Adapter
(Design Pattern)

Presenter: **Muhammad Usman (February 10, 2017)**

EECS 6431: Software Re-engineering

Table of Contents

YORK
UNIVERSITÉ
UNIVERSITY

Overview

- It is one of the “Structure patterns”.
  - They are concerned with how classes and objects are composed to form large structures.

- Types of structure patterns:
  - Adapter
  - Decorator
  - Composite
  - Bridge
  - Flyweight, etc.

Intent

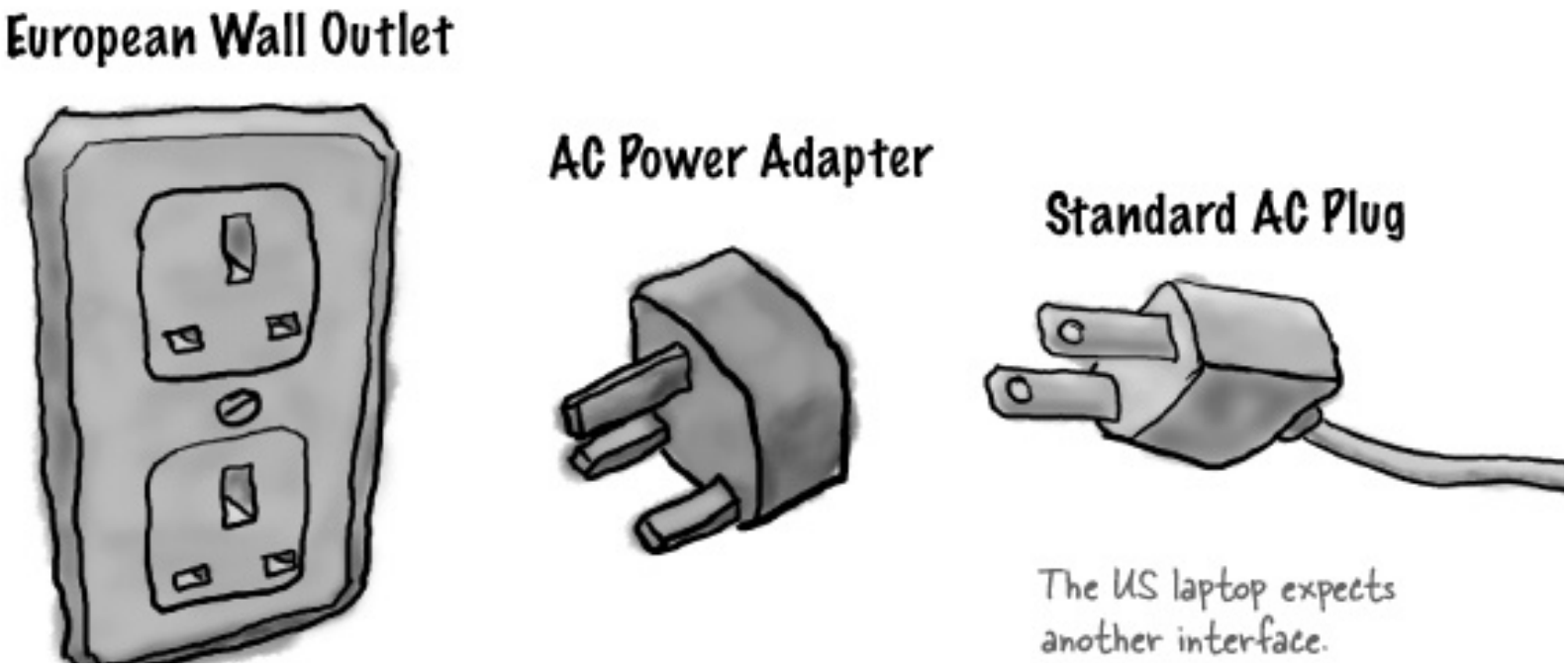
- *Convert the interface of a class into another interface that the clients expect. Adapter lets classes work together that could not otherwise because of incompatible interfaces.*

  [Gang of Four]

- Adapter is also known as “Wrapper”.

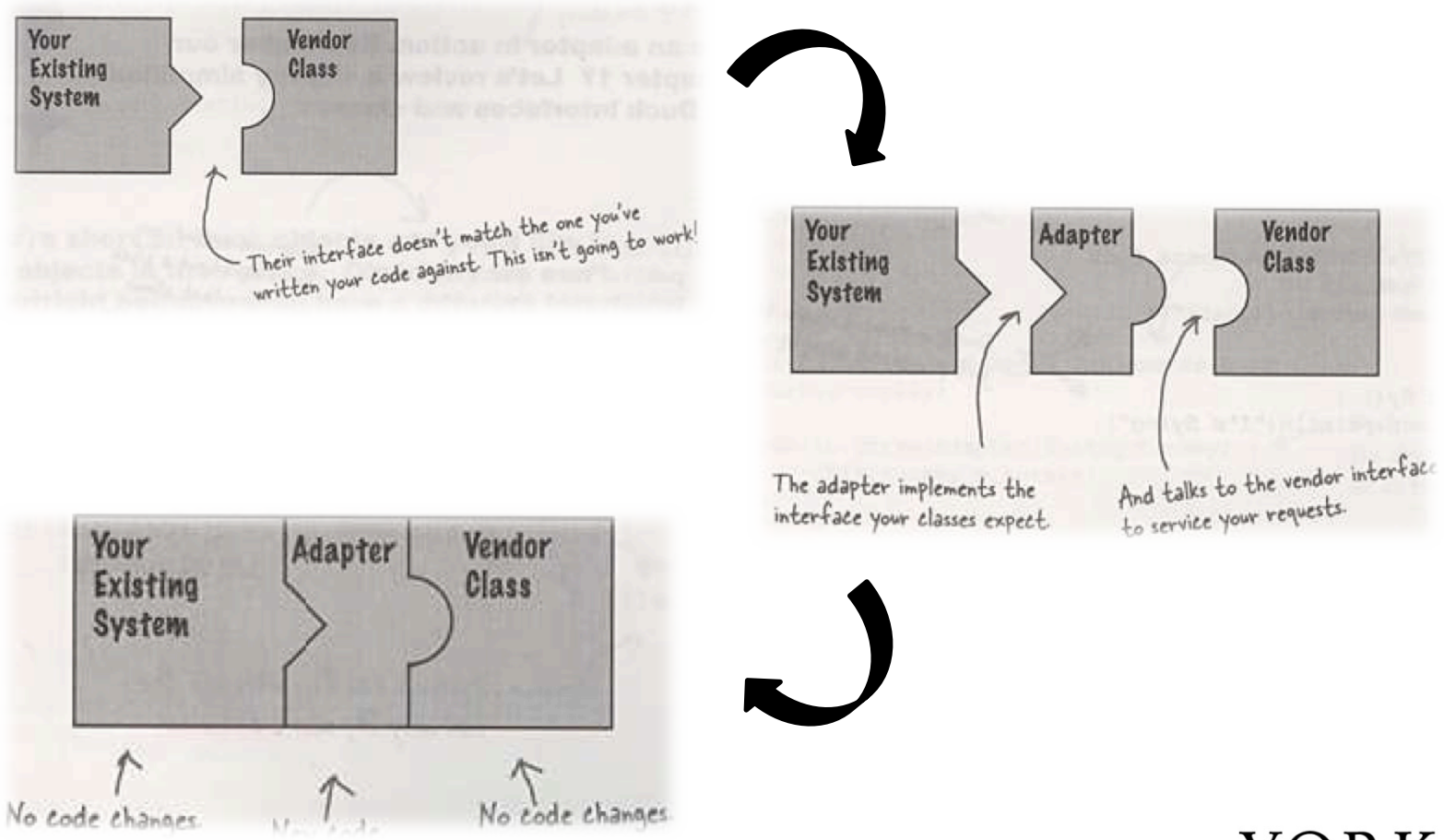
YORK
UNIVERSITÉ
UNIVERSITY

Adapter in Real World



YORK UNIVERSITÉ UNIVERSITY

Adapter in Software Programs



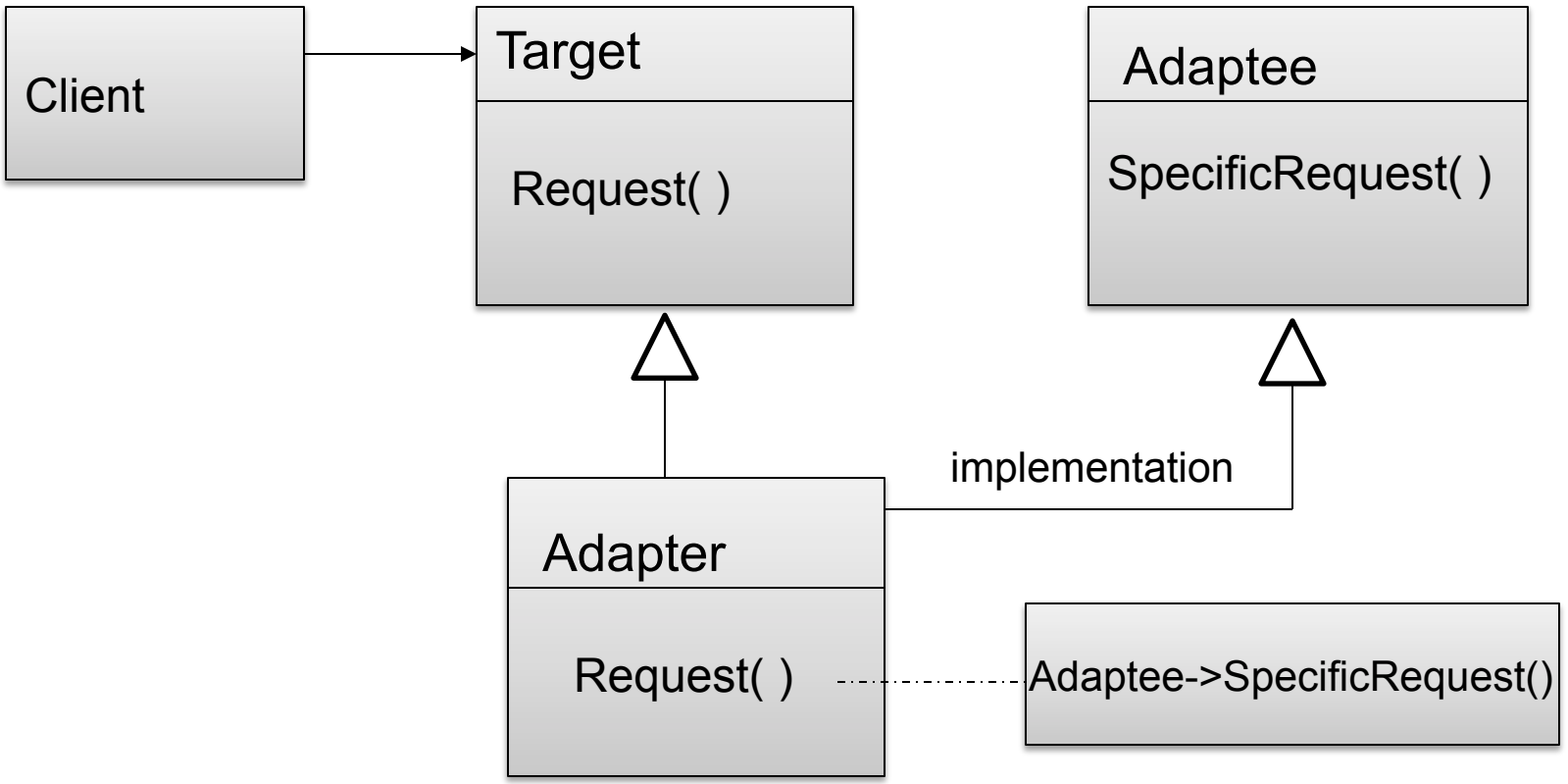
Motivation

- Sometimes a toolkit or class library can not be used because its interface is incompatible with the interface required by an application.
- We can not change the library interface, since we may not have its source code.
- Even if we did have the source code, we probably should not change the library for each domain-specific application.

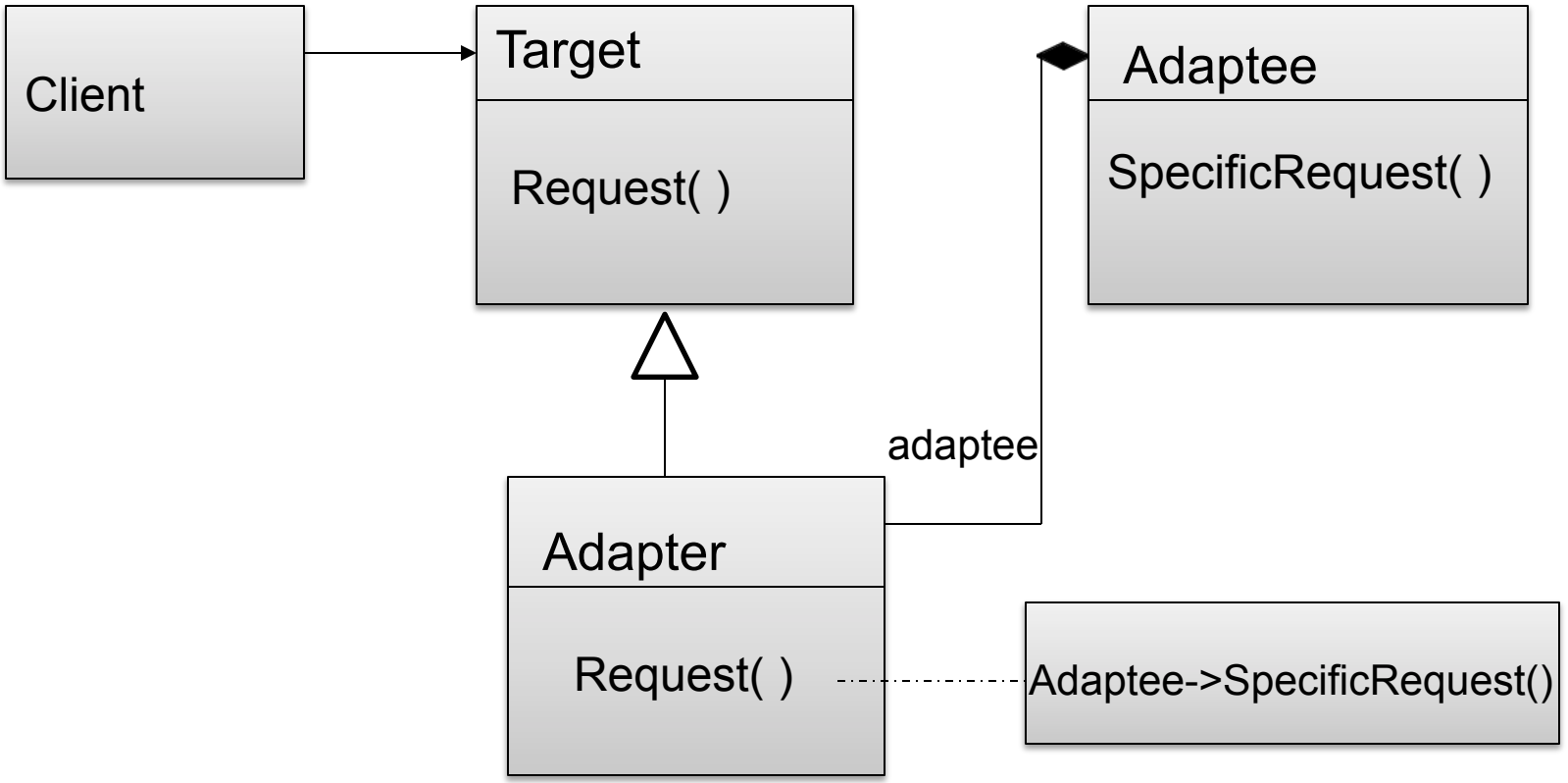
Types of Adapter

- Class Adapter
  - Use multiple inheritance to compose classes
- Object Adapter
  - Object adapters use a compositional technique to adapt one interface to another.

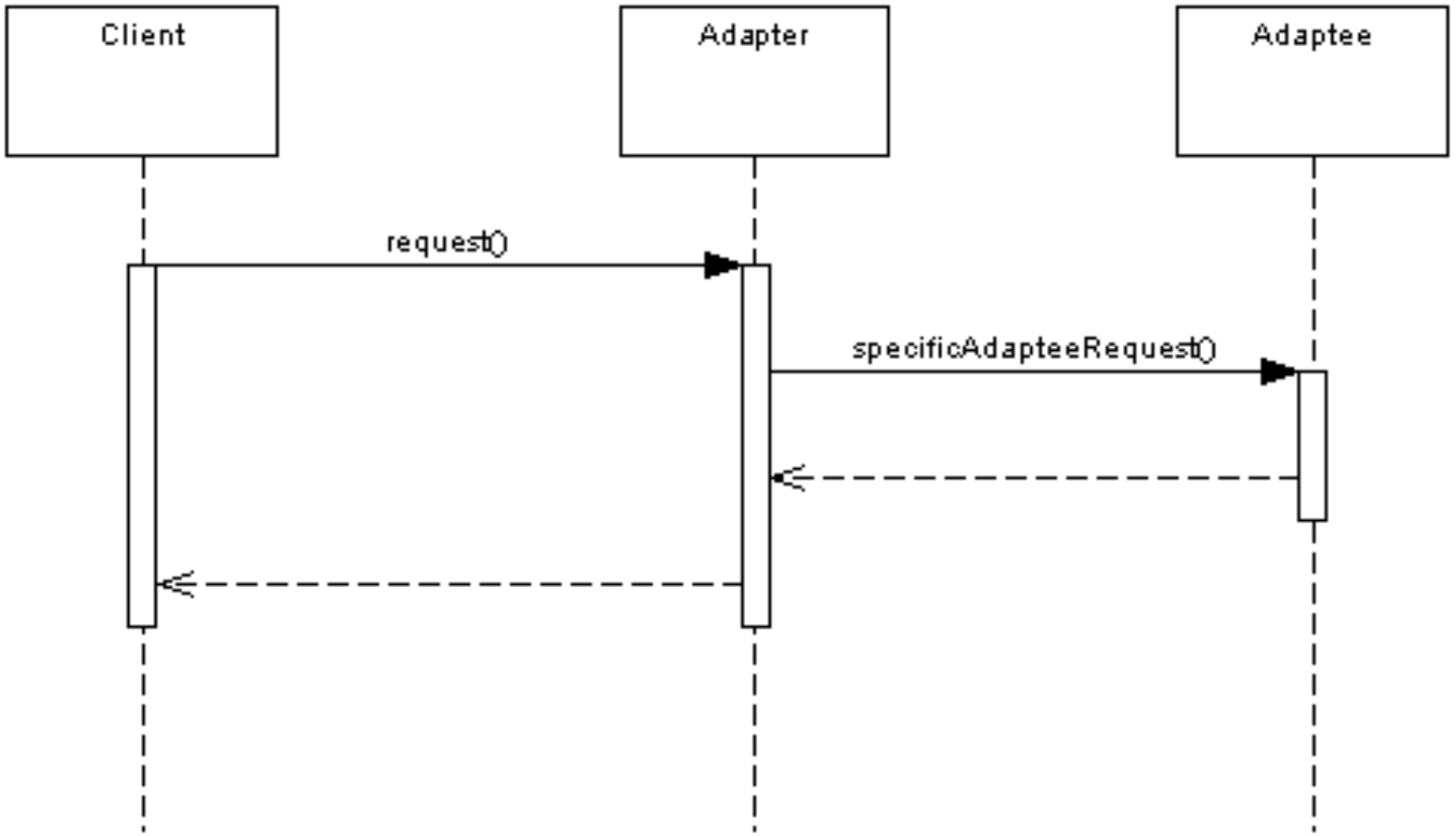
Structure: Class Adapter



Structure: Object Adapter

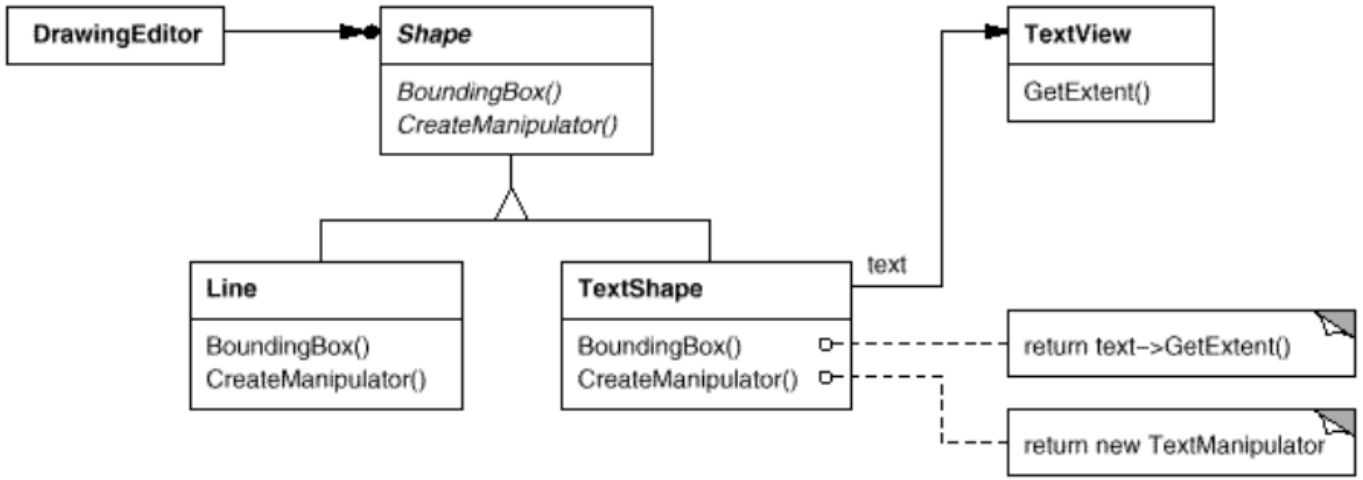


Sequence Diagram



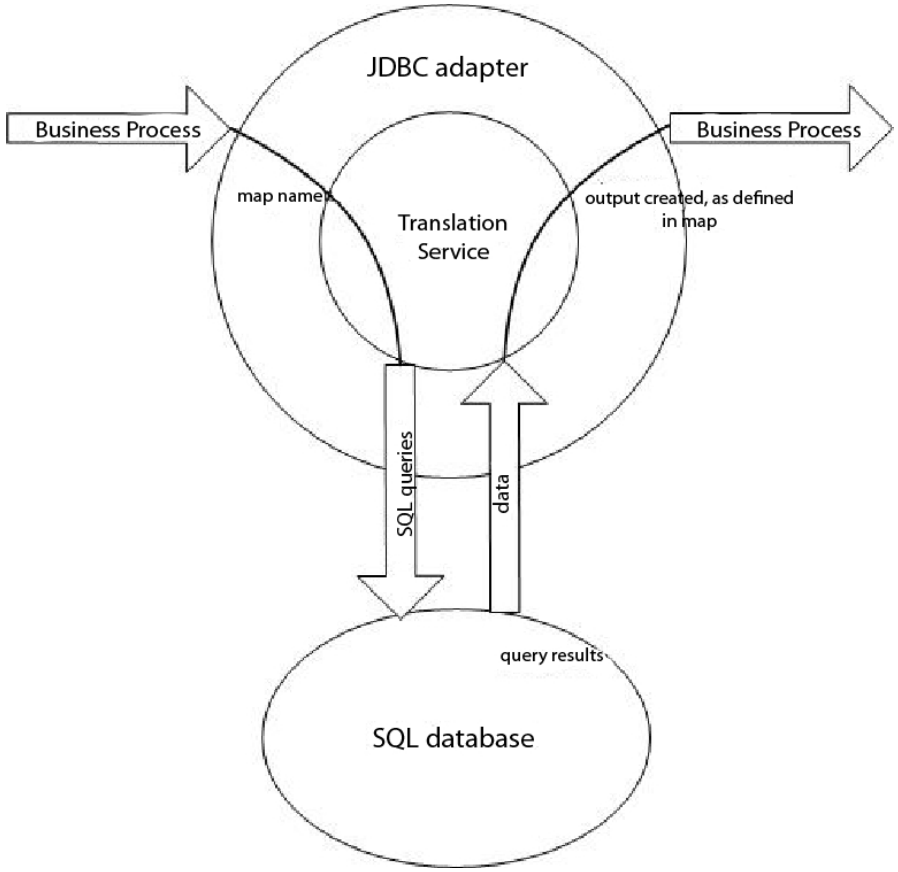
YORK UNIVERSITÉ UNIVERSITY

Example(s)



Example(s)

- Java Data Base Connectivity (JDBC) Adaptor



YORK UNIVERSITÉ UNIVERSITY

Implementation

- How much adapting should be done?
  - For simple interface conversion
    - Change operation names and may be order of arguments
  - When you need different set of operations

- Does the adapter provide two-way transparency?
  - A two-way adapter supports both the Target and the Adaptee interface. It allows an adapted object (Adapter) to appear as an Adaptee object or a Target object

Applicability

- Usually we use adapter when:
  - You want to use an existing class but its interface doesn't match the one you need for your application.
  - You want to use several existing subclasses, but it's impractical to adapt their interface by sub-classing all of them. So an object adapter can adapt the interface of its parent class.

Consequences

- Class Adapter
  - Lets Adapter override some of the Adaptee's behavior by sub-classing.
  - Introduces only one object and no additional pointer indirection is needed to get the Adaptee.
  - Unknown Adaptee subclasses might cause problem

- Object Adapter
  - Lets a single adapter work with a group of Adaptees such as a base class and all its sub classes.
  - The adapter can add functionality to all Adaptees at once.
  - Makes it harder to override Adaptee behavior as the Adapter may not know with what Adaptee it is working with.

References

- Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- https://en.wikipedia.org/wiki/Adapter_pattern

YORK UNIVERSITÉ UNIVERSITY