SENG 350
- Software Architecture & Design

Shuja Mughal

Design Patterns

Fall 2024

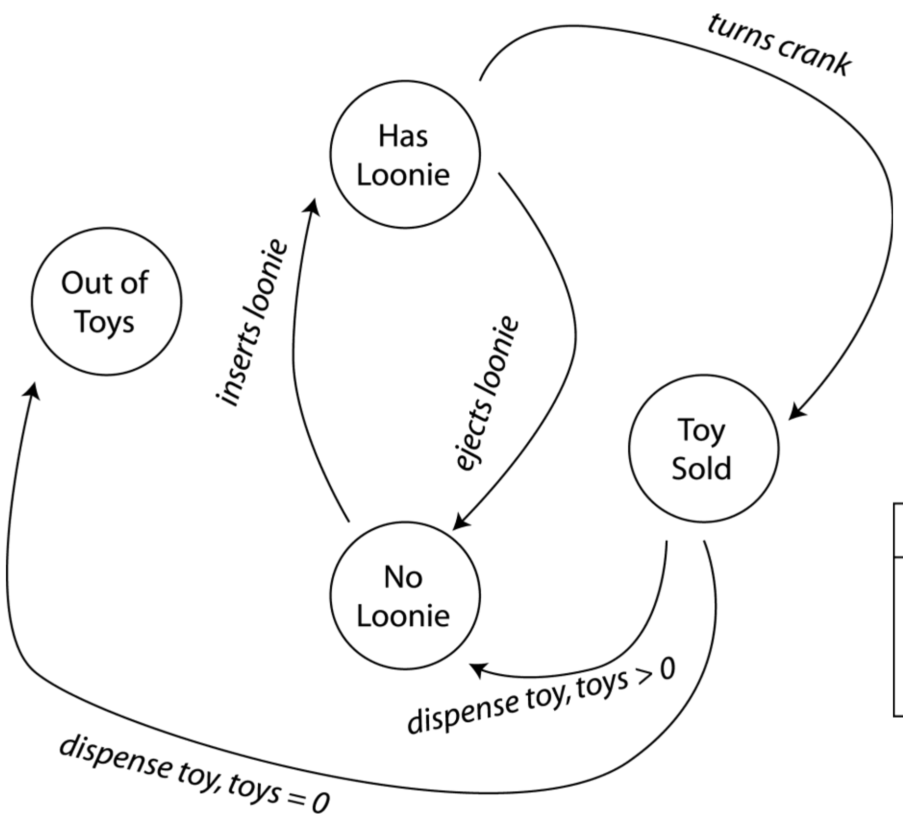
University of Victoria

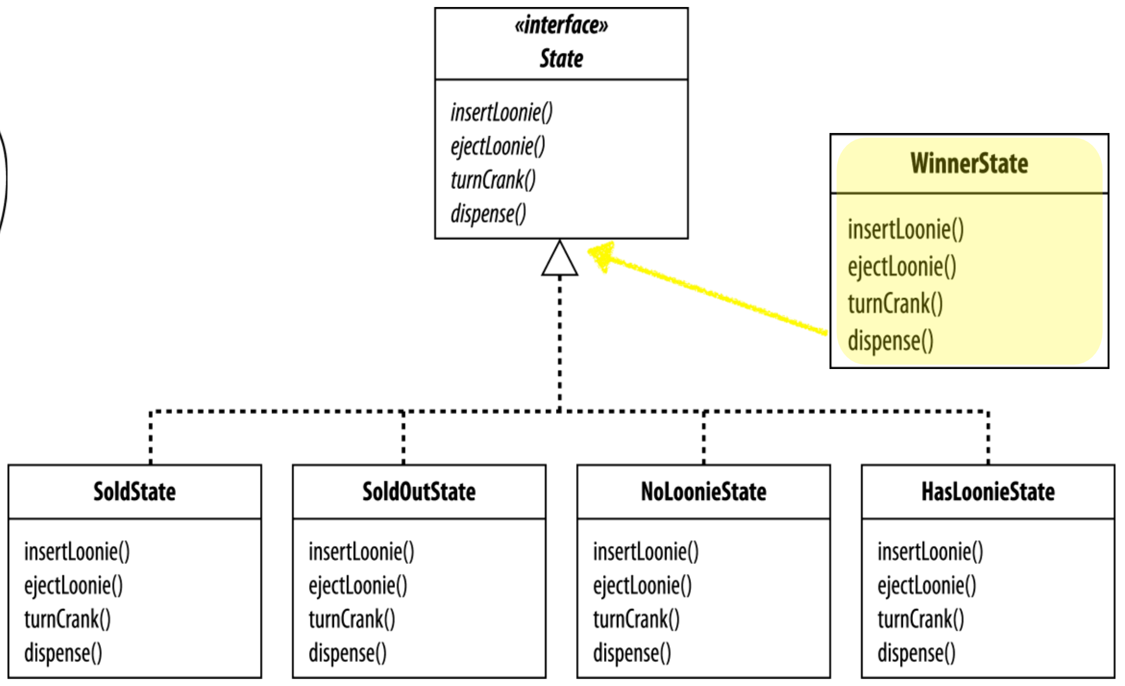
What is the difference between UML Diagrams & Design Patterns?

University of Victoria

What is the difference between a State Diagram & State Pattern?

Difference between State Diagram & State Pattern





The Factory Method Pattern

University of Victoria

Intent

- “Define an interface for creating an object, but let subclasses decide which class to instantiate”
  - It lets a class defer instantiation to subclasses at run time.
  - It refers to the newly created object through a common interface.

Also Known as

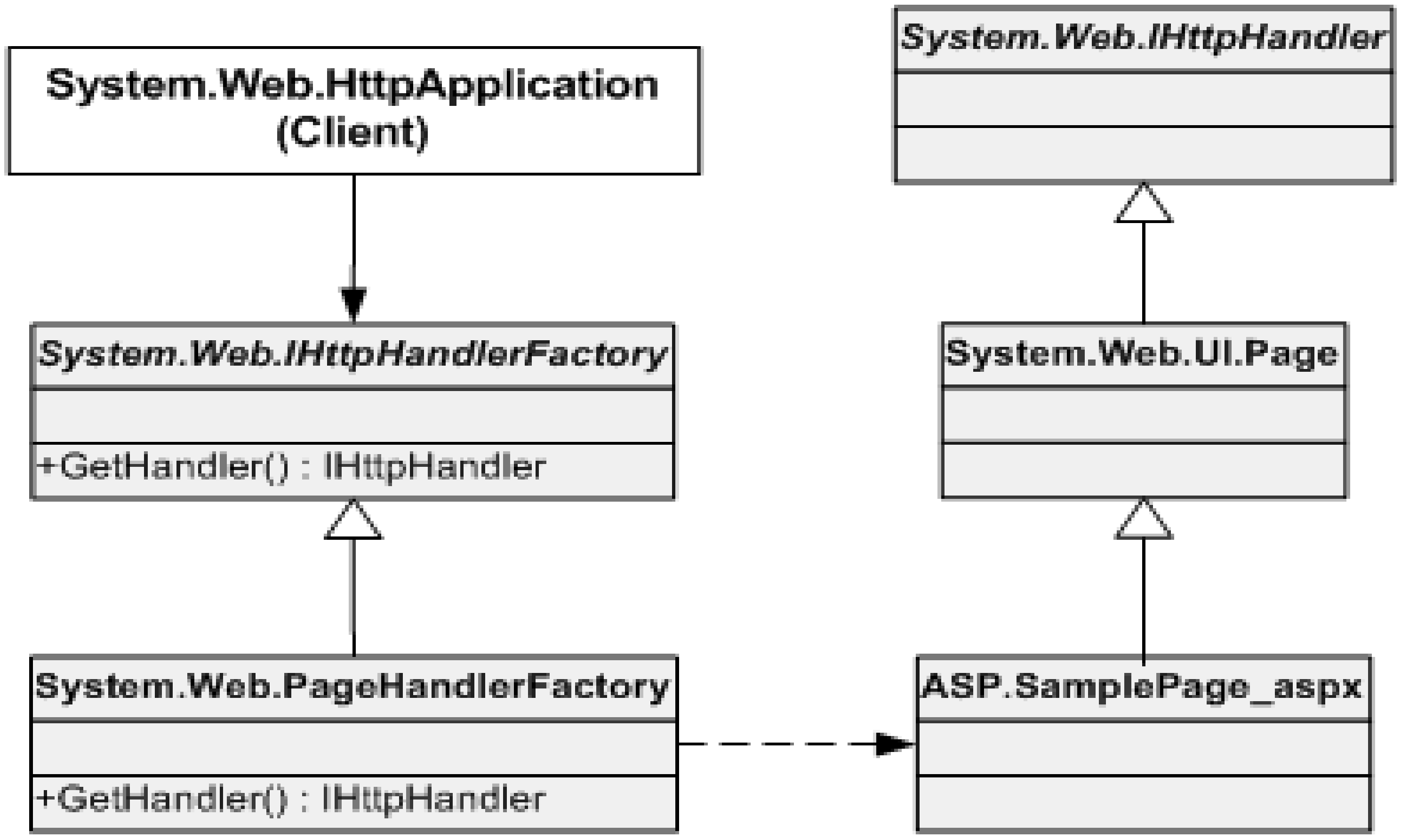
- ❑ Virtual Constructor
- ➢ The main intent of the virtual constructor idiom in C++ is to create a copy of an object or a new object without knowing its concrete type and this is exactly what the Factory Method does.

University of Victoria

Motivation

- Frameworks:
  - Factory Method is used in frameworks where library code needs to create objects of types which may be sub classed by applications using the framework.
  - Since the library knows when an object needs to be created, but not what kind of object it should create, this being specific to the application, it can use the Factory Method.

Motivating Examples – Cont.


System.Web.HttpApplication
(Client)
System.Web.IHttpHandlerFactory
+GetHandler() : IHttpHandler
System.Web.PageHandlerFactory
+GetHandler() : IHttpHandler
System.Web.IHttpHandler
System.Web.UI.Page
ASP.SamplePage_aspx

Forces

❑We want to have a set of reusable classes which are flexible enough to be extended.

❑The client does not know the type of object that needs to be created in advance and still wants to perform operations on them.

Applicability

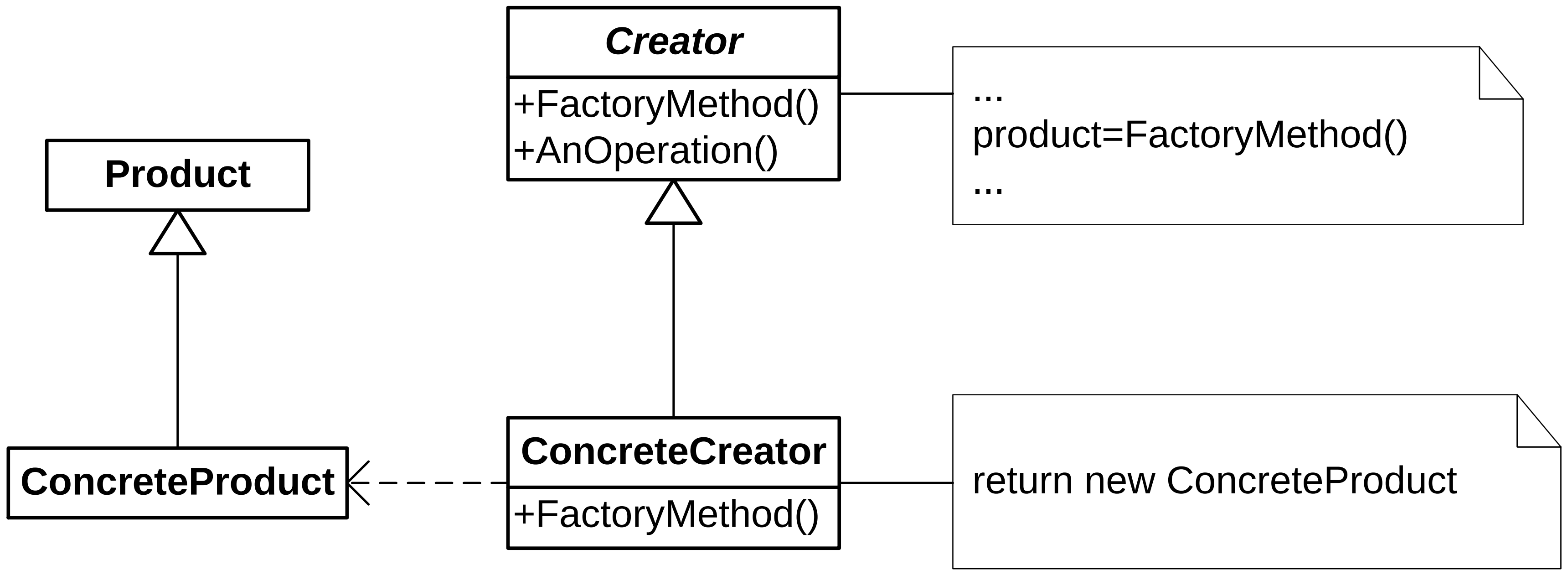
❑ Factory Method is needed when:

➢ A class can’t anticipate the types of objects it must create.

➢ A class wants its subclasses to specify the object to create.

➢ The designer wants to localize knowledge of helper sub classes.

Basic Structure



Participants

- Product (IHttpHandler)
  - Defines the interface of objects the factory method creates.
- ConcreteProduct (ASP.SamplePage_aspx)
  - Implements the Product Interface
- Creator (IHttpHandlerFactory)
  - Declares the factory method and may provide a default implementation for it.
  - Defines the return type as Product.
- ConcreteCreator (PageHandlerFactory)
  - Overrides the factory method to return an instance of ConcreteProduct.

Collaborators

- The Creator relies on the subclass's factory method to return an instance of appropriate ConcreteProduct object.

- The Creator executes some sequence of operations on the object or simply returns a reference to Product (bound to the ConcreteProduct object) to the client.

Implementation example

```
Sample sample=new Sample();

Sample mysample=new MySample();

Sample hissample=new HisSample();
```

```
Public class Factory
  {
Public static Sample creator(int  which)
   {
       if (which==1)
         return new SampleA();
        else if (which==2)
          return new SampleB();
     }
  }


Sample sampleA=Factory.creator(1);
```

Known Uses

- It is a pervasive pattern.
- It is used in several places in the Java API. For example, URLConnection has a method getContent that returns the content as an appropriate object (html, gif etc.)
- .Net Framework Class Library
  Factory method is used in:
  - Systems.Collections.IEnumerable,
  - System.Net.WebRequest
  - System.Security.Cryptography

Related Patterns

- Abstract Factory
- Template Methods
- Prototypes

The Adapter Pattern

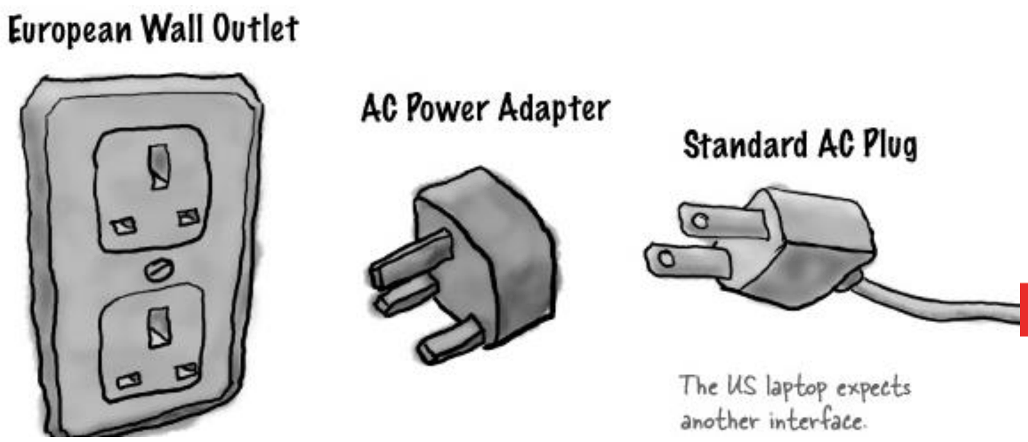
University of Victoria

Adapter Pattern

Gang of Four state the intent of Adapter is to

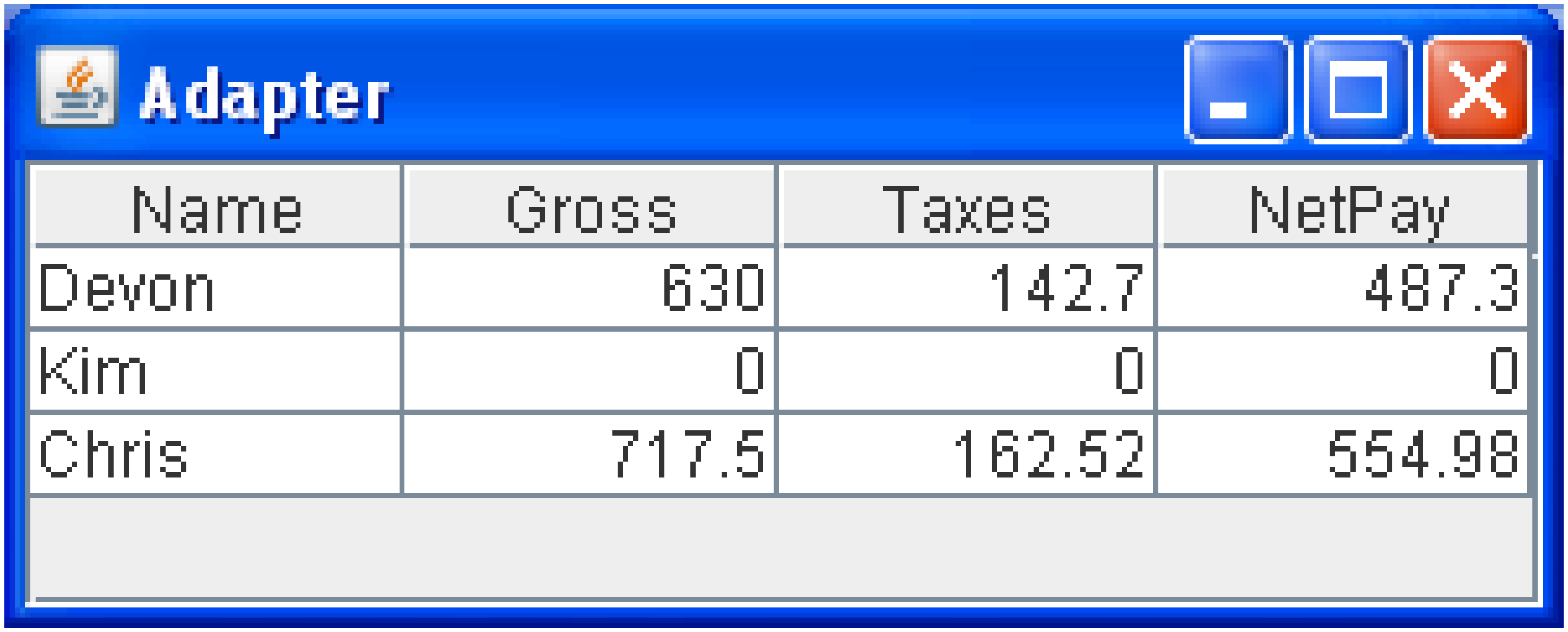
> *Convert the interface of a class into another interface that the clients expect. Adapter lets classes work together that could not otherwise because of incompatible interfaces.*

Use it when you need a way to *create a new interface for an object that does the right stuff but has the wrong interface*
"Alan Shalloway"



Adapt my collection to look like TableModel

JTable shows a list of Employees like this



Name	Gross	Taxes	NetPay
Devon	630	142.7	487.3
Kim	0	0	0
Chris	717.5	162.52	554.98

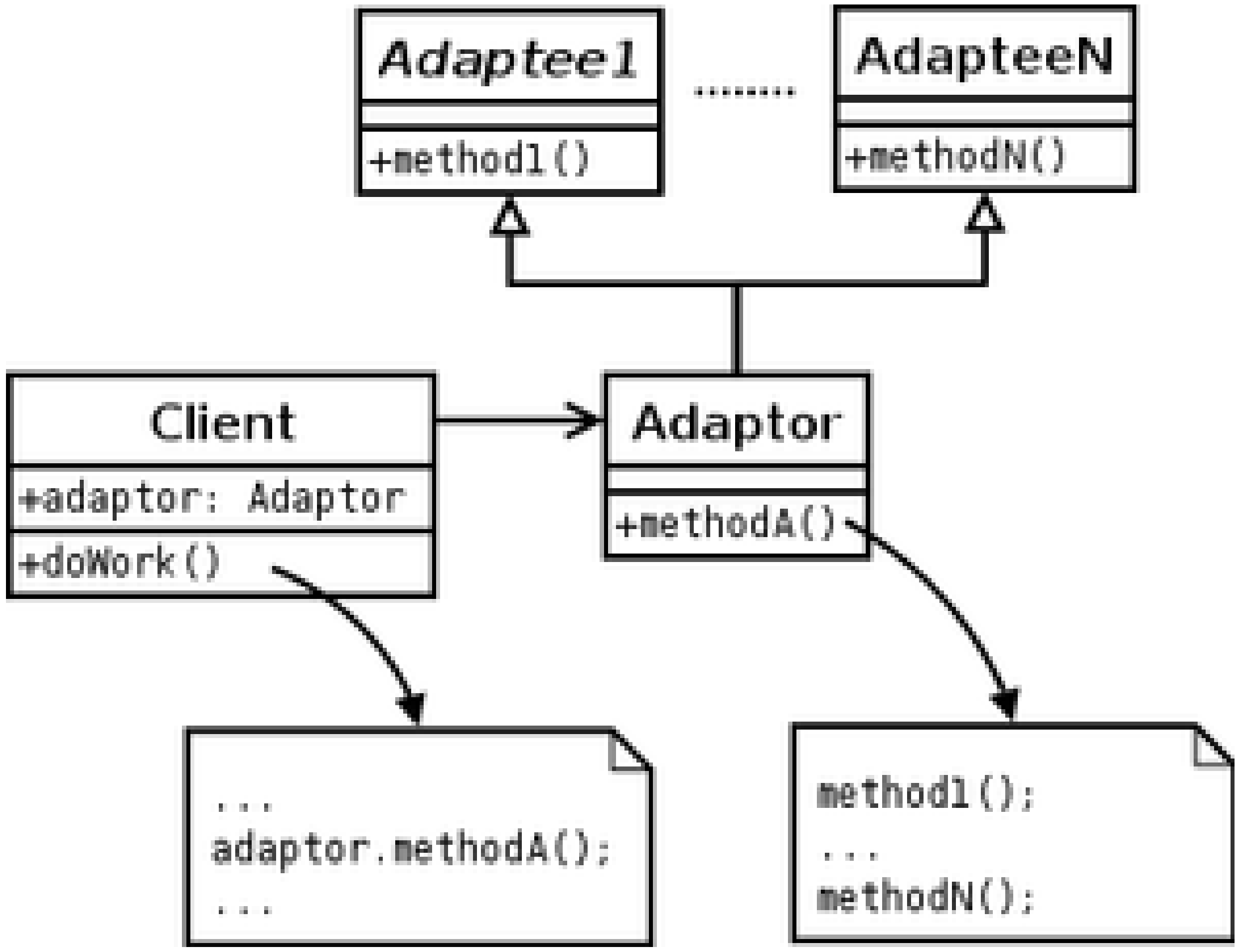
University of Victoria

EmployeeList adapted to TableModel

```java
public class EmployeeList implements TableModel {

private ArrayList<Employee> data =
             new ArrayList<Employee>();

  public EmployeeList() {
    data.add(new Employee("Devon", 40, 15.75, 3, "M"));
    data.add(new Employee("Kim", 0, 12.50, 1, "S"));
    data.add(new Employee("Chris", 35, 20.50, 2, "M"));
  }

  public void add(Employee employee) {
    data.add(employee);
  }

  public Iterator<Employee> iterator() {
    return data.iterator();
  }
}
```

University of Victoria

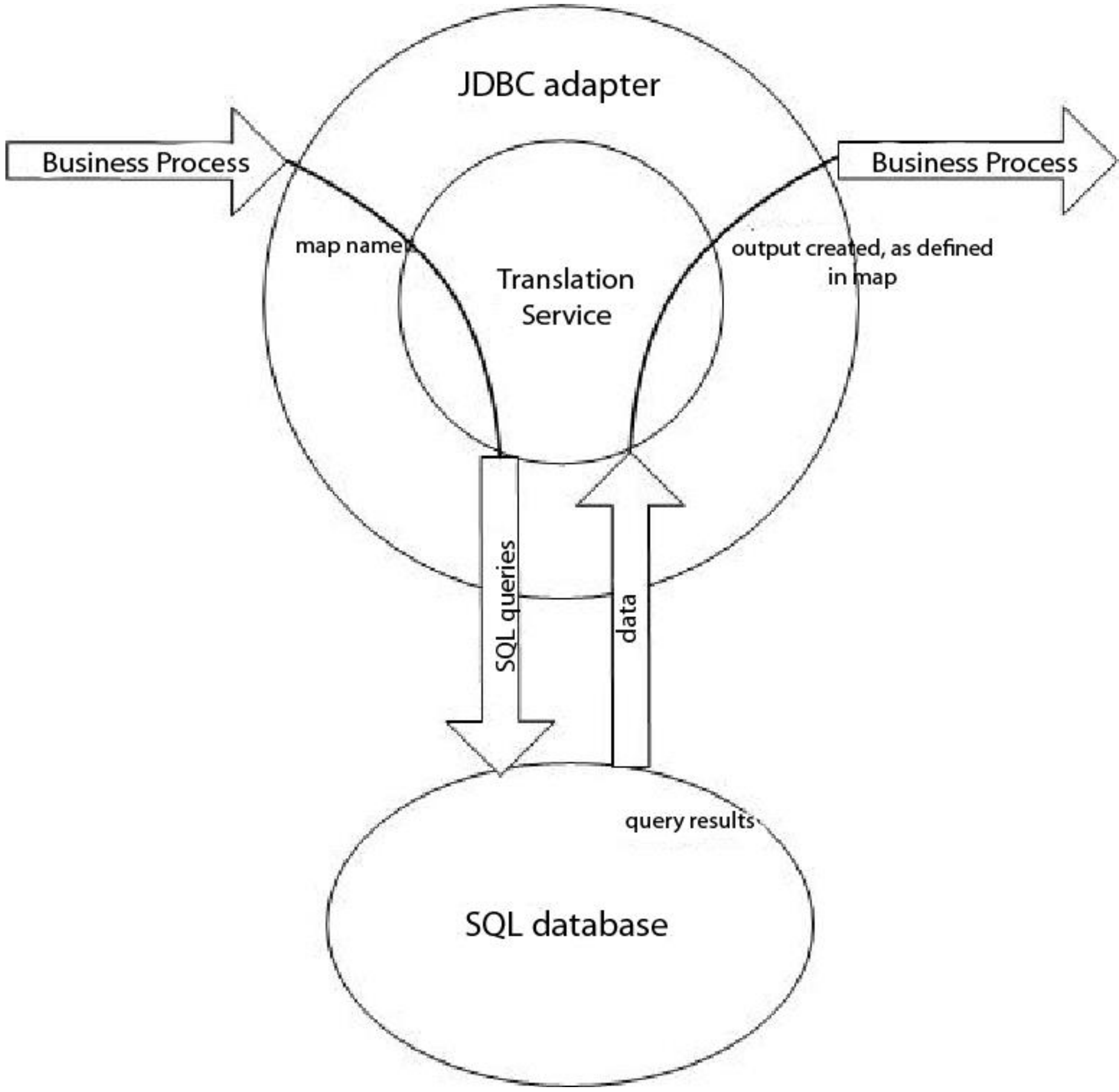
Adapter Design Pattern



Java Data Base Connectivity (JDBC) Adaptor

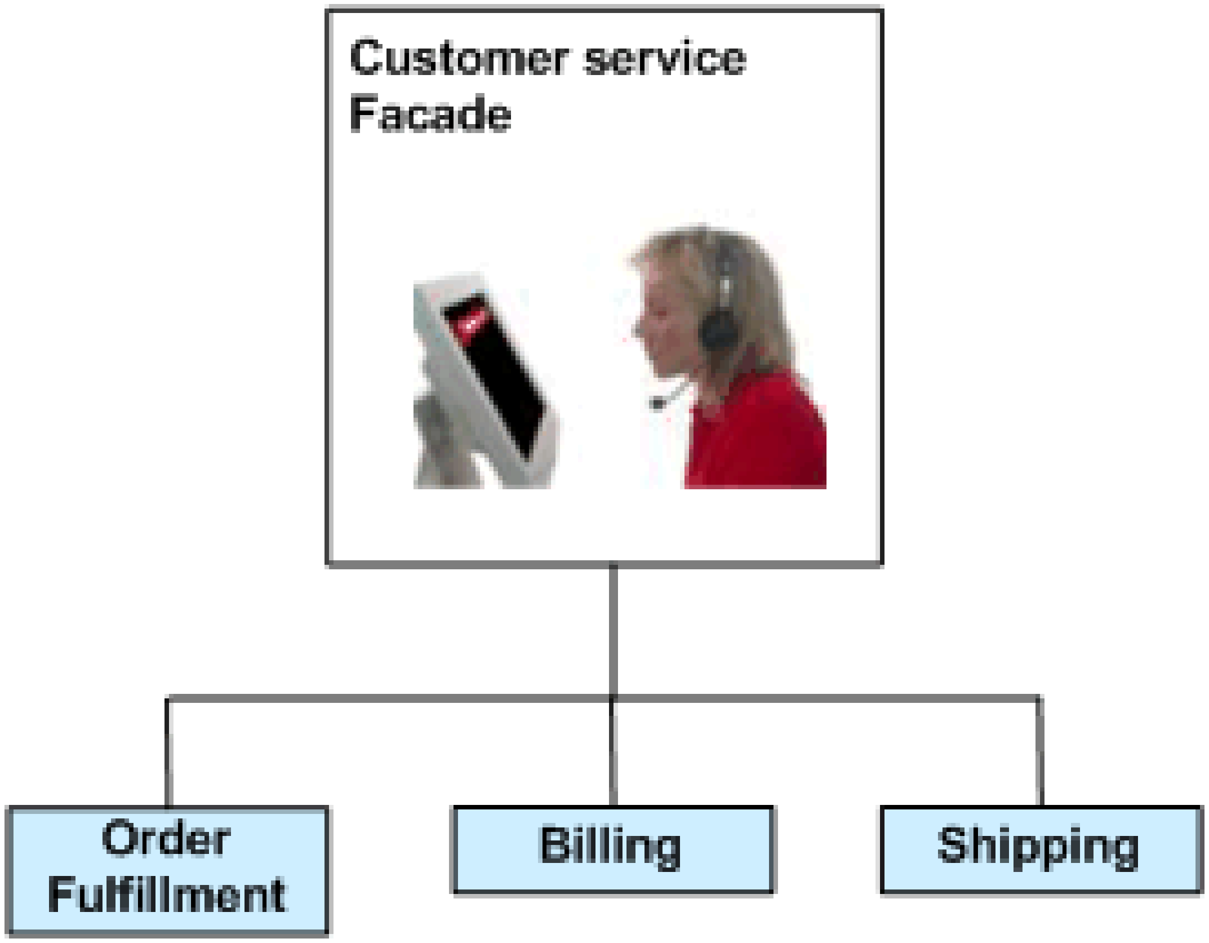
Java uses the methods of the JDBC Adaptor
The Adaptor creates SQL commands for you

Picture from IBM



The Façade Pattern

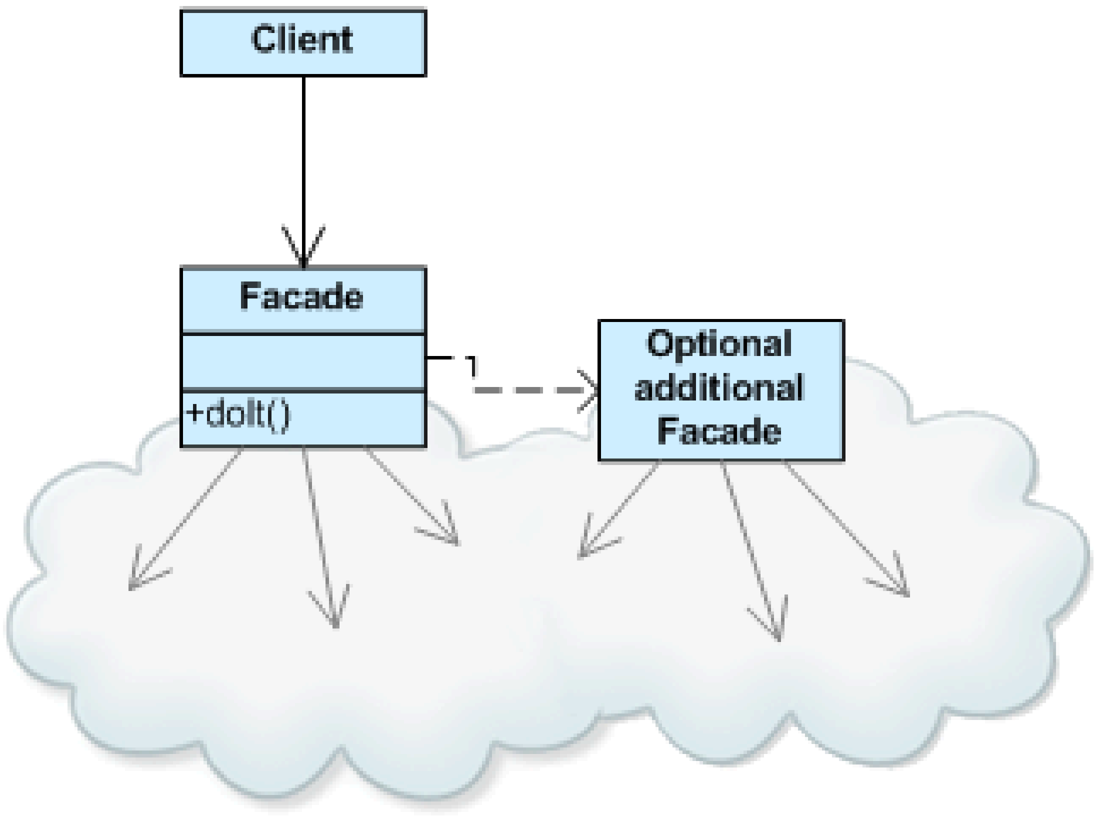
Façade Design Pattern



Façade is closely related to Adapter

- Provide a unified interface to a set of interfaces in a System. Façade defines a higher-level interface that makes the subsystem easier to use *GangOf4*

Facade takes a "riddle wrapped in an enigma shrouded in mystery", and interjects a wrapper that tames the amorphous and inscrutable mass of software.
SourceMaking



Façade Design Pattern

Façade is used to

- Create a simpler interface
- Reduce the number of objects that a client deals with
- Hide or encapsulate a large system

The Template Method Pattern

University of Victoria

What is Template?

Template is actually just a method with some steps in sequence

```
abstract ParentClass  {
        final void TemplateMethod() {
                MethodStep1()
                MethodStep2()
                MethodStep3()
                MethodStep4()
            }

void MethodStep1(){…}
void MethodStep2(){…}
….
}
```

subclass can modify the steps but cannot modify order, flow of control

An abstract class defines various methods and has one non-overridden(Final) method which calls the various other methods.

Definition (GOF)

- The Template Method is know as a behavioral pattern which lets subclasses implement behaviour that can vary
- Define the Skeleton of an algorithm in operation, deferring some steps to subclass
- Template Method lets subclasses redefine certain steps of an algorithm without changing, The algorithm’s structure.

Problem

- ❑ When a majority of problem domain classes are all similar except a few classes that have deviant behaviour
- ❑ Template method uses Inheritance to solve problem

How it might help the designer?

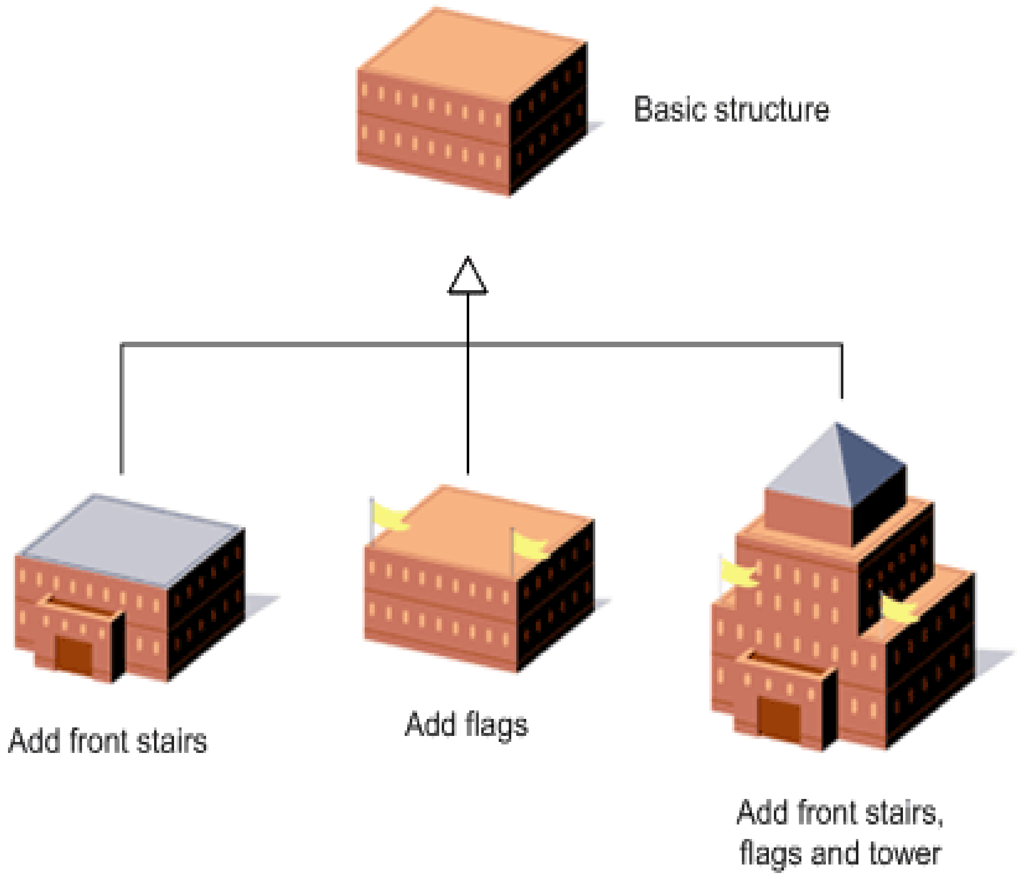
- Template Method allows the designer and the developer to encapsulate the basic, common behavior in a base class and defer the implementation of the deviant behavior to a derived class.

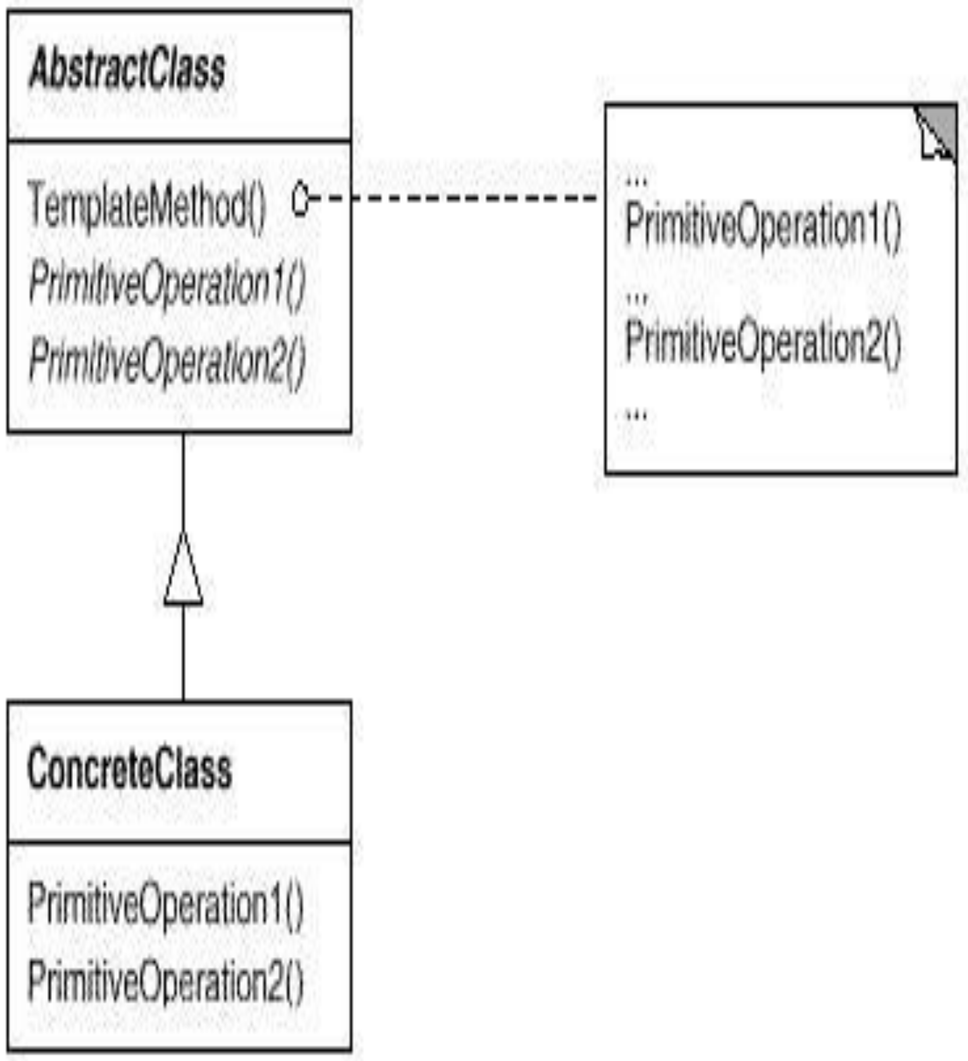
Significance

- ❑ It deals with the assignment of responsibilities between classes.
- ❑ The idea is to distribute responsibility between the members of a family using Inheritance.
- ❑ Eliminate Duplicate code
- ❑ Lets subclasses implement behaviour that can vary
- ❑ Controls at what point(s) subclassing is allowed

University of Victoria

Basic Structure





Abstract class define the template and concrete class manages the implementation

In-class activity (Groups of 2)

- Implement State Pattern with the “example” you’ve provided in the last class.
- Submit Screenshots of the code and output to Week 6
- 20 minutes

University of Victoria