**设计方法：**

在设计之初我们最先想到的就是模块化的设计，这其中比较重要的就是模块之间如何减少联系，最好使模块之间的联系度达到最小，这也就涉及到了信息的隐藏度。所以我们参考了Parnas方法，Parnas方法主要是以信息隐蔽为原则的，在使用这个方法的时候我们需要列出那些可能发生变化的因素，在设计的时候让这些因素只出现在需要的模块里，而其它模块不受该因素的影响，到时候若出现需求的改变我们直接修改这个模块即可。所以我们在最开始的部分把需要用到数据库数据的地方都写成一些调用，之后如若每个模块用到这些数据直接调用即可，每个模块之间互不影响，因为数据库是实时更新的。且每个模块之间只有少数部分有联系，这些地方我们用到了自顶向下的方法，比如买票的部分，我们会先进入买票模块，然后进入选择影院模块，在进入选择作为模块，最后完成买票操作。这些部分使用了自顶向下的设计方法，这些模块之间是有一定联系的，但是这样设计也是为了避免一个模块的功能太多，实现起来太过麻烦，万一出现问题不容易修改等。除了类似这样的模块，其他模块都是相互独立的。

**设计原则：**

1. **模块化**

模块化也被称作关注点分离，也就是把系统中各不相关的部分进行分离的原则，这样利于各部分能够进行独立研究。为了实现模块化，我们要辨析系统不相关的关注点来分解系统，并把他们放入不同的模块中。这样可以是运行速度大大加快，也可以使维护和开发变得非常便利。评估模块化有两个概念，

其一就是耦合度，我们在这方面做的比较好，就比如我们上面设计方法提到了，我们在这方面使用了parnas方法，实现了大部分情况下，各个模块之间的数据互不相关，各个模块之间也基本没有联系。

二就是内聚度。也就是指模块内部各个数据之间的“粘合”程度。模块内部粘合度越高，我们总体目标就越相关，整体性能就越好。为了实现高内聚度，我们在实现代码之前，先画出了数据库各个部分之间的联系，通过他们之间的联系，我们来设计模块，尽量的使模块内部之间的数据具有高内聚度的特性，减少独立数据。

评估：在实现模块化上，我们整体做的还是不错的。为了实现耦合度，我们采用parnas方法来实现高耦合度，但也不可避免的存在了模块之间具有数据互通的弊端，我们思考很久没有找到很好的解决方法，所以最后采用了自底向上的方法来解决这个问题。在内聚度方面，我们刚开始也是没想到很好的方法，后来想到了大二上的数据库结构课设。所以我们先画出数据库结构，根据各个部分之间的联系来设计模块，尽量减少模块之间的联系，并且加大模块内的内聚度。

**2．接口**

接口，这个概念很好理解，为系统其他部分定义了该软件单元的服务， 以及如何获取这些服务。它制定了操作性名称，参数和可能返回的值。当然， 为了数据的安全，我们只有在上下文环境符合一定条件时才被调用，一旦不 满足其中任意一个，都不会得预期的功能。也就是说软件单元的接口描述了 不同环境的需求和不同的服务。

评估：我们的接口实现的比较简单，指示在调用模块内，写入了所有可 能被调用的数据接口，这样从数据库中获取信息，或者向数据库中写入信息 的时候，只需要调用这些接口就可以了。为了保护数据的安全，防止数据库 数据错误导致崩溃，我们在调用的时候返回一个值，若调用错误会返回一个 固定的值，这时就不会得到预期的服务。