河北師范大学软件学院
Software College of Hebei Normal University

第三章 适配器模式

任课教师：武永亮
wuyongliang@eud2act.org

上节回顾

- 当被观察者发生改变的时候，观察者就会观察到这样的变化，并且做出相应的响应
- 实现观察者模式有很多形式，比较直观的一种是使用一种“注册——通知——撤销注册”的形式。
- 减少观察者和被观察对象之间的耦合。

课程内容

- 环境及问题
- 适配器模式详解
- 适配器模式实现
- 扩展练习

课程内容

- 环境及问题
- 适配器模式详解
- 适配器模式实现
- 扩展练习

环境

- 游戏中的坐骑——五彩神鹿
- 第一世界它的行走方式为奔跑，第二世界它的行走方式为飞！

环境

```
class Deer
{
    public void run()
    {
        Console.WriteLine("我是一只五彩神鹿，带你游走四处。");
    }
}
```

But, I want Fly!!!

请思考10分钟如何让这只鹿飞起来？？
别忘了我们的原则——修改封闭

问题

- 想使用一个已经存在的类，
- 但他的接口不符合需求。

适配器模式（Adapter）

- 将一个类的接口转换成客户希望的另外一个接口
- 使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。

课程内容

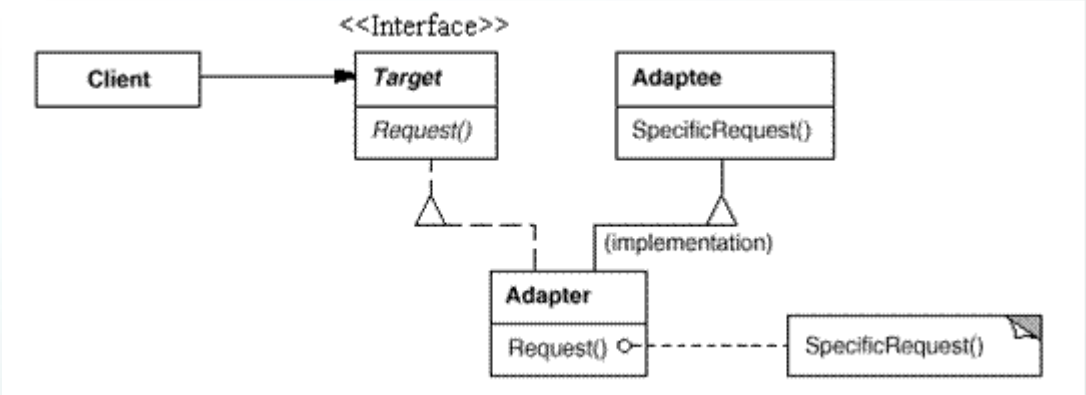
- 环境及问题
- 适配器模式详解
- 适配器模式实现
- 扩展练习

适配器模式（ Adapter Pattern ）

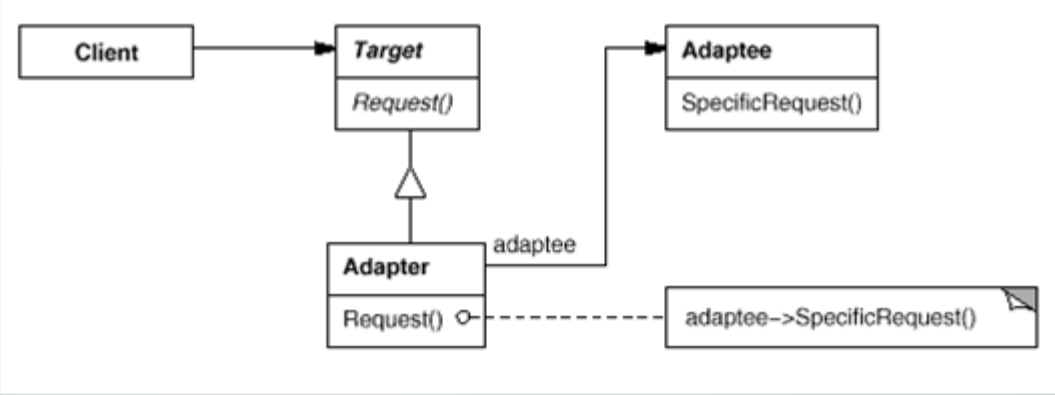
- 适配器模式中有以下的四种角色。
  - 目标(target)：定义客户端使用的与特定领域相关的接口。
  - 被适配者(adaptee)：定义了一个已经存在的接口，这个接口需要匹配。
  - 适配者(adapter)：对Adaptee的接口与target的接口进行适配。
  - 客户端(Client)：与符合target接口的对象协同

适配器模式分类

- 类的适配器模式（采用继承实现）



- 对象适配器（采用对象组合方式实现）



课程内容

- 环境及问题
- 适配器模式详解
- 适配器模式实现
- 扩展练习

适配器模式实现步骤

- 类适配器
  - 确定目标接口
  - 确定被适配者
  - 创建适配器（继承自被适配者，实现目标接口）
- 对象适配器
  - 确定目标接口
  - 确定被适配者
  - 创建适配器（拥有被适配者的对象，实现目标接口）

适配器模式实现步骤一

- 被确定目标接口。

```
interface ITarget
{
    public void run();
    public void fly();
}
```

适配器模式实现步骤二

- 被确定被适配者。

```
class Deer
{
    public void run()
    {
        Console.WriteLine("我是一只五彩神鹿，带你游走四处。");
    }
}
```

适配器模式实现步骤三

- 创建适配器（类适配器）。

```
class classAdapter : Deer,ITarget
{
    public void fly()
    {
        Console.WriteLine("哇啊哦，我可以飞了！！");
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        ITarget flyDeer = new classAdapter();
        flyDeer.run();
        flyDeer.fly();
    }
}
```

思考对象适配器如何实现？

适配器模式实现步骤三

- 创建适配器（对象适配器）。

```
class objectAdapter:ITarget
{
    private Deer deer;

    public objectAdapter(Deer d)
    {
        this.deer = d;
    }

    public void run()
    {
        deer.run();
    }

    public void fly()
    {
        Console.WriteLine("哇啊哦，我一样可以飞！！");
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        ITarget flyDeer = new objectAdapter(new Deer());
        flyDeer.run();
        flyDeer.fly();
    }
}
```

思考：类适配器和对象适配器哪个更好

- 类适配器采用“多继承”的实现方式，带来了不良的高耦合
- 对象适配器采用“对象组合”的方式，更符合松耦合精神
- 类适配器无法面对多个被适配对象。

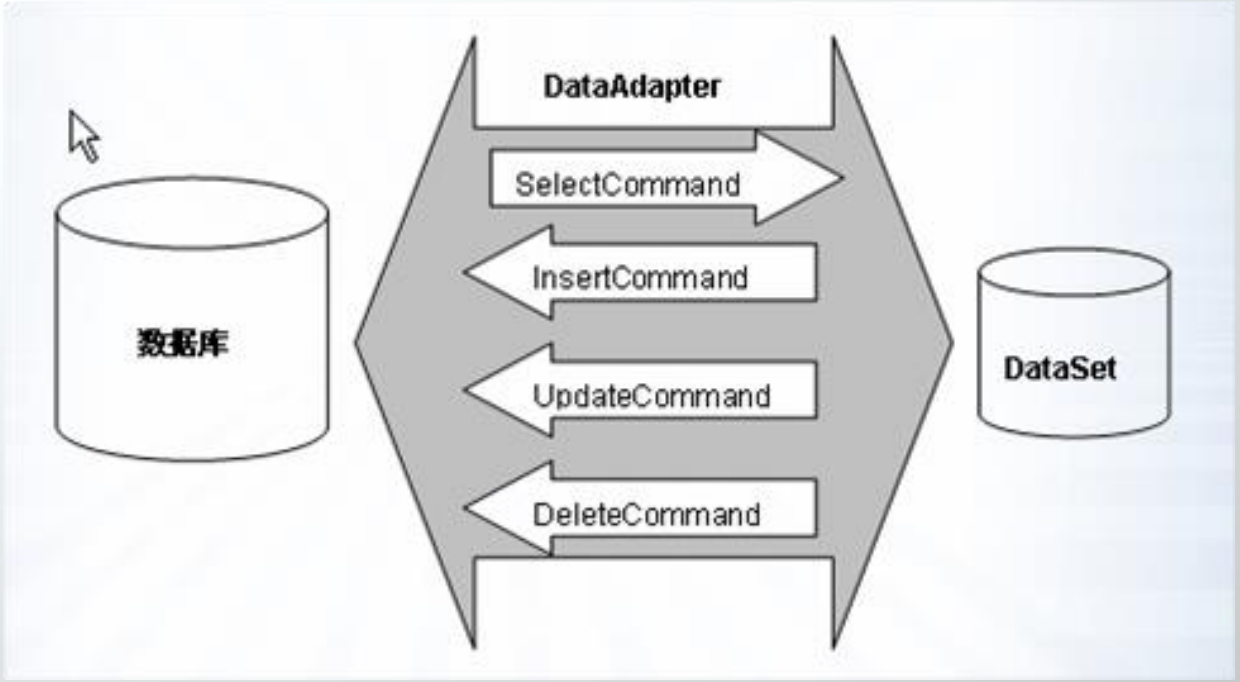
合成复用原则！

课程内容

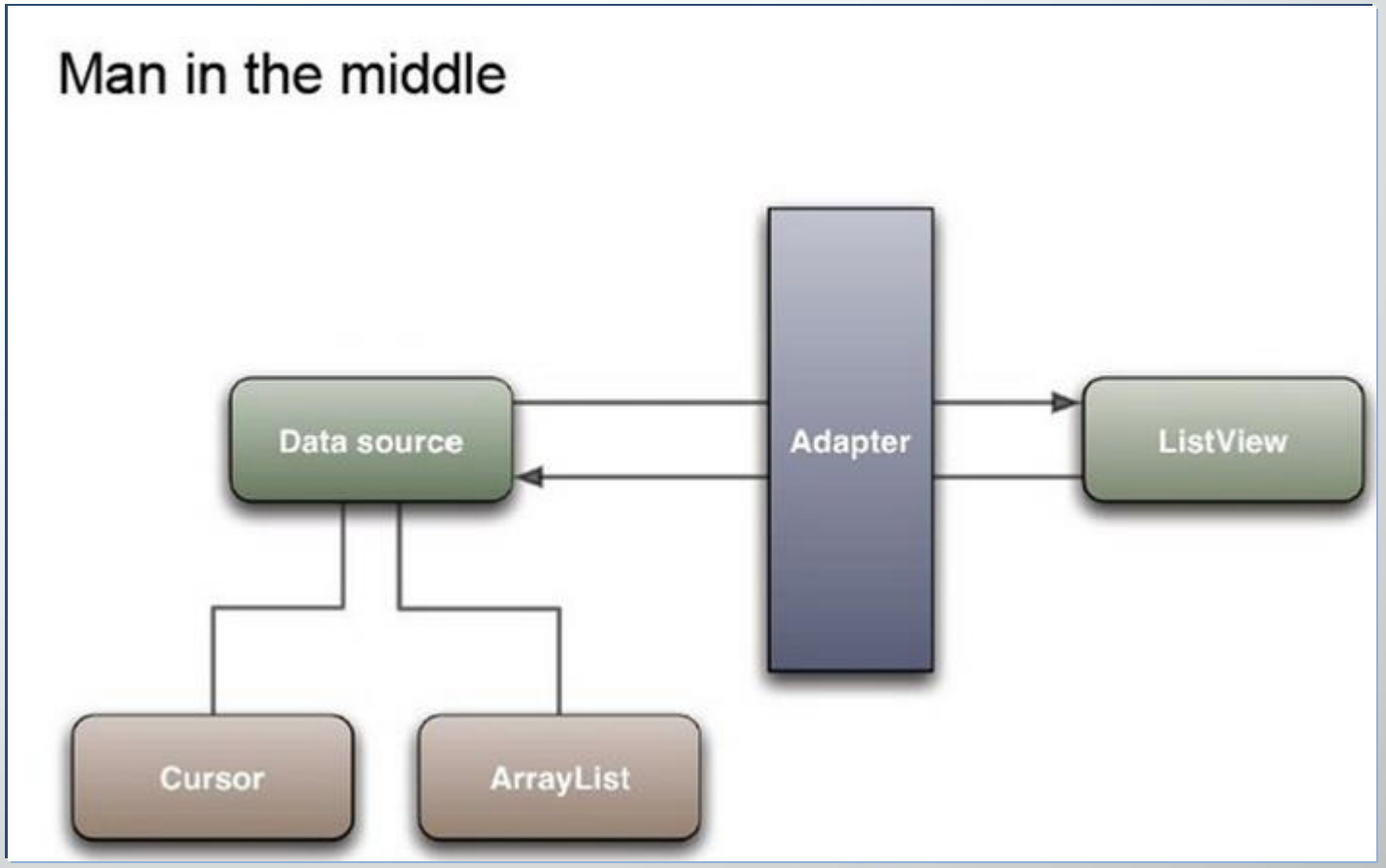
- 环境及问题
- 适配器模式详解
- 适配器模式实现
- 扩展练习

案例练习

扩展说明


DataAdapter
SelectCommand
InsertCommand
UpdateCommand
DeleteCommand
数据库
DataSet

扩展说明


Man in the middle
Data source
Adapter
ListView
Cursor
ArrayList

小结

- 将一个类的接口转换成客户希望的另外一个接口
- 使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。
- 尽可能保持已有的类不变的前提下，适应当前的系统。

Thank You，谢谢！