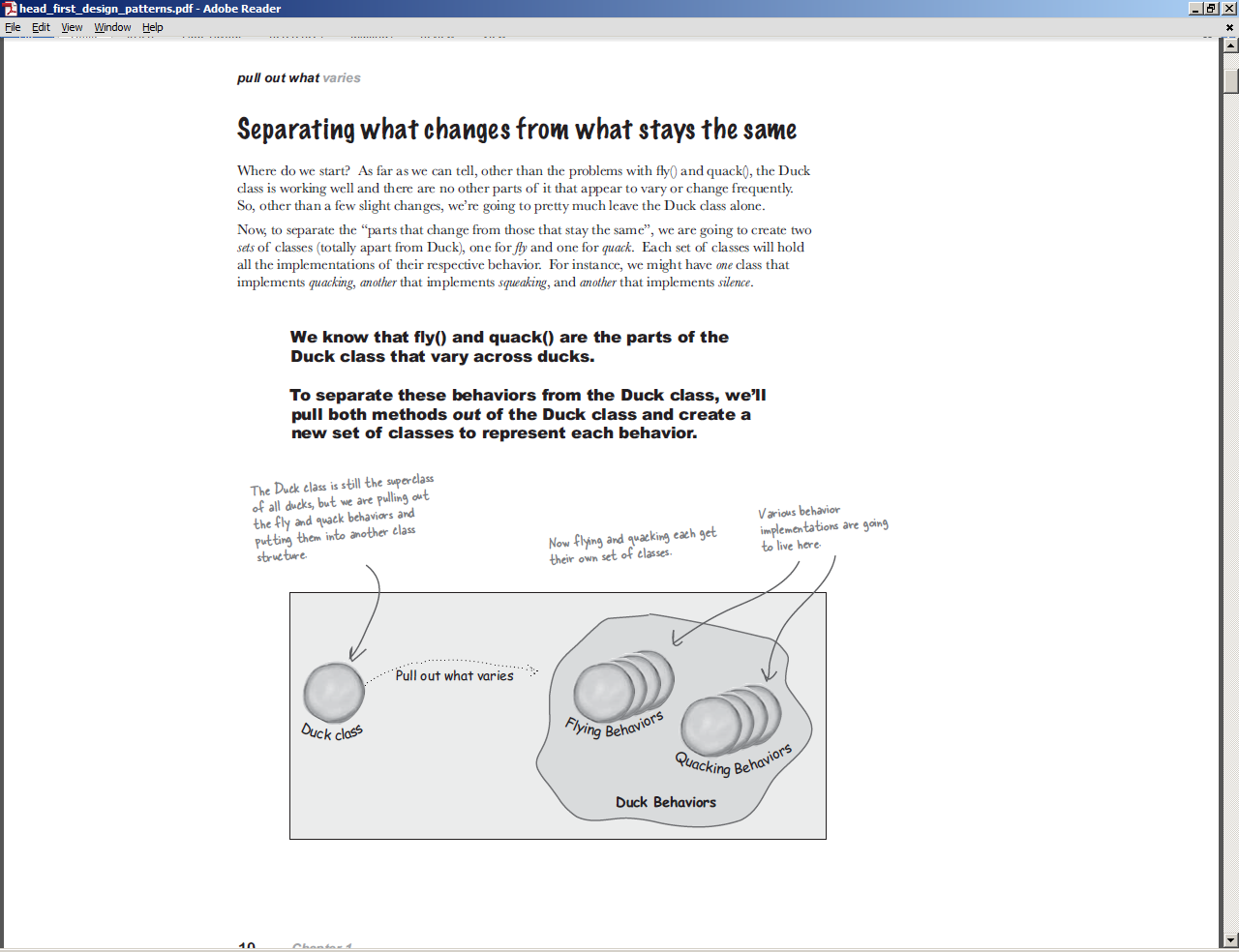
Design Patterns

If the behavior can be changed frequently, we need to create the separate classes for those behaviors.

Eg:



# Types:

1. Creational patterns

1. The Singleton pattern
2. The Factory Method pattern

2. Structural patterns

1. The Adapter pattern
2. The Proxy and Decorator patterns
3. The Composite pattern

3. Behavioral patterns

1. The Observer pattern
2. The Strategy and Template patterns

4. Concurrency patterns

1. The Single Thread Execution pattern

## FACTORY PATTERN:



## ABSTRACT FACTORY PATTERN

# 

## SINGLETON PATTERN



## BUILDER PATTERN



## PROTOTYPE PATTERN:

Prototype pattern refers to creating duplicate object while keeping performance in mind. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

This pattern involves implementing a prototype interface which tells to create a clone of the current object. This pattern is used when creation of object directly is costly. For example, an object is to be created after a costly database operation. We can cache the object, return its clone on next request and update the database as and when needed thus reducing database calls.



## ADAPTER PATTERN:



## COMPOSITE PATTERN:



## DECORATOR PATTERN:



## FACADE PATTERN:



## PROXY PATTERN:

A proxy, in its most general form, is a class functioning as an interface to something else. The proxy could interface to anything: a network connection, a large object in memory, a file, or some other resource that is expensive or impossible to duplicate. In short, a proxy is a wrapper or agent object that is being called by the client to access the real serving object behind the scenes.

