设计模式实验

刘恒星 2022229044

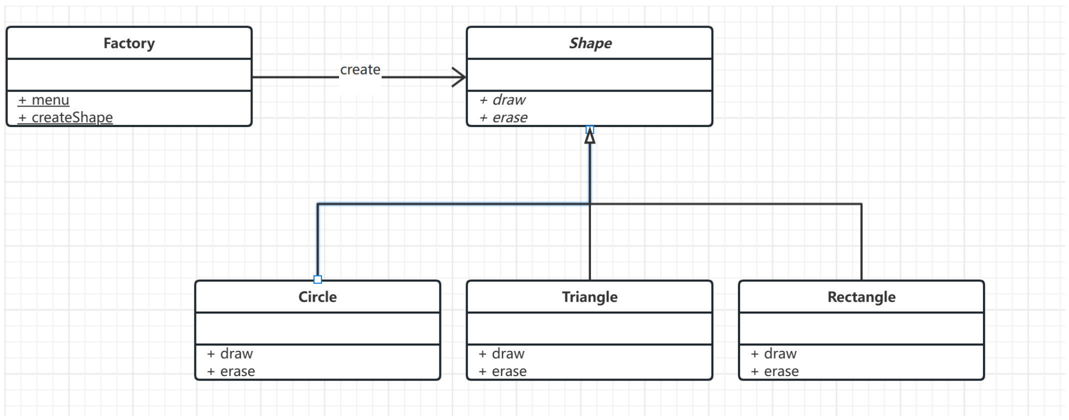
March 27, 2023

1 简单工厂模式

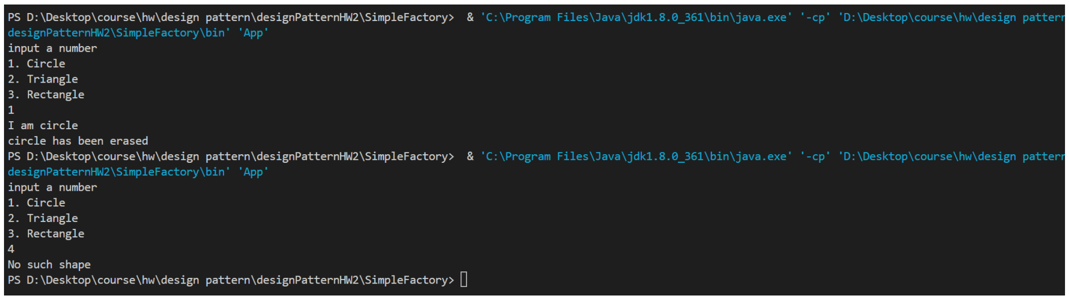
实验要求

简单工厂模式使用简单工厂模式设计一个可以创建不同几何形状（Shape）（例如圆形（Circle）、矩形（Rectangle）和三角形（Triangle）等）的绘图工具类，每个几何图形均具有绘制方法 draw() 和擦除方法 erase()，要求在绘制不支持的几何图形时，抛出一个 UnsupportedShapeException 异常。绘制类图并编程模拟实现

类图设计



代码运行效果



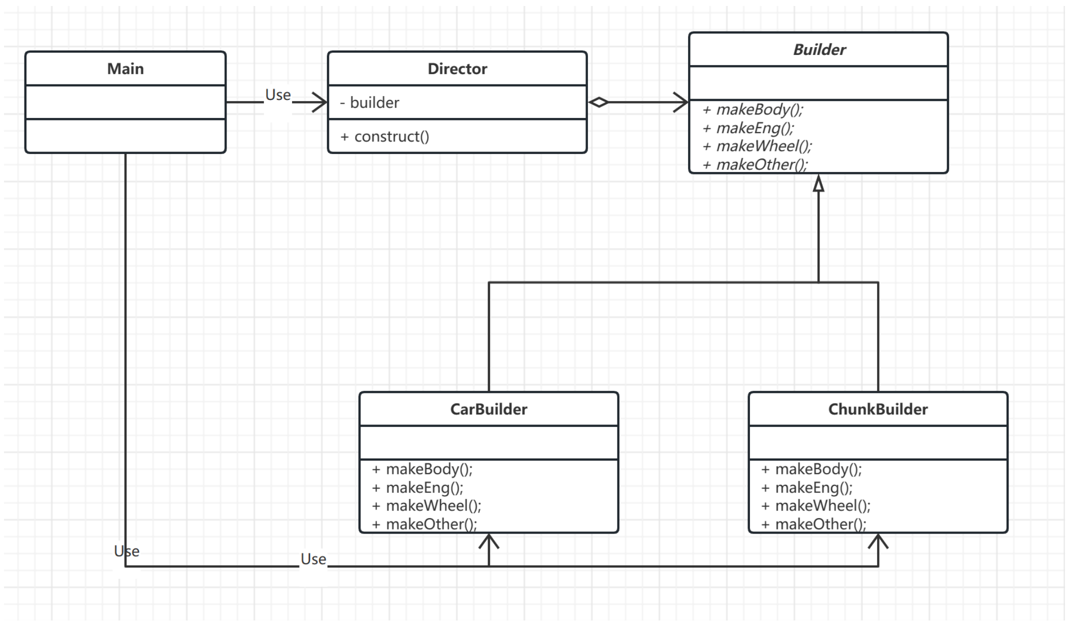
```
PS D:\Desktop\course\hw\design pattern\designPatternHW2\SimpleFactory>  & 'C:\Program Files\Java\jdk1.8.0_361\bin\java.exe' '-cp' 'D:\Desktop\course\hw\design pattern
designPatternHW2\SimpleFactory\bin' 'App'
input a number
1. Circle
2. Triangle
3. Rectangle
1
I am circle
circle has been erased
PS D:\Desktop\course\hw\design pattern\designPatternHW2\SimpleFactory>  & 'C:\Program Files\Java\jdk1.8.0_361\bin\java.exe' '-cp' 'D:\Desktop\course\hw\design pattern
designPatternHW2\SimpleFactory\bin' 'App'
input a number
1. Circle
2. Triangle
3. Rectangle
4
No such shape
PS D:\Desktop\course\hw\design pattern\designPatternHW2\SimpleFactory> 
```

2　建造者模式

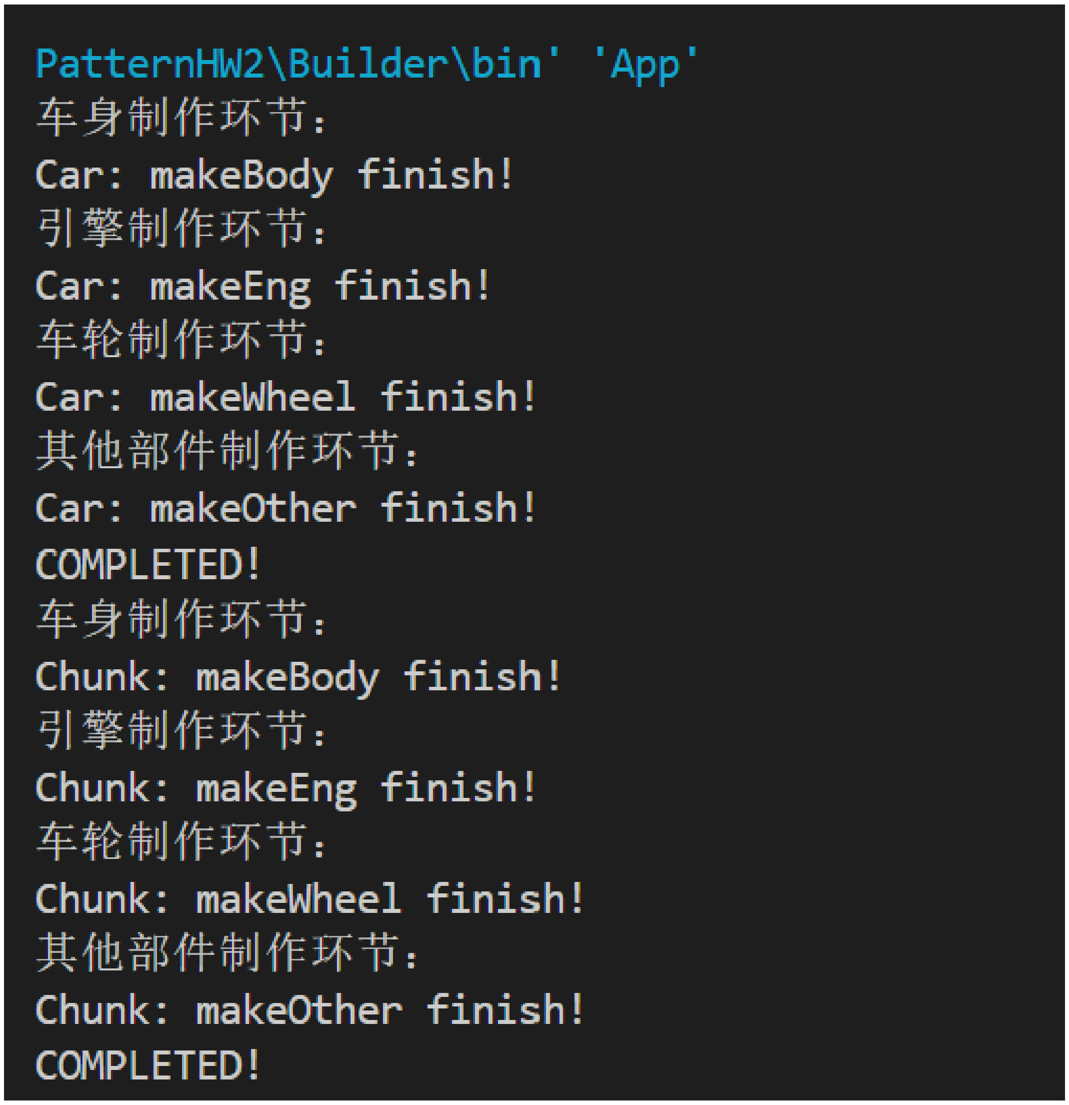
实验要求

在某赛车游戏中，赛车包括方程式赛车、场地越野赛车、运动汽车、卡车等类型，不同类型的赛车的车身、发动机、轮胎、变速箱等部件有所区别。玩家可以自行选择赛车类型，系统将根据玩家的选择创建出一辆完整的赛车。现采用建造者模式实现赛车的构建，绘制对应的类图并编程模拟实现。

类图设计



代码运行效果



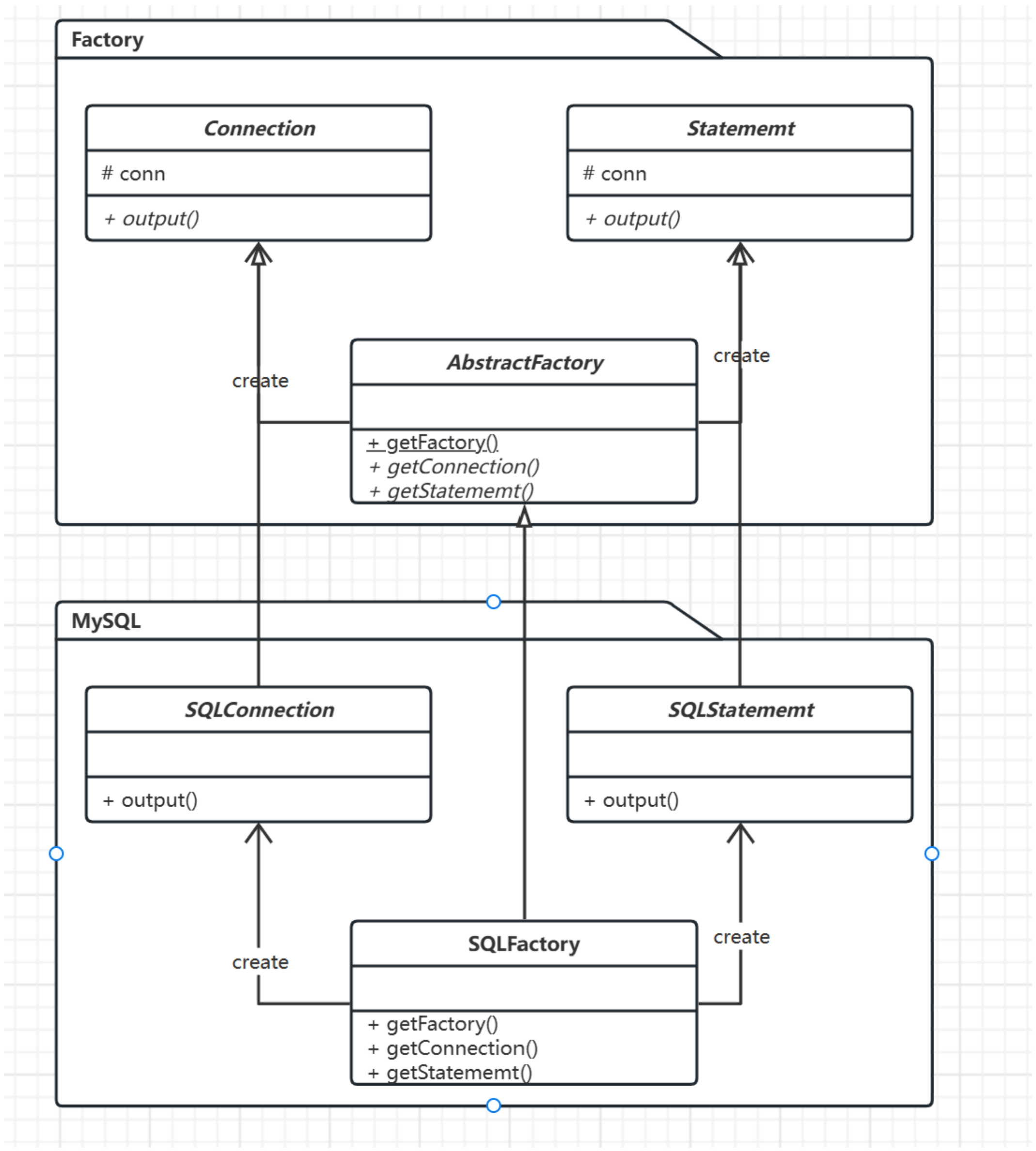
```
PatternHW2\Builder\bin' 'App'
车身制作环节:
Car: makeBody finish!
引擎制作环节:
Car: makeEng finish!
车轮制作环节:
Car: makeWheel finish!
其他部件制作环节:
Car: makeOther finish!
COMPLETED!
车身制作环节:
Chunk: makeBody finish!
引擎制作环节:
Chunk: makeEng finish!
车轮制作环节:
Chunk: makeWheel finish!
其他部件制作环节:
Chunk: makeOther finish!
COMPLETED!
```

3 抽象工厂模式

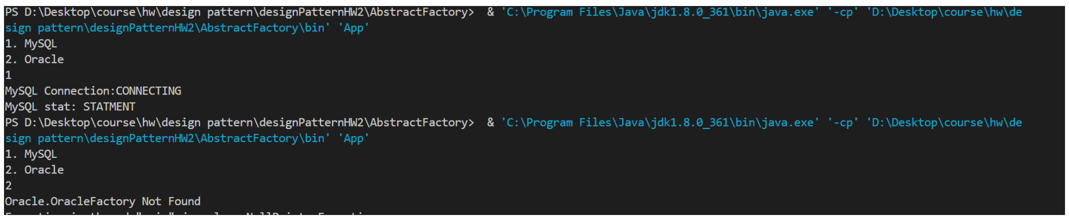
实验要求

某系统为了改进数据库操作的性能，用户可以自定义数据库连接对象 Connection 和语句对象 Statement，针对不同类型的数据库提供不同的连接对象和语句对象，例如提供 Oracle 或 MySQL 专用连接类和语句类，而且用户可以通过配置文件等方式根据实际需要动态更换系统数据库。使用抽象工厂模式设计该系统，绘制对应的类图并编程模拟实现。

类图设计



代码运行效果



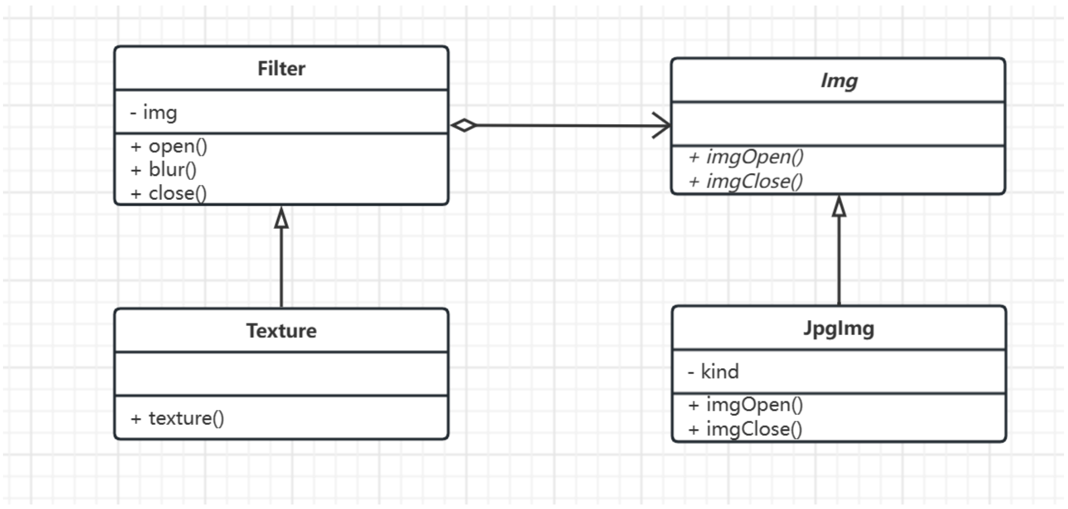
```
PS D:\Desktop\course\hw\design pattern\designPatternHW2\AbstractFactory>  & 'C:\Program Files\Java\jdk1.8.0_361\bin\java.exe' '-cp' 'D:\Desktop\course\hw\de
sign pattern\designPatternHW2\AbstractFactory\bin' 'App'
1. MySQL
2. Oracle
1
MySQL Connection:CONNECTING
MySQL stat: STATMENT
PS D:\Desktop\course\hw\design pattern\designPatternHW2\AbstractFactory>  & 'C:\Program Files\Java\jdk1.8.0_361\bin\java.exe' '-cp' 'D:\Desktop\course\hw\de
sign pattern\designPatternHW2\AbstractFactory\bin' 'App'
1. MySQL
2. Oracle
2
Oracle.OracleFactory Not Found
```

4 桥接模式

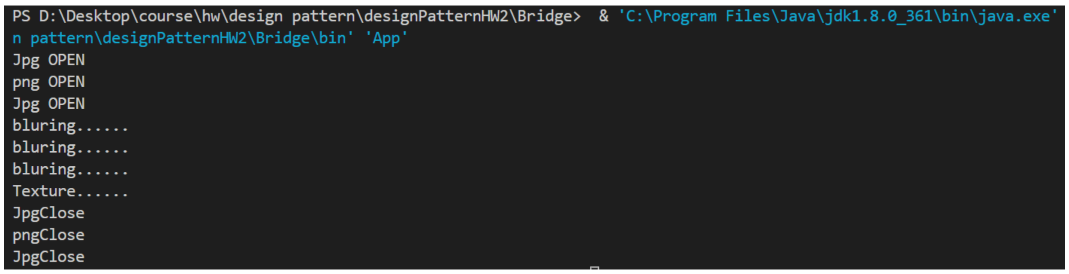
实验要求

某手机美图 APP 软件支持多种不同的图像格式，例如 JPG、GIF、BMP 等常用图像格式，同时提供了多种不同的滤镜对图像进行处理，例如木刻滤镜（Cutout）、模糊滤镜（Blur）、锐化滤镜（Sharpen）、纹理滤镜（Texture）等。现采用桥接模式设计该 APP 软件，使得该软件能够为多种图像格式提供一系列图像处理滤镜，同时还能够很方便地增加新的图像格式和滤镜，绘制对应的类图并编程模拟实现。

类图设计



代码运行效果



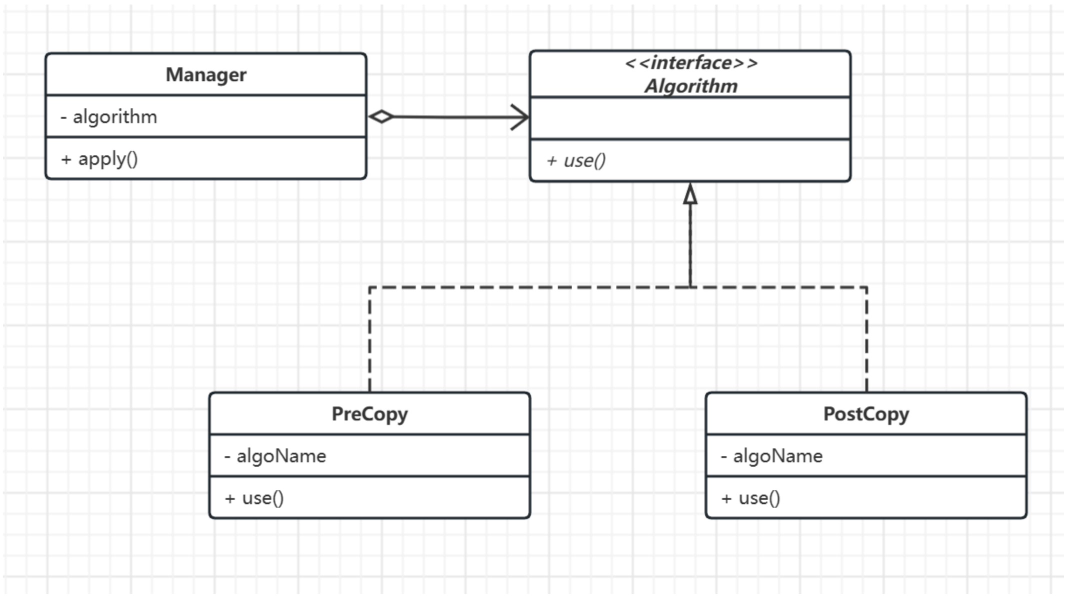
```
PS D:\Desktop\course\hw\design pattern\designPatternHW2\Bridge>  & 'C:\Program Files\Java\jdk1.8.0_361\bin\java.exe'
n pattern\designPatternHW2\Bridge\bin' 'App'
Jpg OPEN
png OPEN
Jpg OPEN
bluring......
bluring......
bluring......
Texture......
JpgClose
pngClose
JpgClose
```

5 策略模式

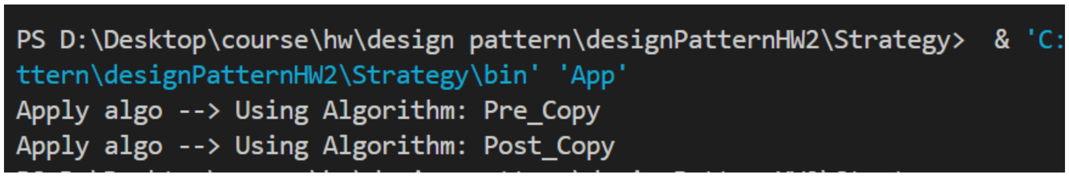
实验要求

在某云计算模拟平台中提供了多种虚拟机迁移算法，例如动态迁移算法中的 Pre - Copy （预拷贝）算法、Post - Copy （后拷贝）算法、CR / RT - Motion 算法等，用户可以灵活地选择所需的虚拟机迁移算法，也可以方便地增加新算法。现采用策略模式进行设计，绘制对应的类图并编程模拟实现。

类图设计



代码运行效果



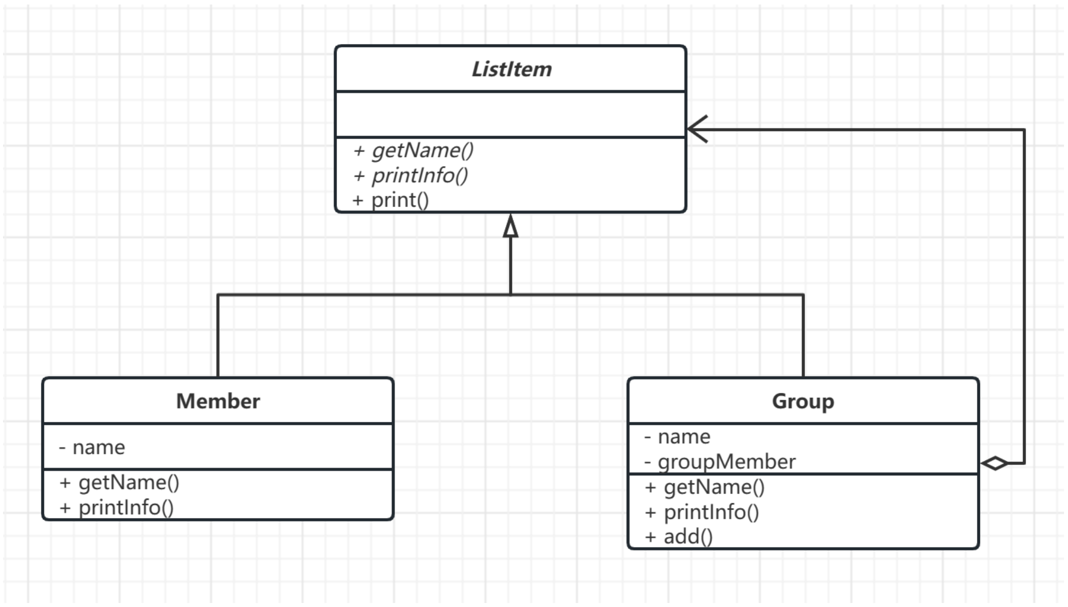
```
PS D:\Desktop\course\hw\design pattern\designPatternHW2\Strategy>  & 'C:
ttern\designPatternHW2\Strategy\bin'  'App'
Apply algo --> Using Algorithm: Pre_Copy
Apply algo --> Using Algorithm: Post_Copy
```

6 组合模式

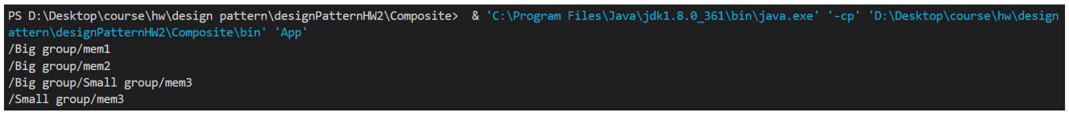
实验要求

某移动社交软件要增加一个群组（Group）功能。通过设置，用户可以将自己的动态信息（包括最新动态、新上传的视频以及分享的链接等）分享给某个特定的成员（Member）. 也可以分享给某个群组中的所有成员；用户可以将成员加至某个指定的群组；此外，还允许用户在一个群组中加子群组，以便更加灵活地实现面向特定人群的信息共享。现采用组合模式设计该群组功能，绘制对应的类图并编程模拟实现。

类图设计



代码运行效果



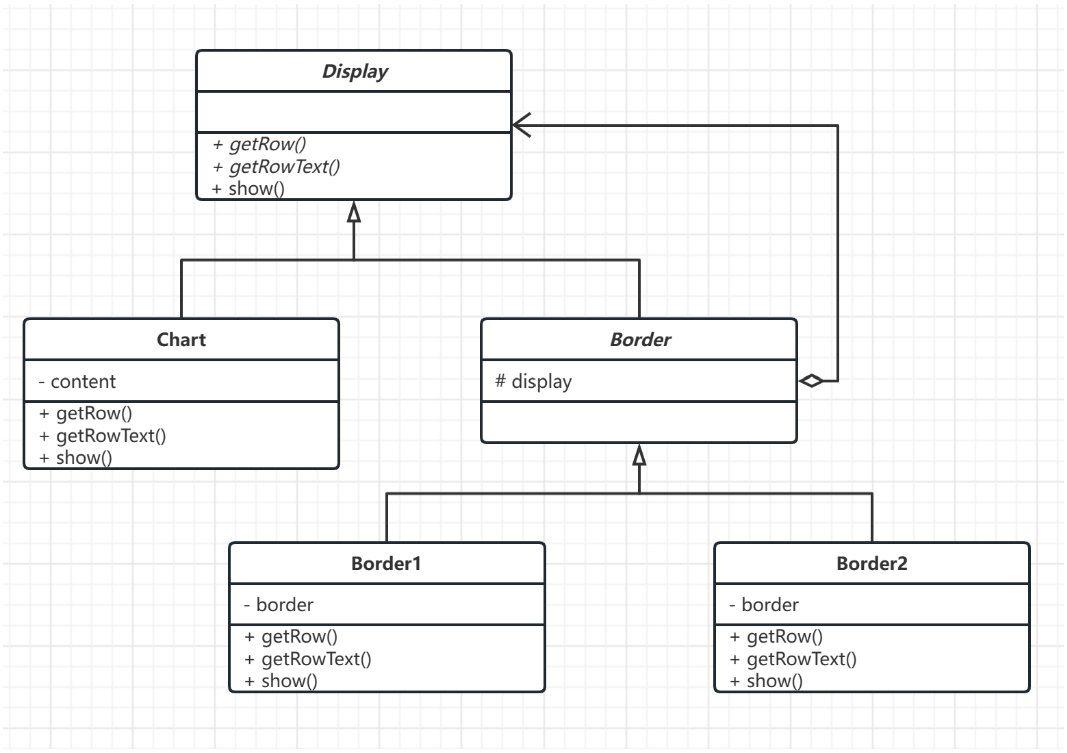
```
PS D:\Desktop\course\hw\design pattern\designPatternHW2\Composite>  & 'C:\Program Files\Java\jdk1.8.0_361\bin\java.exe' '-cp' 'D:\Desktop\course\hw\design
attern\designPatternHW2\Composite\bin' 'App'
/Big group/mem1
/Big group/mem2
/Big group/Small group/mem3
/Small group/mem3
```

7　装饰模式

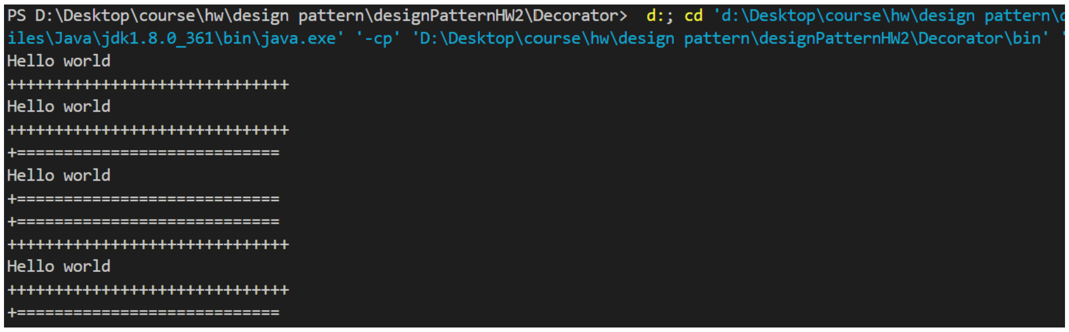
实验要求

在某 OA 系统中提供一个报表生成工具，用户可以通过该工具为报表增加表头和表尾，允许用户为报表增加多个不同的表头和表尾，用户还可以自行确定表头和表尾的次序。为了能够灵活设置表头和表尾的次序并易于增加新的表头和表尾，现采用装饰模式设计该报表生成工具，绘制对应的类图并编程模拟实现。

类图设计



代码运行效果



```
PS D:\Desktop\course\hw\design pattern\designPatternHW2\Decorator>  d:; cd 'd:\Desktop\course\hw\design pattern\c
iles\Java\jdk1.8.0_361\bin\java.exe' '-cp' 'D:\Desktop\course\hw\design pattern\designPatternHW2\Decorator\bin' '
Hello world
++++++++++++++++++++++++++++++
Hello world
++++++++++++++++++++++++++++++
+===========================
Hello world
+===========================
+===========================
++++++++++++++++++++++++++++++
Hello world
++++++++++++++++++++++++++++++
+===========================
```

8 感想

在这次实验中，学会了这 7 个设计模式的相关知识。初步掌握了这些设计模式的原理，并且能够编写代码。在编写代码的时候越来越感觉设计模式的精髓就是把原本具象的代码按照其功能逐步抽象化，使得代码易于修改，维护。