound();

NOTE

We know it’s a dog, but we can now use the animal reference polymorphically

* + - EVEN BETTER, RATHER THAN HARDCODING THE INSTANTIATION OF THE SUBTYPE( LIKE NEW DOG()) INTO THE CODE, ASSIGN THE CONCRETE IMPLEMENTATION OBJECT AT RUNTIME

a = getAnimale();

a.makeSound();

NOTE

We don’t know what the actual animal subtype is… all we care about is that it knows how to respond to makeSound()

Location 605 IN *Head First Design Patterns: A Brain-Friendly Guide*

* + - Pay careful attention to the relationships between the classes. In fact, grab your pen and write the appropriate relationship (IS-A, HAS-A, and IMPLEMENTS) on each arrow in the class diagram.
  + 3. Favor composition over inheritance
    - Creating system using composition gives you a lot more flexibility. Not only does it let you encapsulate a family of algorithms into their own set of classes, but it also lets you change behavior at runtime as long as the object you’re composing with implements the correct behavior interface.
    - Composition is used in many design patterns and you’ll see a lot more about its advantages throughout the book