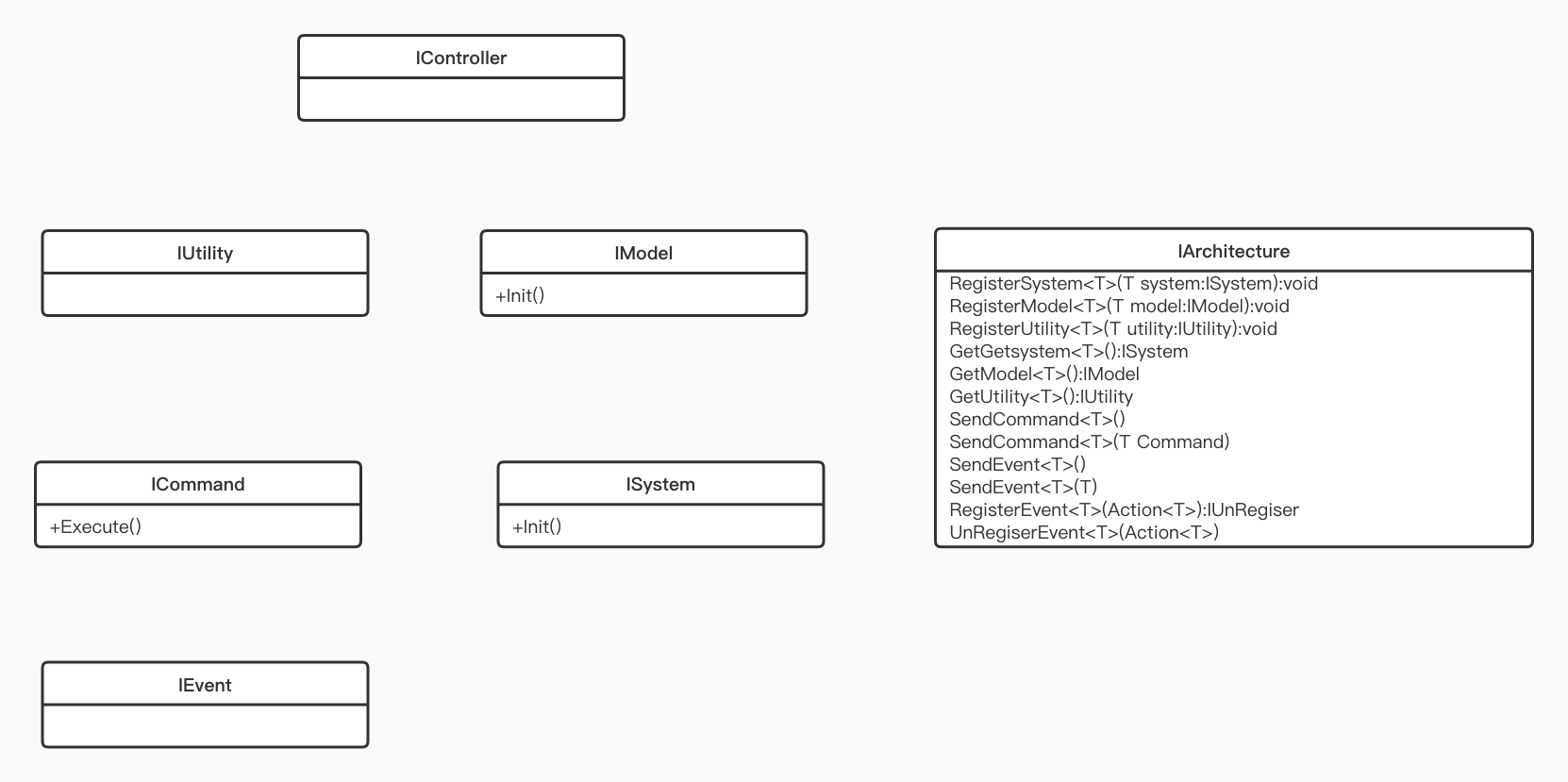
# QFrameWork

UML图：



IController: 在QFrameWork中 View和Controller两个合并为一块。一般与MonoBehavior一起使用。具备发送命令和监听事件职能。通过监听Event事件进行刷新，通过Command的更新Model。

IUtility:工具类，一般用于System和Model。不具备任何职能。

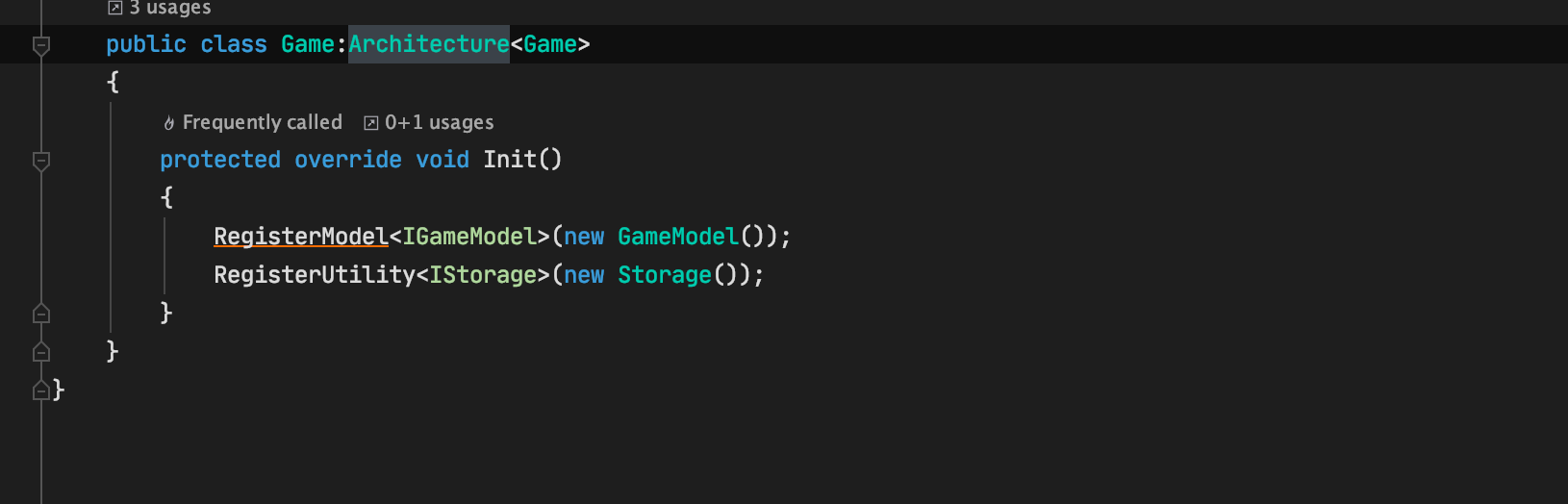
IModel:数据类。用于管理数据。具有发送事件职能。

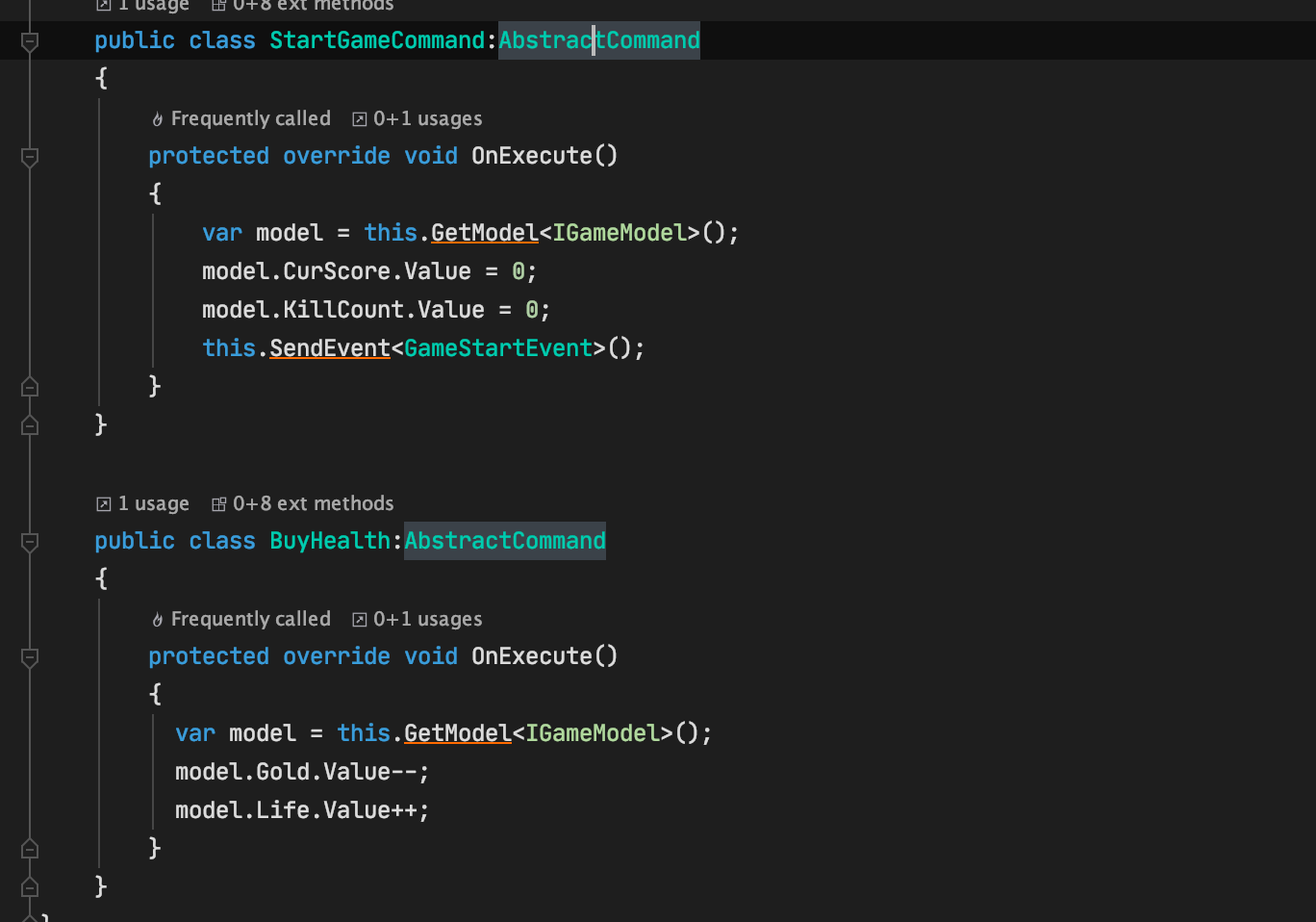
ISystem:系统类，用于处理与数据相关的处理，如成就系统等。具有监听事件，发送事件职能。

ICommand：命令类，用于执行操作数据或系统。具有发送事件和发送命令职能。可以在命令中调用命令实现复用。

IArchitecture: 架构类，用于绑定除Controller以外的其他类。各个类的交互以及事件和命令都是通过该类进行转发和调用。用于保证数据和View完全解耦。

在项目中，Controller通过GetArchitecture设置架构类。通过监听事件和发送命令完成数据更新和视图刷新。Architecture中都是通过接口注册，保证后续切换了具体的类不会对其他类造成影响，其他类也是通过接口对具体的类进行操作。

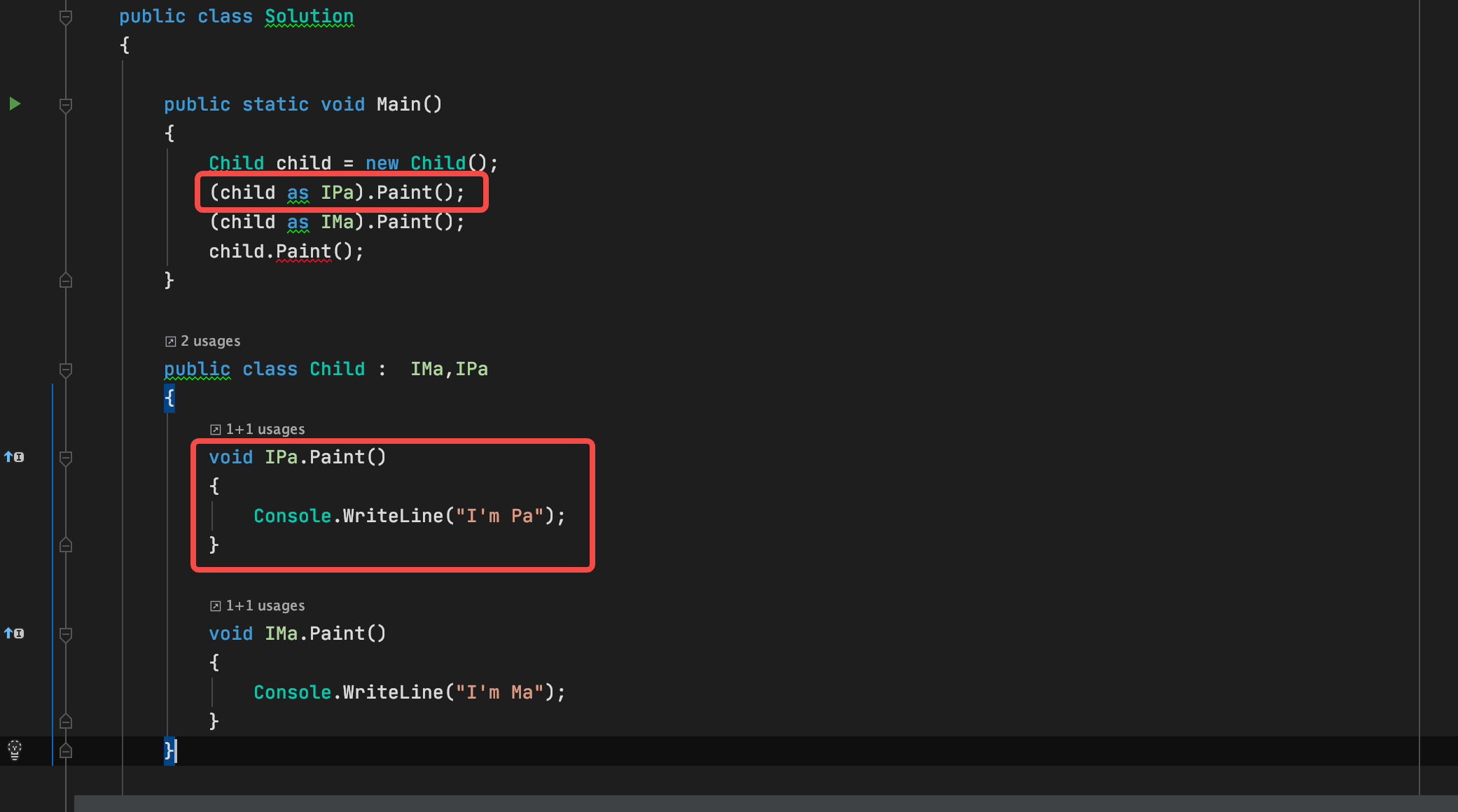




（其中，项目通过接口使各部分职责分明 图如下）



PS:当有多个接口中的方法同名时，但你又不希望为两个接口调用同一个实现时。可以显式实现接口成员。通过显式实现的是一个类成员，只能通过指定接口进行调用。



QFramework优缺点

好处：解耦，Model，Controller，Command 三个部分相互独立，一个具体的类被改变时可以完全不影响到其他的类。具有灵活性，扩展性好。通过IArchitecture统一管理，提供了可控性，当想要过滤或记录某些命令或事件时，可在Architecture中进行处理而不需要修改其他的代码。

缺点：每次新增操作都需要新建Command类，增加了系统的复杂性。通过接口调用保证了与具体的类解耦，同时也意味着需要创建更多的类（接口）。