Adapter ID

Student:

Izabela Kuźniar

Teacher:

Andrea Corradini

Course:

Software Design Patterns

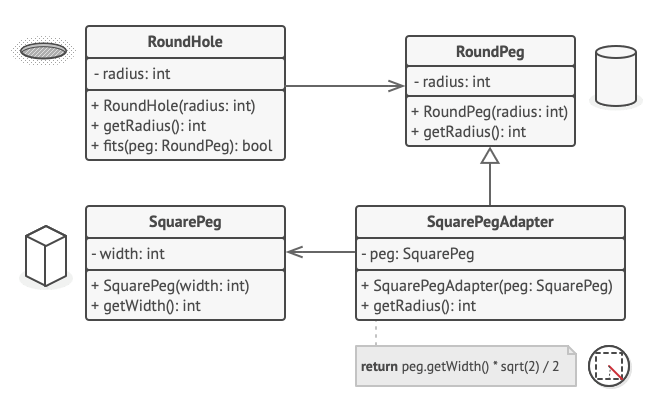
# Name and category

Adapter, also known as Wrapper (like Decorator) is a structural design pattern.

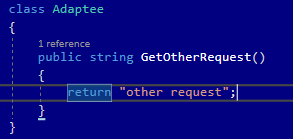
# Intent:

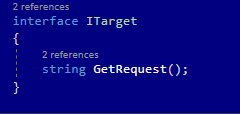
Allows objects with incompatible interfaces to collaborate.

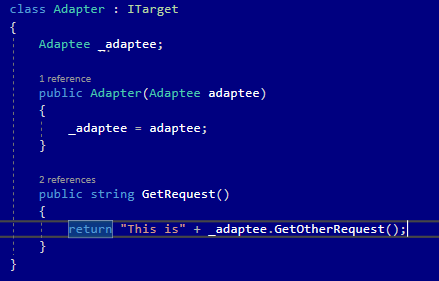
# Structure as a UML class diagram

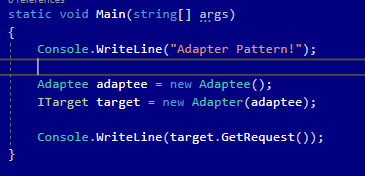


# Implementation:









# Consequences:

Benefits:

* Single Responsibility Principle. Ability to separate interface or data conversion from the primary business logic of the program.
* Open/Closed Principle. Ability to introduce new types of adapters to the program without breaking the existing client code, as long as they work with the adapters through the client interface.

Drawbacks:

* Overall complexity of the code increases because of the introduced set of new interfaces and classes. Sometimes it is better to change the service class so that it matches the rest of your code.

# Related patterns

1. Adapter changes the interface of an existing object, while Decorator enhances an object without changing its interface. In addition, Decorator supports recursive composition, which Adapter does not.
2. Adapter provides a different interface to the wrapper object, Proxy provides it with the same interface, and Decorator provides it with an enhanced interface.
3. Bridge is usually designed up-front, letting you develop parts of an application independently of each other. On the other hand, Adapter is commonly used with an existing app to make some otherwise-incompatible classes work together nicely.
4. Adapter is making existing interface usable, while Façade defines a new one for existing objects. Adapter wraps only one object, Façade an entire subsystem of objects.
5. Bridge, State, Strategy (and to some degree Adapter) have very similar structures. All of those patterns are based on composition, that is delegating work to other objects but they solve different problems. ***A pattern isn’t just a recipe for structuring your code in a specific way. It can also communicate to other developers the problem the pattern solves.***