ThoughtWorks®

OO Training

CLASSIFIER & MARS-ROVER

周颖　(ybzhou@thoughtworks.com)
郑培真（pzzheng@thoughtworks.com)

CLASSIFIER

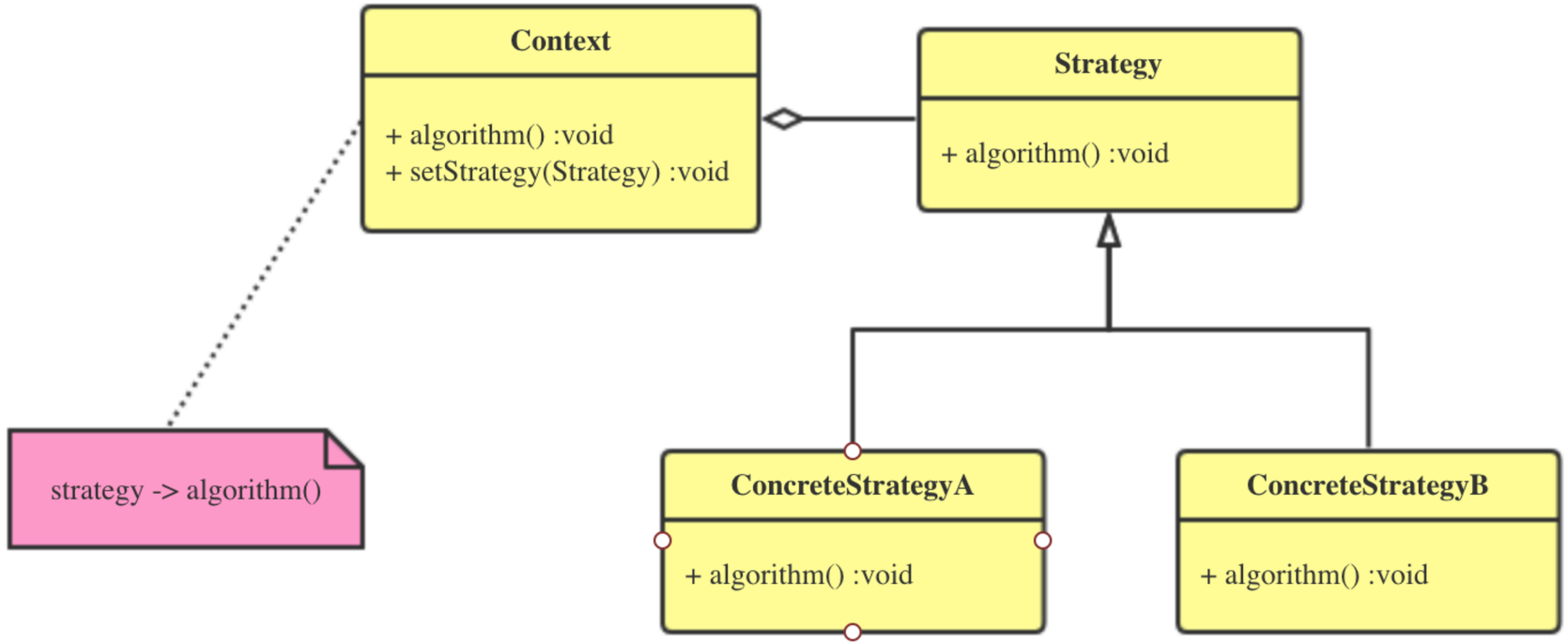
- 给定一个整数列表
- 输出正整数个数
- 输出负整数个数
- 输出偶数个数
- 输出奇数个数

HARD CODING

- 示例：提供多种过滤算法
- 方法一
  - 将提供的算法写到同一个类中
  - 该类提供多个方法，每一个方法对应一个具体的查找算法
- 方法二
  - 将查找算法封装在一个统一的方法中
  - 通过 if…else… 等判断条件进行选择
- 缺陷
  - 增加一种新的查找算法，需要修改封装算法类的源代码
  - 更换查找算法，需要修改客户端的调用代码
  - 算法类比较复杂，维护困难

STRATEGY PATTERN

- 定义一系列算法，将每一个算法封装起来，并让他们可以互相替换
- 策略模式让算法独立于使用它的客户而变化
- 包含Context、Strategy、ConcreteStrategy



CODE ANALYSIS

```
//Strategy Interface
public interface CompressionStrategy {
  public void compressFiles(ArrayList<File> files);
}
```

```
public class ZipCompressionStrategy implements CompressionStrategy {
  public void compressFiles(ArrayList<File> files) {
    //using ZIP approach
  }
}
```

```
public class RarCompressionStrategy implements CompressionStrategy {
  public void compressFiles(ArrayList<File> files) {
    //using RAR approach
  }
}
```

```
public class Client {
  public static void main(String[] args) {
    CompressionContext ctx = new CompressionContext();
    //we could assume context is already set by preferences
    ctx.setCompressionStrategy(new ZipCompressionStrategy());
    //get a list of files...
    ctx.createArchive(fileList);
  }
}
```

```
public class CompressionContext {
  private CompressionStrategy strategy;
  //this can be set at runtime by the application preferences
  public void setCompressionStrategy(CompressionStrategy strategy) {
    this.strategy = strategy;
  }

  //use the strategy
  public void createArchive(ArrayList<File> files) {
    strategy.compressFiles(files);
  }
}
```

BENEFITS & DRAWBACKS

- 优势
  - 完美支持“开闭原则”（Open Closed Principle）
  - 运行时修改算法
  - 避免使用多重条件转移语句
  - 提供了可以替换继承关系的办法

- 缺陷
  - 客户端必须知道所有的策略类，并自行决定使用哪一个策略类
  - 系统中产生很多策略类

APPLICABLE SCENES

- 一个系统中有许多类，区别仅在于行为
- 一个系统需要动态地在几种算法中选择一种
- 一个对象有很多的行为，如果不用恰当的模式，这些行为只能使用多重条件选择语句
- 不希望客户端知道复杂的、与算法相关的数据结构

STRATEGY PATTERN USING LAMBDA

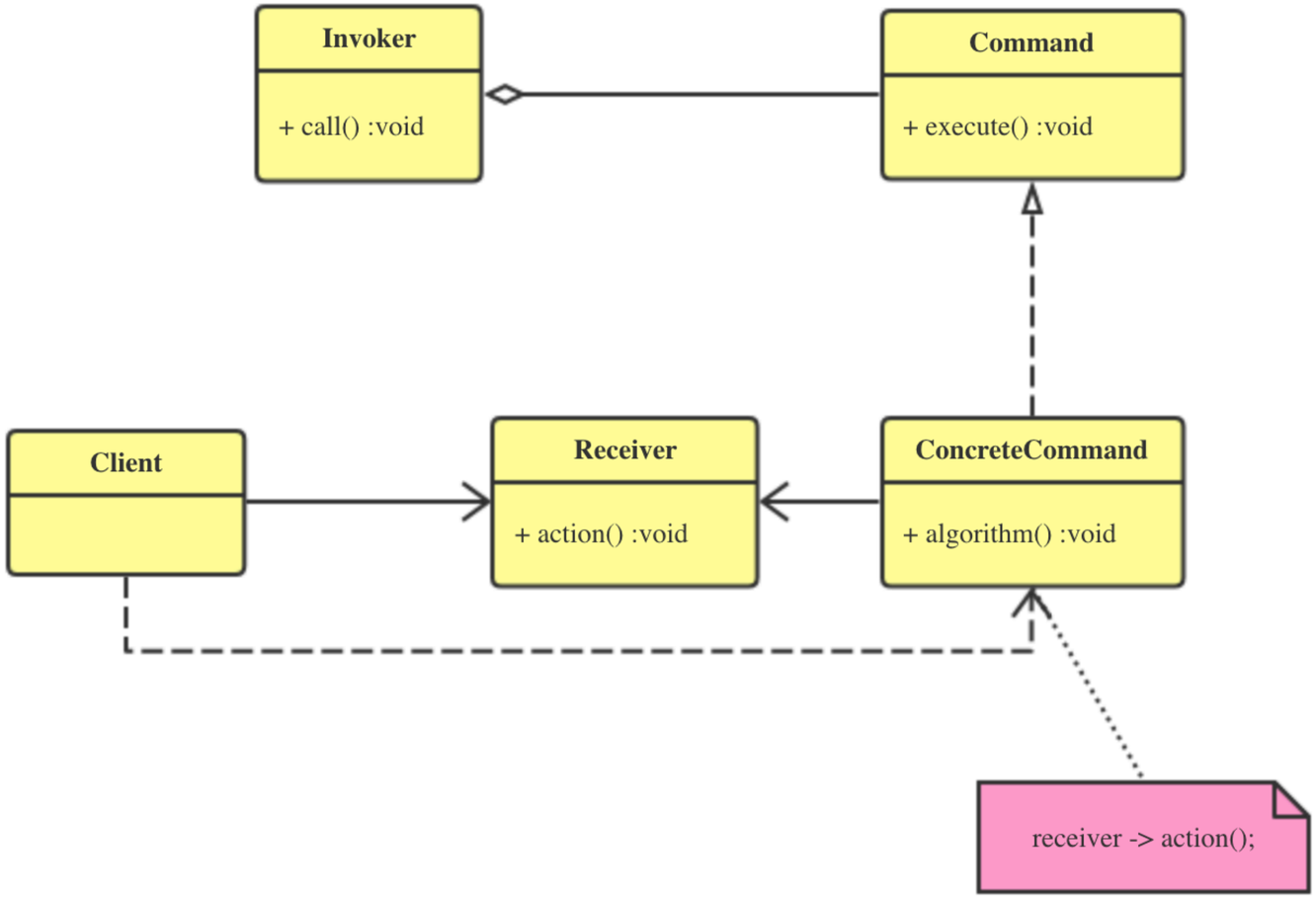
- 定义策略接口
- 可选注解：@FunctionalInterface
- 动态传入不同的函数实现

MARS ROVER

- 一个机器人探测器将由NASA送上火星高原，探测器将在这个奇特的矩形高原上行驶。用它们携带的照相机将周围的全景地势图发回到地球。每个探测器的方向和位置将由一个x,y系坐标图和一个表示地理方向的字母表示出来。为了方便导航，平原将被划分为网格状。位置坐标示例：0，0，N，表示探测器在坐标图的左下角，且面朝北方。为控制探测器，NASA会传送一串简单的字母。可能传送的字母为： 'L ', 'R '和 'M '。 'L ',和 'R '分别表示使探测器向左、向右旋转90度，但不离开他所在地点。 'M ' 表示向前开进一个网格的距离，且保持方向不变。假设以广场（高原）的直北方向为y轴的指向。
- 输入一系列的指令，输出探测器目前的位置坐标

COMMAND PATTERN

- 封装对对象的消息请求
- 运行时指定具体的请求接收者
- 包含Invoker、Command、ConcreteCommand、Receiver



CODE ANALYSIS

```java
//Command
public interface Command{
  public void execute();
}
```

```java
//Concrete Command
public class LightOnCommand implements Command{
  //reference to the light
  Light light;
  public LightOnCommand(Light light){
    this.light = light;
  }
  public void execute(){
    light.switchOn();
  }
}
```

```java
//Concrete Command
public class LightOffCommand implements Command{
  //reference to the light
  Light light;
  public LightOffCommand(Light light){
    this.light = light;
  }
  public void execute(){
    light.switchOff();
  }
}
```

```java
//Receiver
public class Light{
  private boolean on;
  public void switchOn(){
    on = true;
  }
  public void switchOff(){
    on = false;
  }
}
```

```java
//Invoker
public class RemoteControl{
  private Command command;
  public void setCommand(Command command){
    this.command = command;
  }
  public void pressButton(){
    command.execute();
  }
}
```

```java
//Client
public class Client{
  public static void main(String[] args)    {
    RemoteControl control = new RemoteControl();
    Light light = new Light();
    Command lightsOn = new LightsOnCommand(light);
    Command lightsOff = new LightsOffCommand(light);

    //switch on
    control.setCommand(lightsOn);
    control.pressButton();

    //switch off
    control.setCommand(lightsOff);
    control.pressButton();
  }
}
```

BENEFITS & DRAWBACKS

- 优势
  - 将命令发送者与命令接收者完全解耦
  - 系统很容易支持新的命令
  - 可以比较容易地设计命令队列和组合命令
  - 方便地实现对请求的Undo和Redo
- 缺陷
  - 可能会导致系统中有过多的具体命令类

APPLICABLE SCENES

- 需要将请求调用者与请求接收者解耦
- 需要异步执行请求
- 需要支持命令的撤销和恢复操作
- 需要将一组操作组合在一起

STRATEGY PATTERN V.S. COMMAND PATTERN

- 命令模式含有不同的命令，做不同的事情
- 策略模式含有不同的算法，做相同的事情

- 命令模式中有接收者
- 策略模式没有接收者

- 命令模式针对菜单上的压缩、解压等
- 策略模式针对不同的压缩和解压缩算法

ASSIGNEMTNS

- 可以正确打印 Author 的信息
- 可以正确打印 Book 的信息
- 书店 BookStore
  - BookStore 初始拥有金钱 10000 人民币
  - BookStore 可以按进价购买各种书籍以便出售
  - 30元及以下的书进价是售价的一半，超过30元的书进价是售价的40%
  - Book 上标注的 price 是售价
- BookStore 可以根据销售记录，补充库存
  - 卖多少本，补充多少本
  - 销量前三的书籍，补充卖出数量的两倍，其他书籍补充卖出的数量
  - 销量第一的书籍，补充卖出数量的两倍，销量最后一名的不补充，其他补充卖出的数量（只有一种书籍，按销量第一处理）
- BookStore 可以输出目前的销售额和利润

ASSIGNEMTNS

- 完善Mars Rover
- 限制活动范围，例如给定右上角坐标为（x, y）
- 熟悉策略模式、命令模式
- 我们的项目代码里面是否有地方可以重构

ThoughtWorks®

QUESTIONS?