Design Patterns Tutorial Notes

Kayllen Company
Dave Retterer

Table of Contents

Introduction 1
Adapter Pattern 1
UML Diagram of AdapterTemplate 2
Class Diagram of AdapterDemo 2
Adapter Demo Narrative 3

Introduction

The tutorials in this group describe the structure of many of the "Gang of Four" (GoF) design patterns described in the reference book titled *Design Patterns: Elements of Reusable Object-Oriented Software*. The author has taught this material to college students for many years and believes that by studying this material developers can improve their skills as software and data designers while developing a more powerful skillset with object-oriented software development.

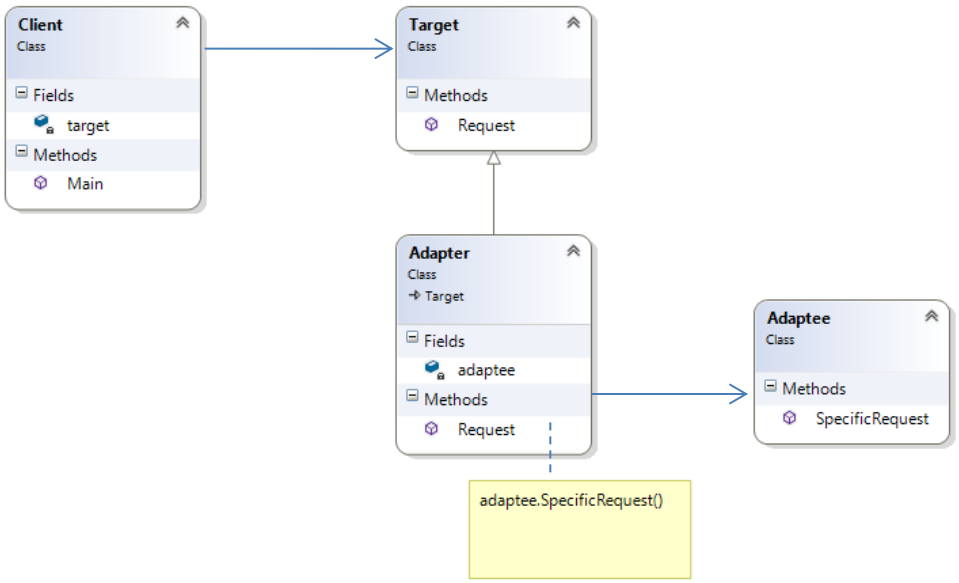
The patterns are presented in alphabetical order by the name in the GoF book with each pattern starting at the top of a page. The Table of Contents provides navigation via the clickable page numbers. All demonstration applications are Windows Form apps written in C# and is available for free at github.com/KayllenTutorials. The video portion of tutorials is available for free at KayllenTutorials on YouTube. Some tutorials have ads to help support this effort. Your subscription to these sites also helps to support the effort.

The narratives included in this document are, in part, the narrative used in the videos. No maximum time has been set for the videos but the intent is for them to be as short as possible to properly introduce the pattern. Further study and use in practice will be helpful to master the pattern's use.

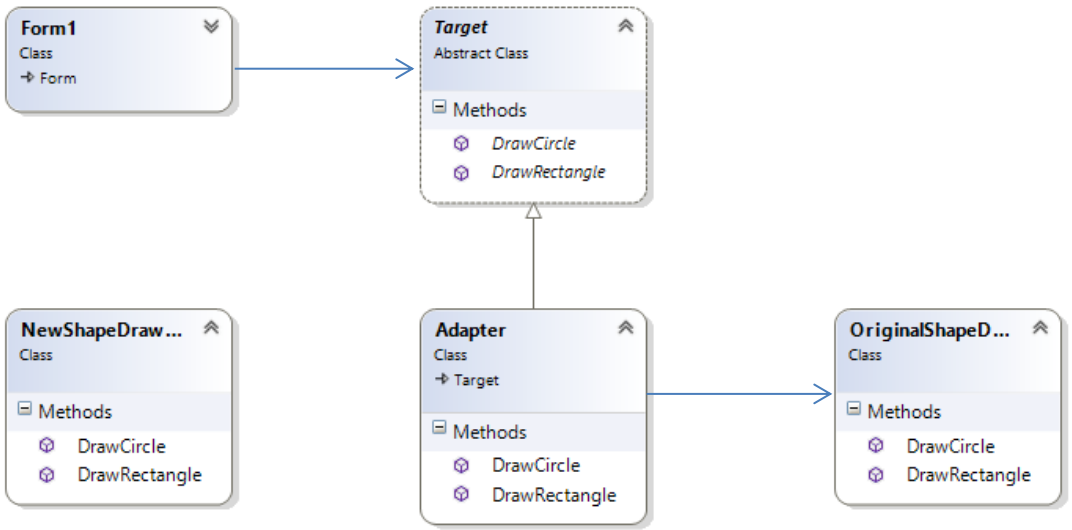
Adapter Pattern

The intent of the Adapter Pattern is to provide a standard way to convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces. Another name for an adapter is a wrapper.

UML Diagram of AdapterTemplate



Class Diagram of AdapterDemo



Adapter Demo Narrative

- In the sample application, the NewShapeDrawing class is not part of the adapter pattern. That class would be written independent of the OriginalShapeDrawing class and used instead of the Adapter Pattern.
- The class Form1 houses the means for an external actor to interact with the application.
- The class Target defines the interface that is exposed to that external actor. Note that if the drawing code already has the desired interface, there is no need for the Adapter pattern. That is the case with the NewShapeDrawingClass. However, if the drawing code that is available (e.g. the legacy class OriginalShapeDrawing) has an interface other than that which is expected by Form1 then the Adapter Pattern is used to adapt the desired interface into the interface available in the legacy code)
- The design pattern implementation can contain more than one Request method as indicated in the UML Diagram . In this example, both DrawCircle and DrawRectangle act as Request methods.
- The job of a Request method in the Adapter is to convert the request with the "new" interface into a similar request with the "old" interface. The old interface may not be able to do everything that is implied by the new interface.
- The signature of the SpecificRequest method(s) in the Adaptee may be completely different than that in the Target interface. That means that the name, return type and parameters can all be different. In fact, the Specific Request may make use of multiple pieces of legacy code to accomplish the adaptation needed. Other patterns can help with that.