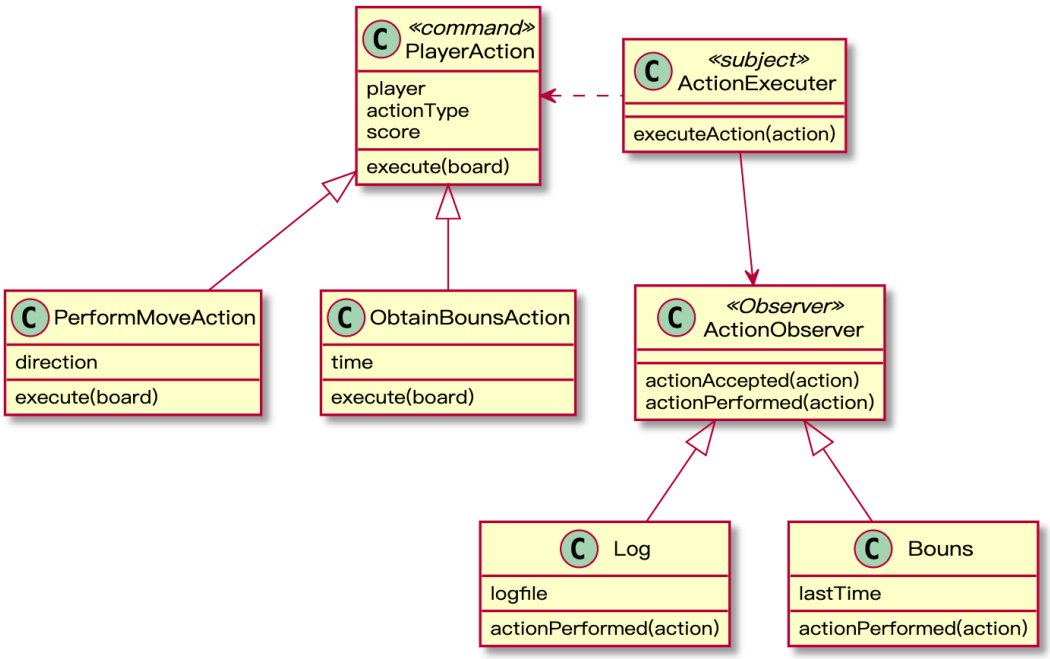
Lab7 说明及 Lab8 设计参考

Lab7 的第一问多数同学选择了观察者模式，对于第二问，很多人的直觉也是观察者模式，但是似乎各种尝试都有些问题。大家对自己代码存在的问题的分析很好，事实上，在实际项目中使用设计模式时，过度设计或是不恰当地使用确实会让代码变得更加复杂。

第二问比较经典的解决方案实际上需要用到设计模式中的**命令模式**（这个在 Lab7 中不做要求，大家只需要使用已经学过的技术尝试做解耦），下面给出参考的解法：



1. **PlayerAction** 目前指的是可能对玩家的总分产生影响的一个“动作”。
2. **ActionExecuter** 可以就是 Game 或者 自己原来代码中已有的合适的类型。
3. 这个模式应用恰当，可以用来作为 **Lab8** 的设计。

```
class PerformMoveAction : public PlayerAction{
    direction
    execute(board){
        //在这里调用原来对应于的 move 的代码，
        //需获取用户最终获得的 score，
        //而原先调用 move 的地方，改为构造一个
        //PerformMoveAction，然后用 ActionExecuter 去执行。
        score = ???.move(player,board,direction)
    }
};
```

```
class ObtainBounsAction : public PlayerAction{
    timeDuration
    execute(board){
        score = 1
        player.addScore(score)
    }

};
//可以就是 Game 或者其他已有的合适的类型
class ActionExecuter{
    observers: ActionObserver[]
    executeAction(action){
        for(auto o : observers) o.actionAccepted(action);
        action.execute(board)
        for(auto o : observers) o.actionPerformed(action);
    }
};

class Logger : public ActionObserver{
    logfile:ofstream;
    void actionPerformed(action){
        logfile << action << endl;
        //也可以用 switch/case 判定 action 的类型，分别输出
    }
};

class Bouns : public ActionObserver{
    actionExecuter : ActionExecuter
    lastMovementActionTime
    void actionPerformed(action){
        if(action.actionType == movement){
            timeDuration = ...
            //check movemeent time duration, if bouns criteria is satisfied:
            actionExecuter.execute(
                new ObtainBounsAction(action.player,timeDuration)
            );
        }
    }
};
```

注意： 上述设计的伪代码仅供参考，只是提供框架性的解决方案。如果使用这个模式改造自己的代码，其中类里具体的属性，方法中的参数以及类型需要根据各自的代码进行调整。