设计模式实验

刘恒星 2022229044

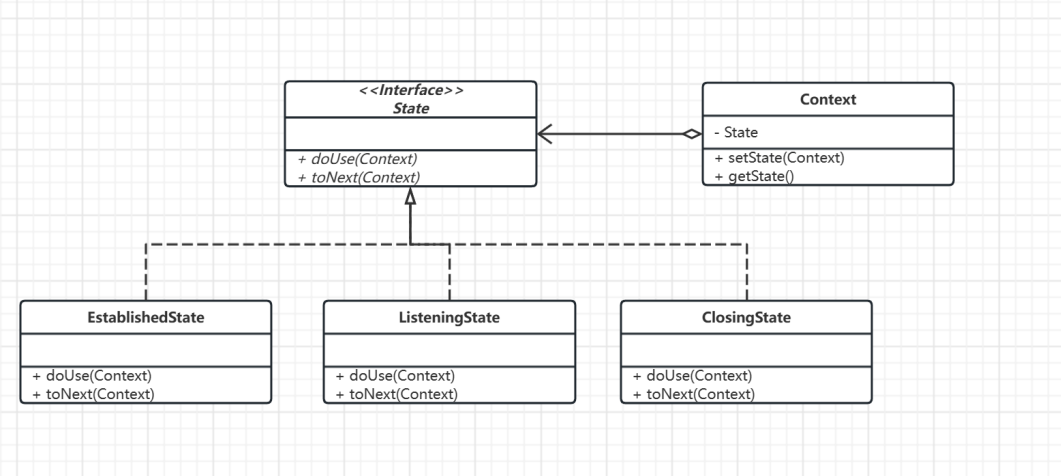
April 10, 2023

1 状态模式

实验要求

在某网络管理软件中，TCP 连接（TCP Connection）具有建立（Established）、监听（Listening）、关闭（Closed）等多种状态，在不同的状态下 TCP 连接对象具有不同的行为，连接对象还可以从一个状态转换到另一个状态。当一个连接对象收到其他对象的请求时，它根据自身的当前状态做出不同的反应。现采用状态模式对 TCP 连接进行设计，绘制对应的类图并编程模拟实现。

类图设计



代码运行效果

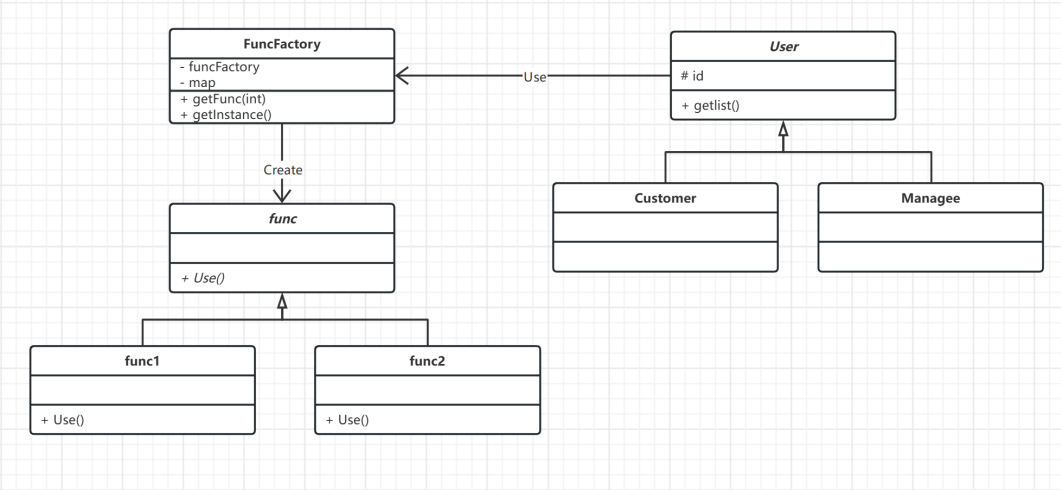
```
PS D:\Desktop\course\hw\design pattern\designPatternHW4\State>  & 'C:\Program Files\Java\jdk1.8.0_361\bin\java.exe
rn\designPatternHW4\State\bin' 'App'
TCP establisted state
EstablishedState......
Changing State......
TCP listening state
ListeningState......
Changing State......
TCP closed state
ClosingState......
Already closed.
ClosingState......
```

2 享元模式

实验要求

某 OA 系统采用享元模式设计权限控制与管理模块，在该模块中，将与系统功能相对应的业务类设计为享元类并将相应的业务对象存储到享元池中（提示；可使用 Map 实现，key 为业务对象对应的权限编码，value 为业务对象）。用户身份验证成功后，系统通过存储在数据库中的该用户的权限编码集从享元池获取相应的业务对象并构建权限列表，在界面上显示用户所拥有的权限。根据以上描述，绘制对应的类图并编程模拟实现。

类图设计



代码运行效果

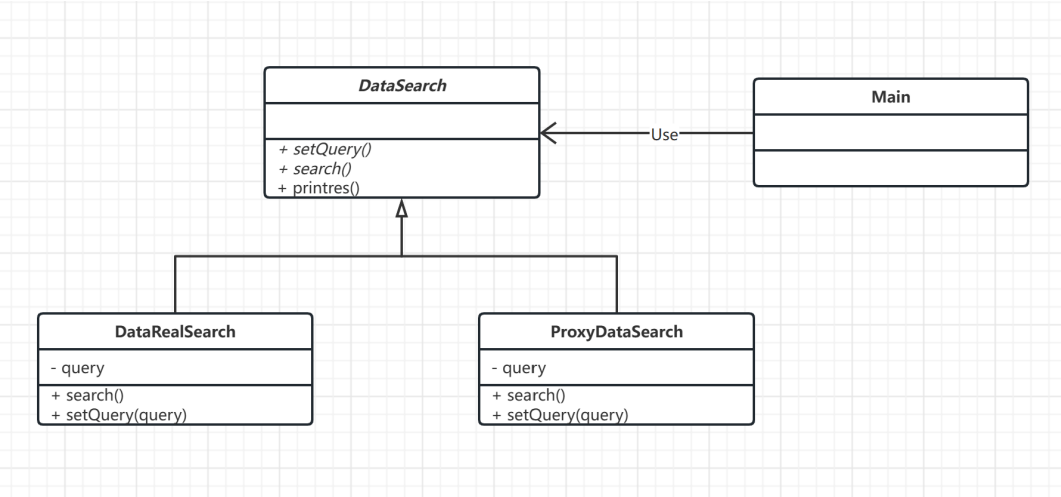
```
PS D:\Desktop\course\hw\design pattern\designPatternHW4\FlyWeight>  & 'C:\Program Files\Java\jdk1.8.0_
gnPatternHW4\FlyWeight\bin' 'App'
User level: 1
PowerList:
First call, create object <func1>
using func1
User level: 2
PowerList:
using func1
First call, create object <func2>
using func2
```

3 代理模式

实验要求

在某电子商务系统中，为了提高查询性能，需要将一些频繁查询的数据保存到内存的辅助存储对象中（提示：可使用 Map 实现）。用户在执行查询操作时，先判断辅助存储对象中是否存在待查询的数据，如果不存在，则通过数据操作对象查询并返回数据，然后将效据保存到辅助存储对象中，否则直接返回存储在辅助存储对象中的数据。现采用代理模式中的缓冲代理实现该功能，要求绘制对应的类图并编程模拟实现。

类图设计



代码运行效果

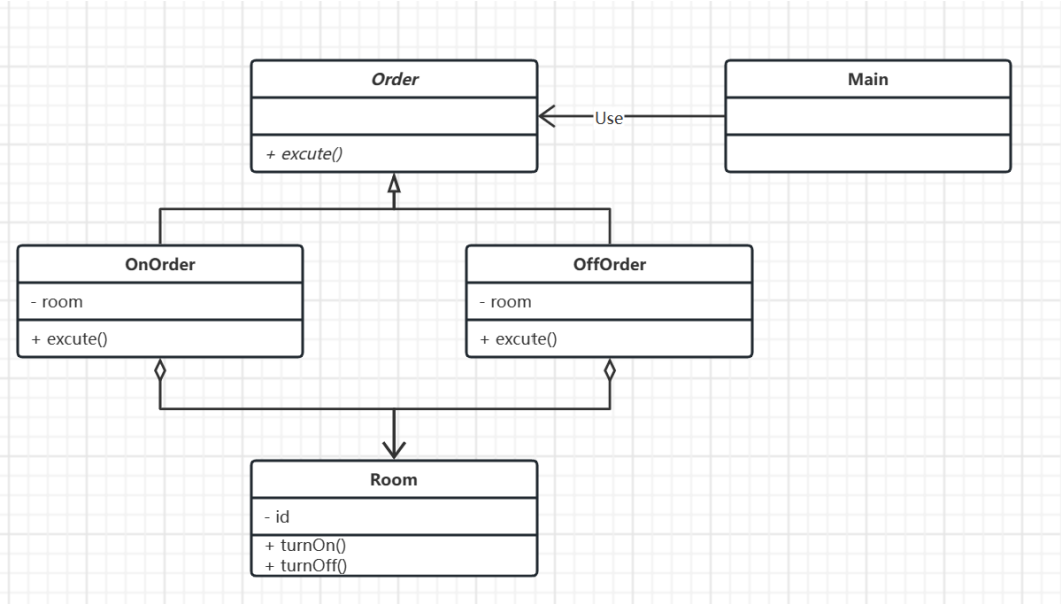
```
PS D:\Desktop\course\hw\design pattern\designPatternHW4\Proxy>  d:; cd 'd
a\jdk1.8.0_361\bin\java.exe' '-cp' 'D:\Desktop\course\hw\design pattern\d
FINDING.......
Respoonse 1: FINISHED
ALREADY EXISTED
Respoonse 1: FINISHED
FINDING.......
Respoonse 2: FINISHED
```

4　命令模式

实验要求

某灯具厂商要生产一个智能灯具遥控器，该遥控器具有 5 个可编程的插槽，每个插槽都有一个控制灯具的开关，这 5 个开关可以通过蓝牙技术控制 5 个不同房间灯光的打开和关闭，用户可以自行设置每一个开关所对应的房间。现采用命令模式实现该智能遥控器的软件部分，绘制对应的类图并编程模拟实现。

类图设计



代码运行效果

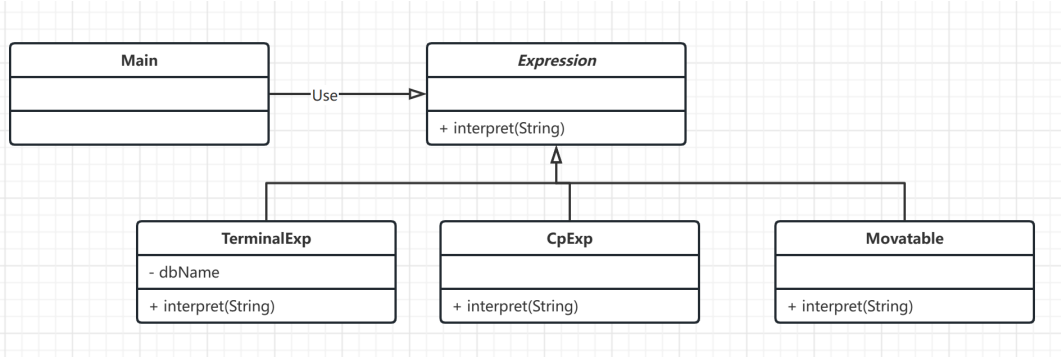
```
PS D:\Desktop\course\hw\design pattern\des
tern\designPatternHW4\Command\bin' 'App'
Room 0 turns on the light
Room 0 turns off the light
Room 1 turns on the light
```

5 解释器模式

实验要求

某软件公司要为数据库备份和同步开发一套简单的数据库同步指令，通过指令可以对数据库中的数据和结构进行备份。例如，输入指令“COPY VIEW FROM srcDB TO desDB ”，表示将数据库 srcDB 中的所有视图（View）对象都拷贝至数据库 desDB ；输人指令“MOVETABLE Student FROM srcDB TO desDB ”，表示将数据库 srcDB 中的 Student 表移动至数据库 desDB 。现使用解释器模式来设计并编程模拟实现该数据库同步指令系统。

类图设计



代码运行效果

```
PS D:\Desktop\course\hw\design pattern\designPatternHW4\Interpret>  d:; cd 'd:\Desktop\course\hw\design pattern\designPatterr
iles\Java\jdk1.8.0_361\bin\java.exe' '-cp' 'D:\Desktop\course\hw\design pattern\designPatternHW4\Interpret\bin' 'App'
COPY VIEW FROM srcDB TO desDB: true
MOVETABLE Student FROM srcDB TO desDB: true
MOVETABLE Student FROM srcDB TO UnknownDB: false
```

6 感想

在这次实验中，学会了所有的设计模式，初步学习了这些设计模式之后，对设计模式有了一些自己的认识。设计模式是面向更好的编程架构的产生的，让代码能更好的维护，有更高的效率，是一个很有利的准则。