**设计模式说明**

成员：蒋博为、苏洋、郑友强

**1、设计模式：**

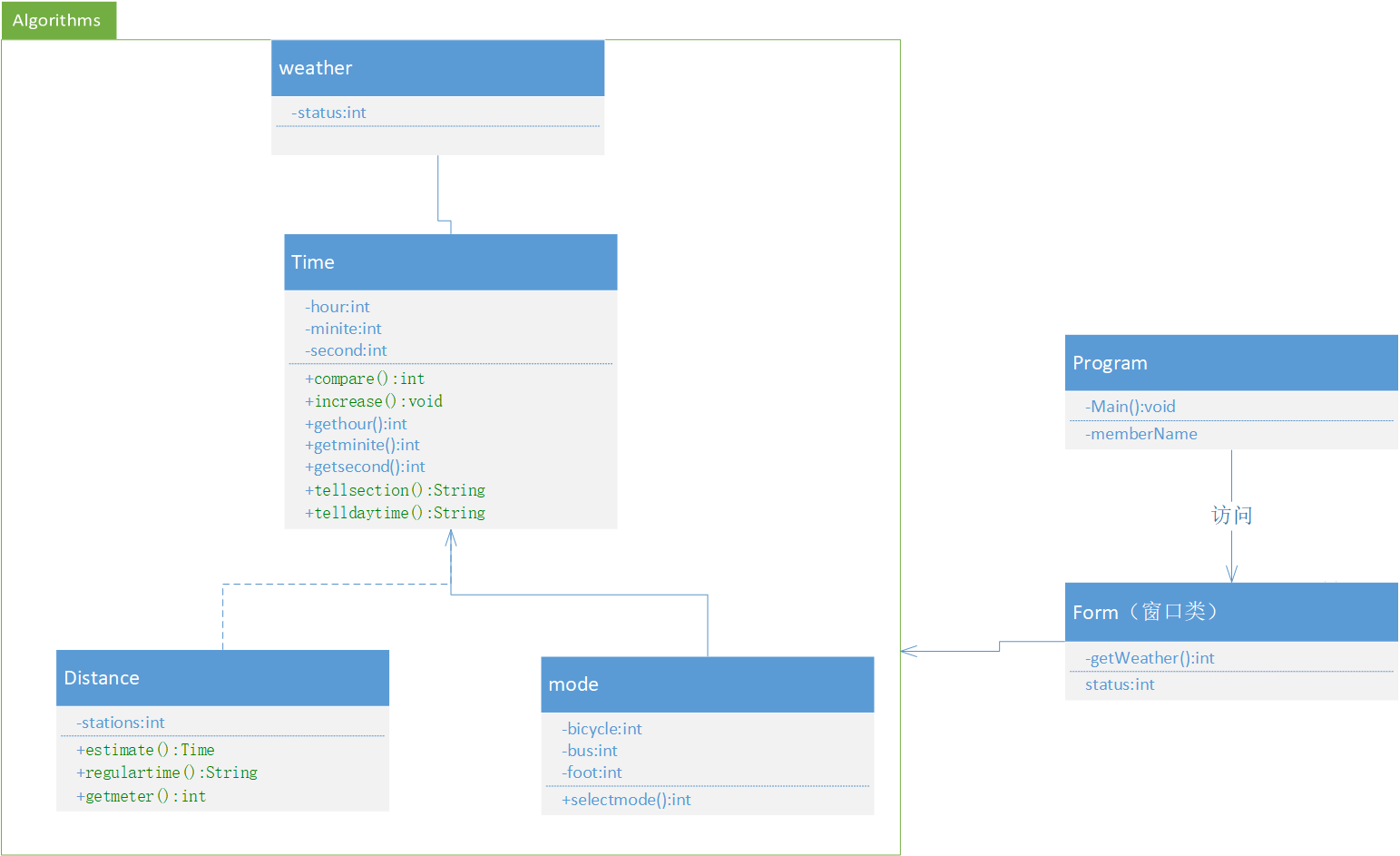
在本次开发过程中，我们共使用了三种设计模式，分别为外观模式、观察者模式、单例模式。

外观模式：我们使用C++代码编写后端，用C#代码编写前端页面，两者分属Algorithm和SchoolBus两个不同的project，互不影响，减少系统相互依赖，提高了灵活性，提高了安全性。与此同时，降低访问复杂系统的内部子系统时的复杂度，简化客户端与之的接口。采用外观模式隐藏了系统的复杂性，给客户提供了多种选择，既可以预测到达时间，也可以获取推荐信息，同时也可以获取各个站点常规时间表等信息。

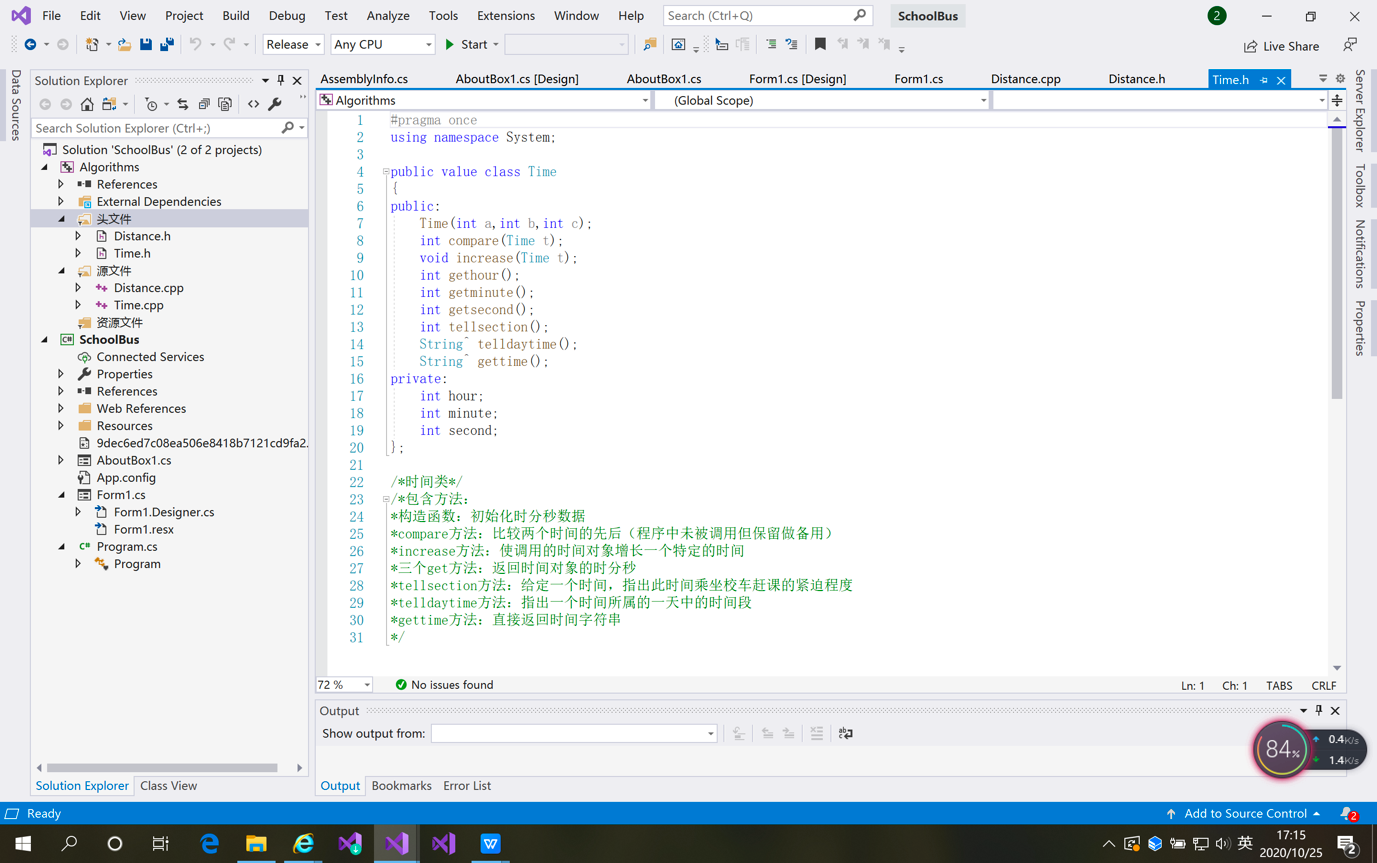
观察者模式：我们在C#前端代码中定义一个全局变量status，这个变量会随从第三方平台上获得的天气的变化而变化，定义一个天气类的对象（被观察类），Distance类（实现时间估计等）中的方法通过观察当日天气变化发生与之对应的变化。观察者模式可以让观察者和被观察这是抽象耦合的，建立一套触发机制实现变化。采用观察者模式可以降低对象之间的耦合度，在系统中创建一个触发链，当当日天气为“阴雨天气”时，对应的车辆的停留时间和行车时间随之延长。

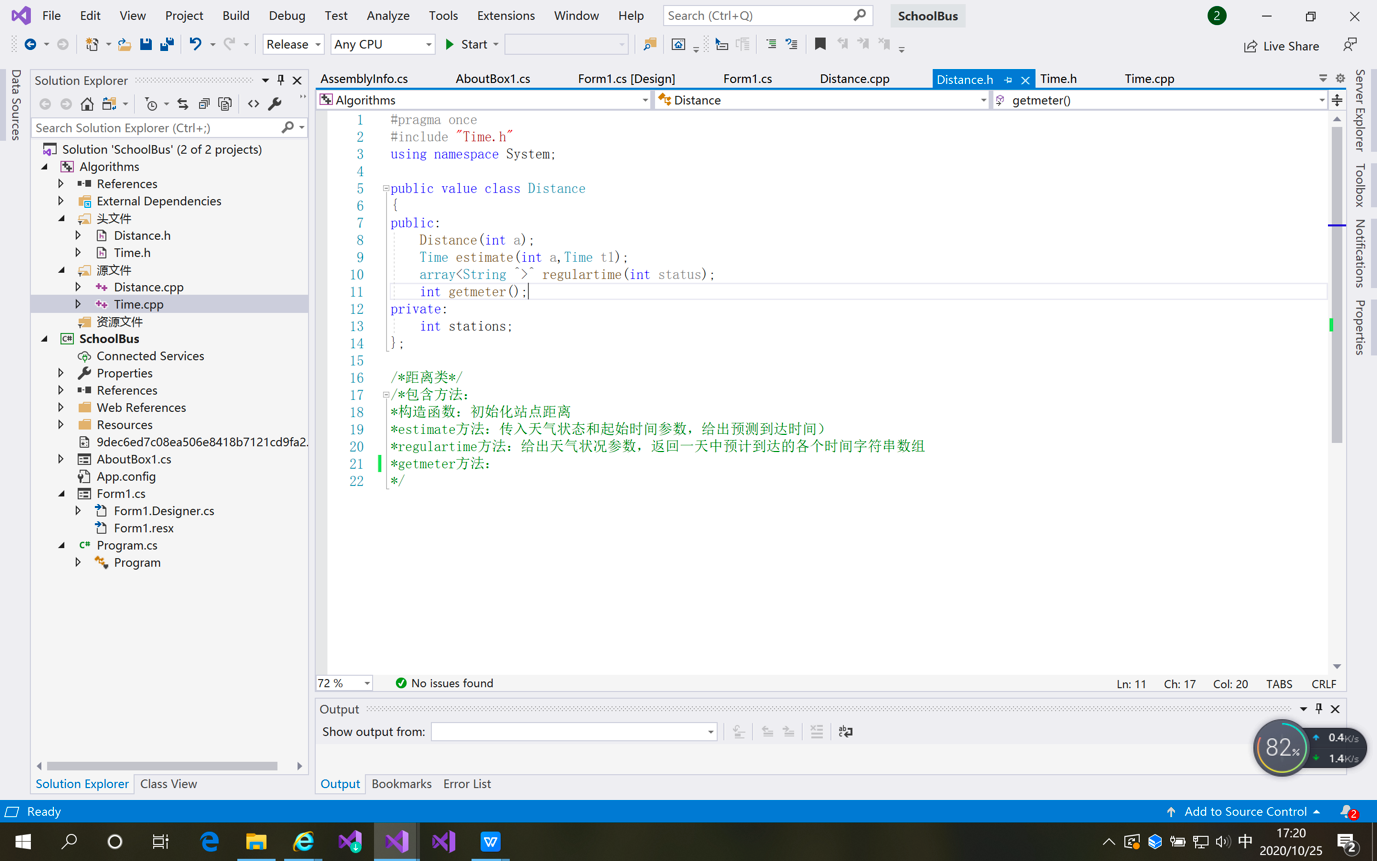
**2、具体实现（部分）**

（1）外观模式：



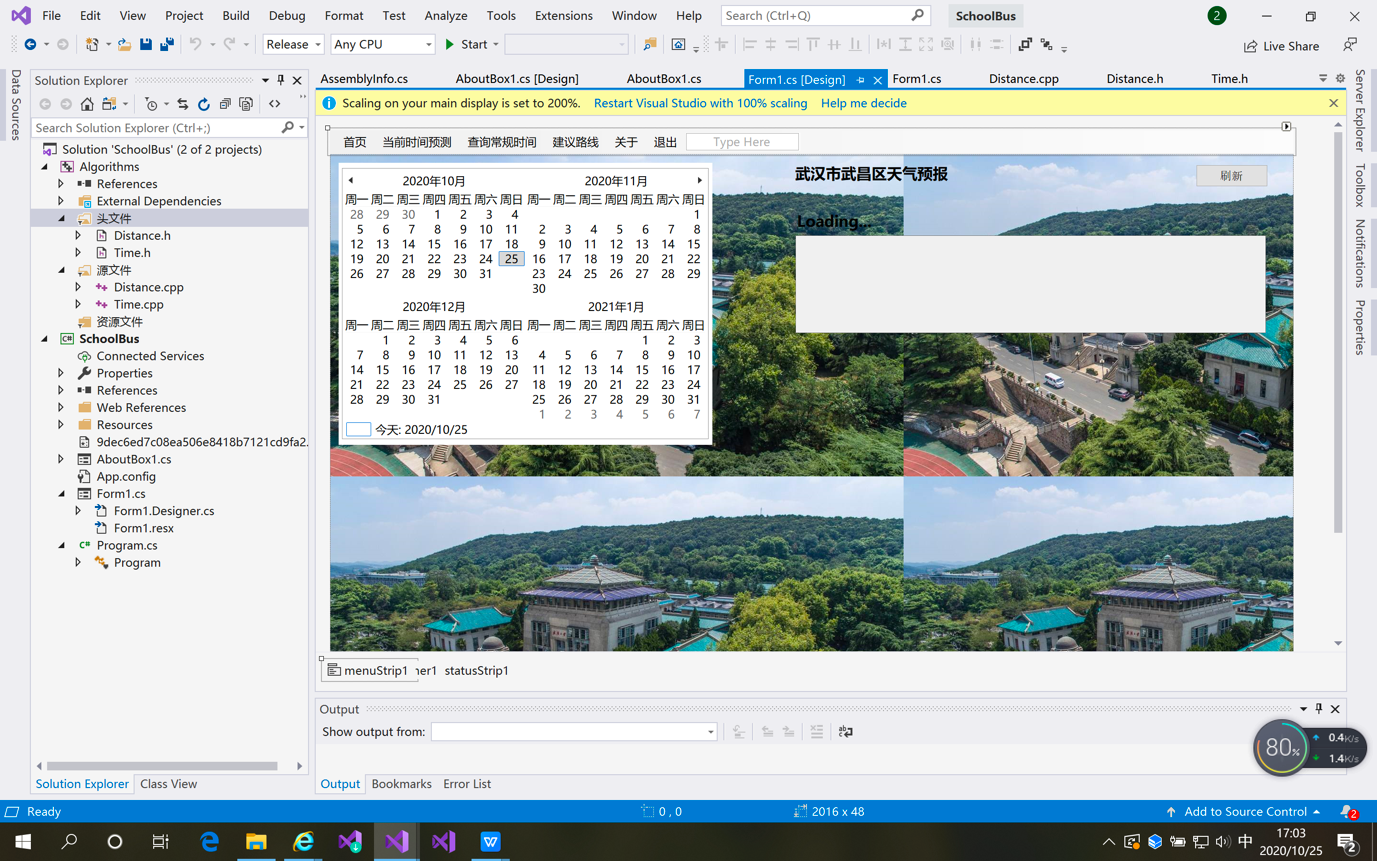
后端C++（代码实现）：



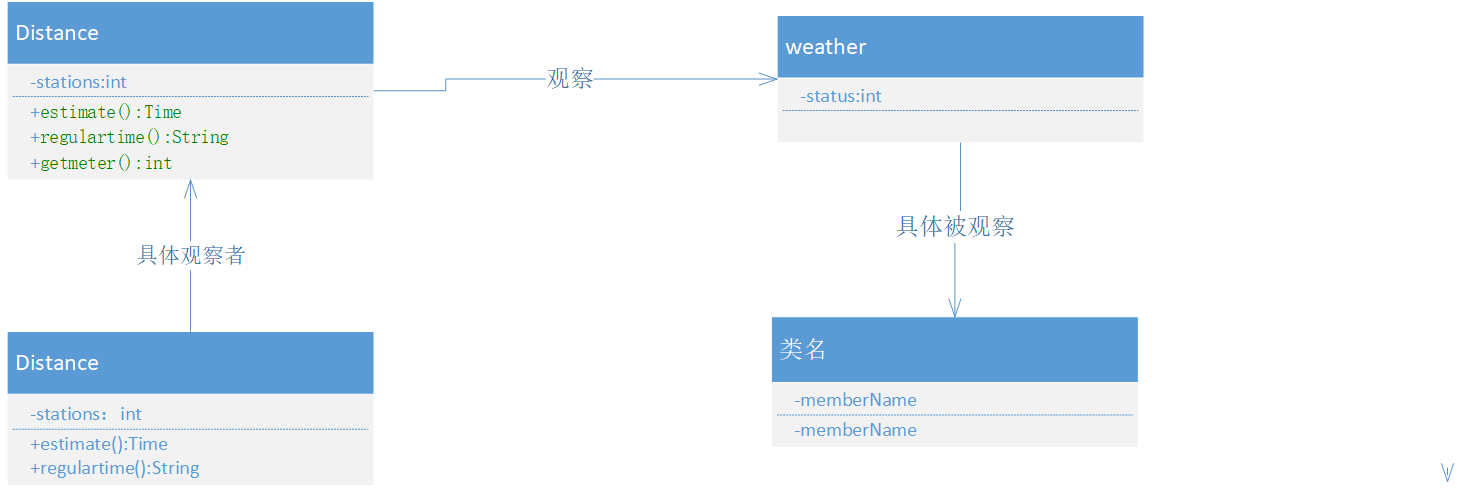


前端C# winform代码实现效果图

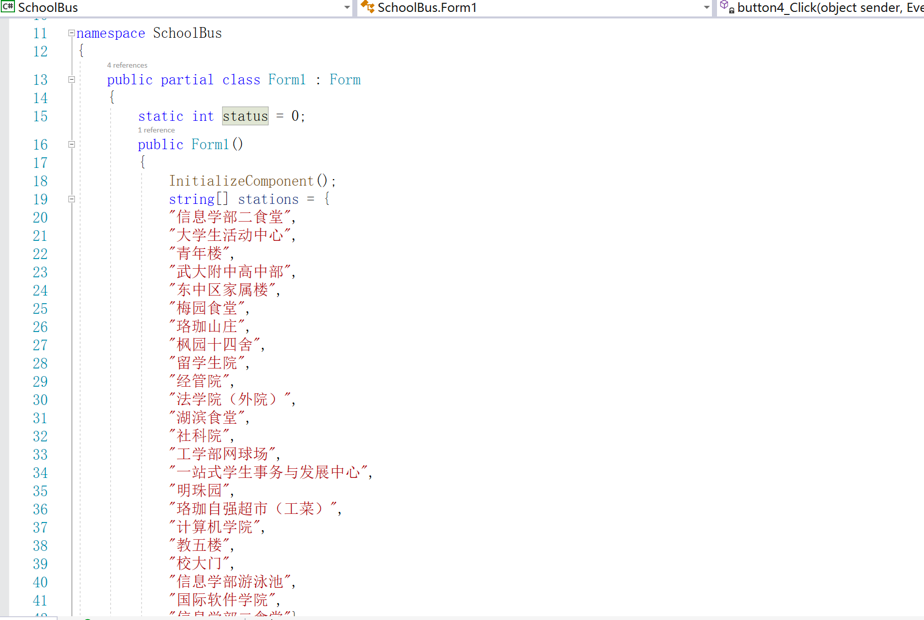
前端C# winform：

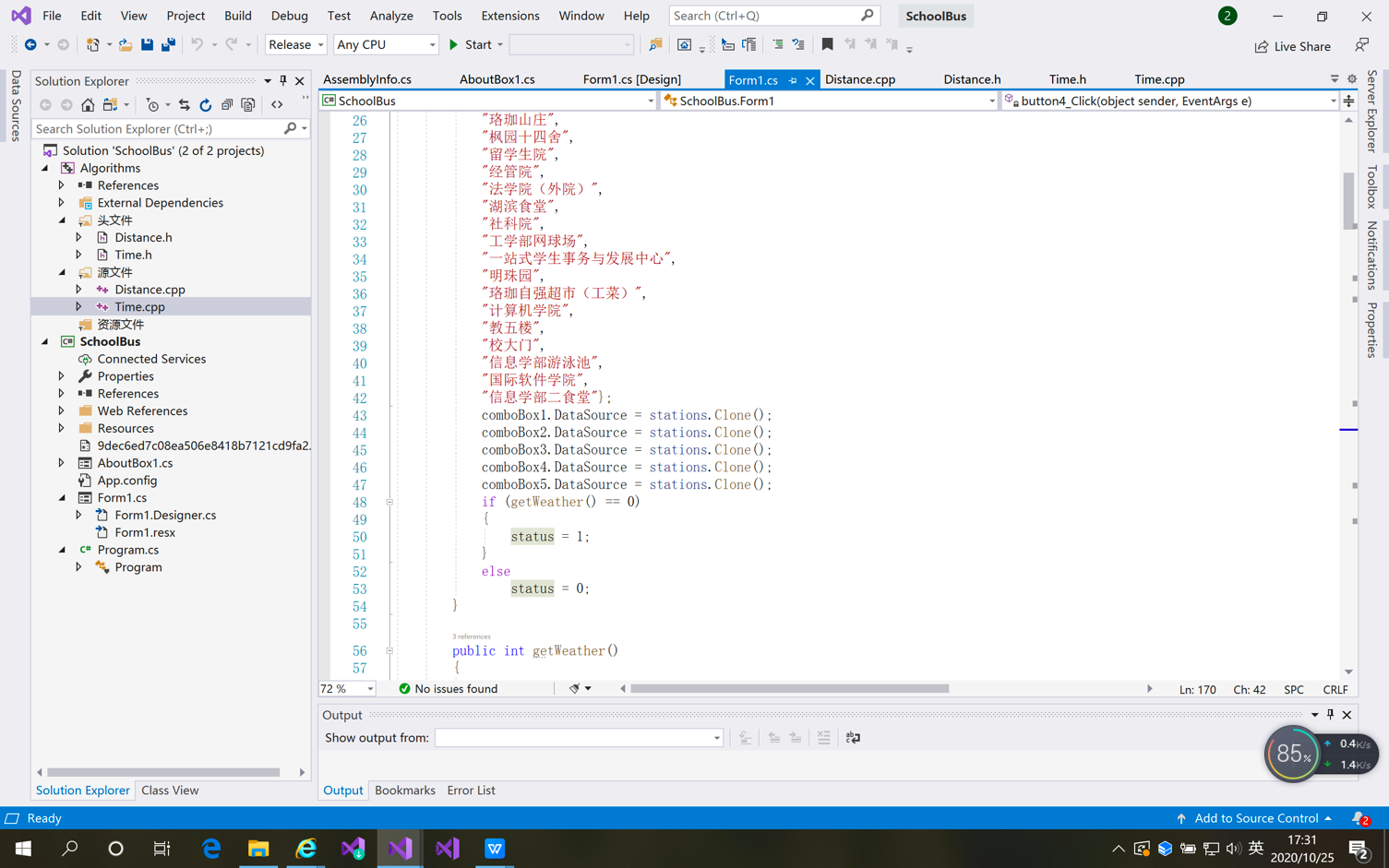


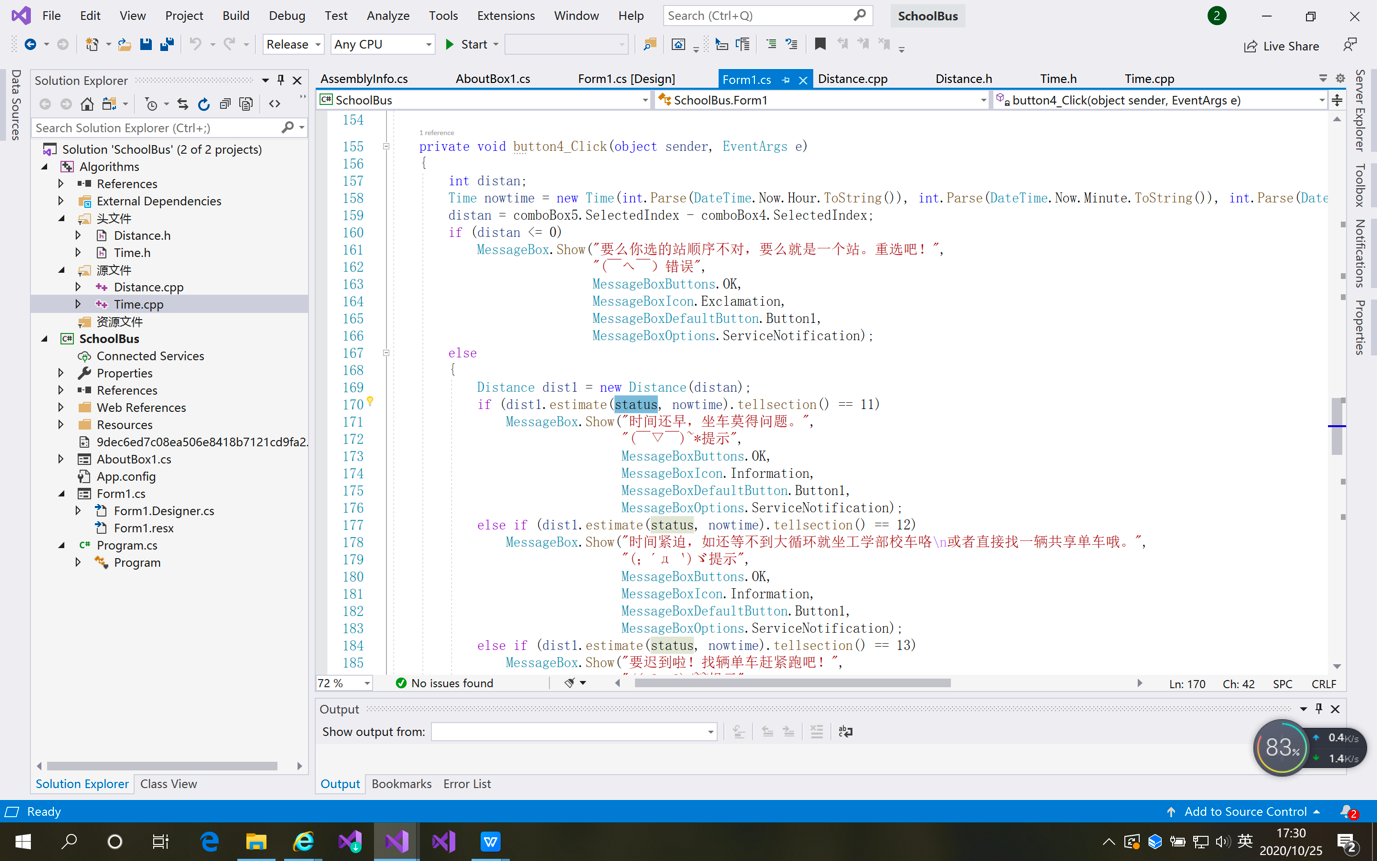
（2）观察者模式：

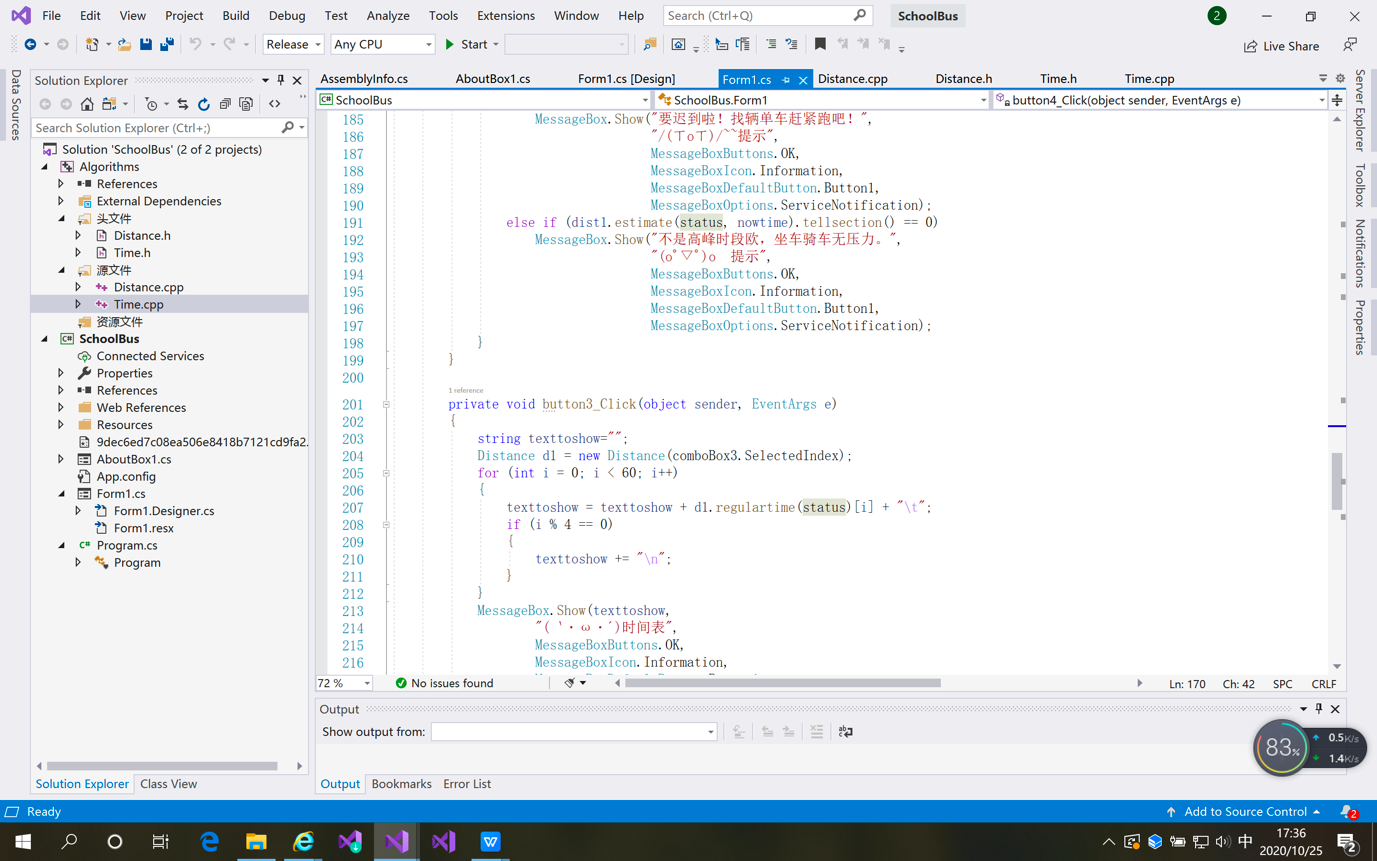


被观察status的代码实现：





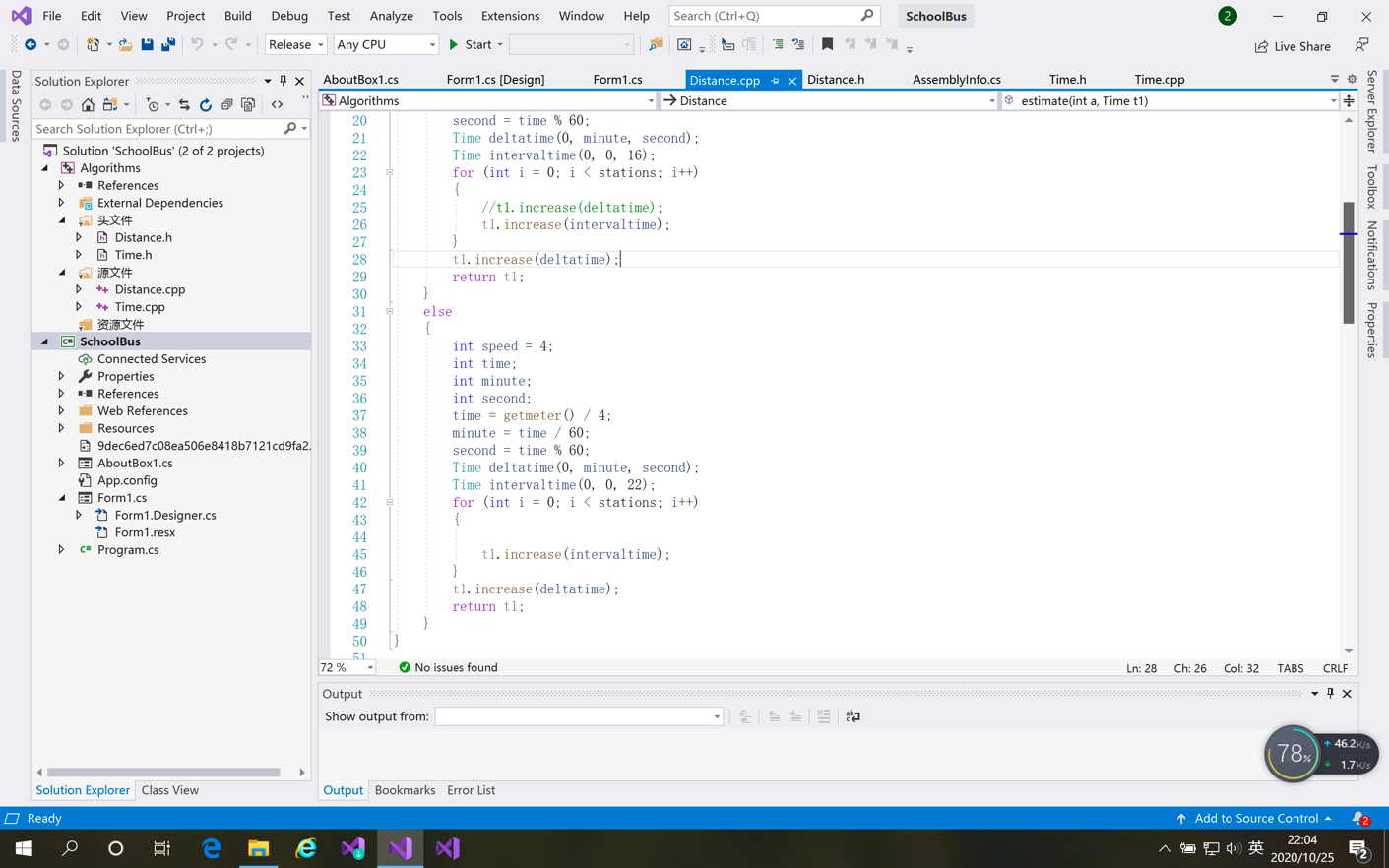




观察者代码实现：

Distance中estimate方法实现：





Distance中regulartime方法实现：

