本次多电梯多线程调度工程，共有以下 9 个类———

类及其职能：

GlobFlag：全局标识类，内含全局标志 TimeFlag，且对标志的权限控制的很好，有效地保护了数据；

Timer：时钟类，作为一个单独的线程，稳步增加时间；

Request：命令请求类，作为一个单独的线程，将输入的命令很好地进行了处理，方便了后续操作对输入信息的获取；

Queue：队列类，在命令请求类的基础上，将各命令组合成队列，并对队列的有效性和合理性进行了检查；

Simulator：模拟器类，作为一个单独的线程，根据时间变化，将队列中的命令根据时间抽取到等待队列中；

Queuing：等待队列，存放和时间同步的即将被执行的命令；

Manager：调度类，作为一个单独的线程，将待处理指令从等待队列中根据相应的原则调度到合适的电梯中；

Elevator：电梯类，作为一个单独的线程，根据调度器调度过来的指令进行响应的动作，实现响应的指令；

Test：主类，将各个类结合到一起，实现了多电梯多线程调度任务。

本次工程，各个类都遵照了“懂我原则”，各类和变量都进行了有效的命名。最为重要的是这些类的层次化抽象都相当合理，类的划分都是按照对问题的逻辑处理方式进行的，所以职能明晰，各司其职，分工合理，不存在无用的类，也不存在任务量太过杂乱的类，总体来说还是比较不错的。

样例解析———

GlobFlag.java

```
1
2 public class GlobFlag {
3     private int timerFlag = 1;
4
5     private GlobFlag(){}
6
7     private static GlobFlag instance = new GlobFlag();
8
9     public static GlobFlag getInstance(){
10        return instance;
11    }
12
13    public int getTimerFlag() {
14        return timerFlag;
15    }
16
17    public void setTimerFlag(int timerFlag) {
18        this.timerFlag = timerFlag;
19    }
20
21 }
```

GlobFlag 类的主要职能为提供全局变量，供线程之间调度用。

1.首先，在 GlobFlag 类中定义 timeFlag 变量，设为私有对其进行保护。

2.函数第五行，将本类的构造函数设为私有，防止在外界生成 GlobFlag 类的实例，从而避免了 timeFlag 标志被任意更改，是 timeFlag 的安全性大大提升。

3.紧接着，构建静态的 getInstance 方法。由于不能在外部实例化对象，因此使得方法可以直接通过类名静态调用。

4.最后，针对返回的唯一的对象，只能通过 setTimeFlag 函数对其值进行修改，数据安全性保护的很好。