1. Summarise the main points in this module. You may include references to the learning objectives.
2. The introduction of Design patterns
3. Features of Design Patterns: Smart, Well-Proven, Reusable, Generic, Simple, Object-Oriented.
4. A Design Pattern has a pattern name, the problem, the solution, the consequences.
5. Categorizing Design Patterns: Purpose, Structural, Behavioural, Scope, Class, Object.
6. Factory Patterns: Simple Factory, Factory Method, Abstract Factory.
7. Singleton.
8. Façade.
9. Observer.
10. Word of Caution.
11. How is this useful?

Design patterns are general, reusable solution to commonly occurring problem within

a given context in software design. Design patterns are used to reusable code, make code easier to understand by others, and ensure code reliability.

1. How do you plan to use this information?

In the future programming, think first, draw UML diagrams, then select the appropriate design pattern according to the actual needs, and finally implement the code.

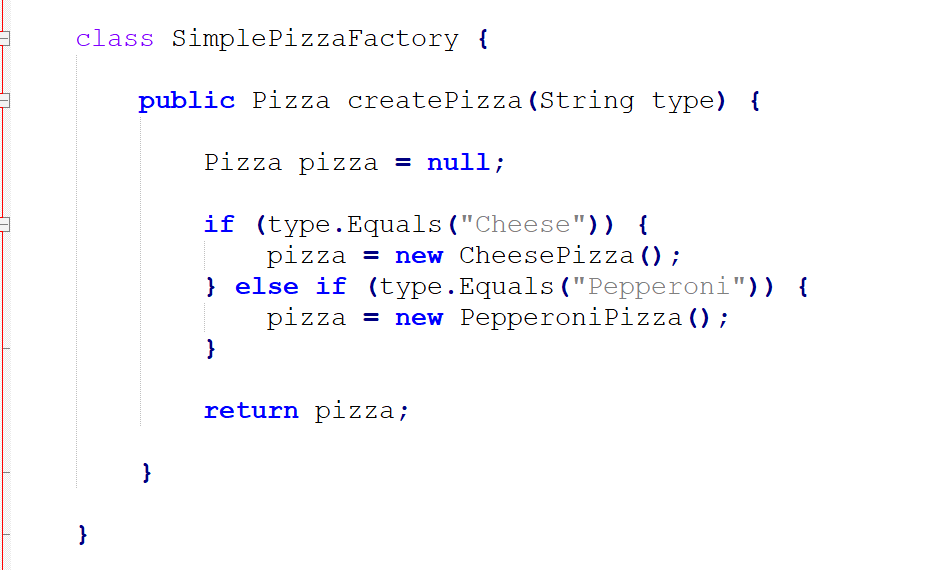
1. Provide summary of your reading list — external resources, websites, book chapters, code libraries, etc.
2. <http://hillside.net/patterns/patterns-catalog>.
3. Patterns that help to model situations that often arise during analysis.
4. Use concepts rather than actual classes to be implemented in code.
5. Design Patterns: Elements of Reusable Object- Oriented Software (1995) by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (a.k.a. GoF, The Gang of Four).
6. Used object modeling techniques to represent common solutions to problems in the design of OO software, taken from multiple actual systems.
7. Since the GoF book came out, there have been many more.

**Reflecting on the content:**

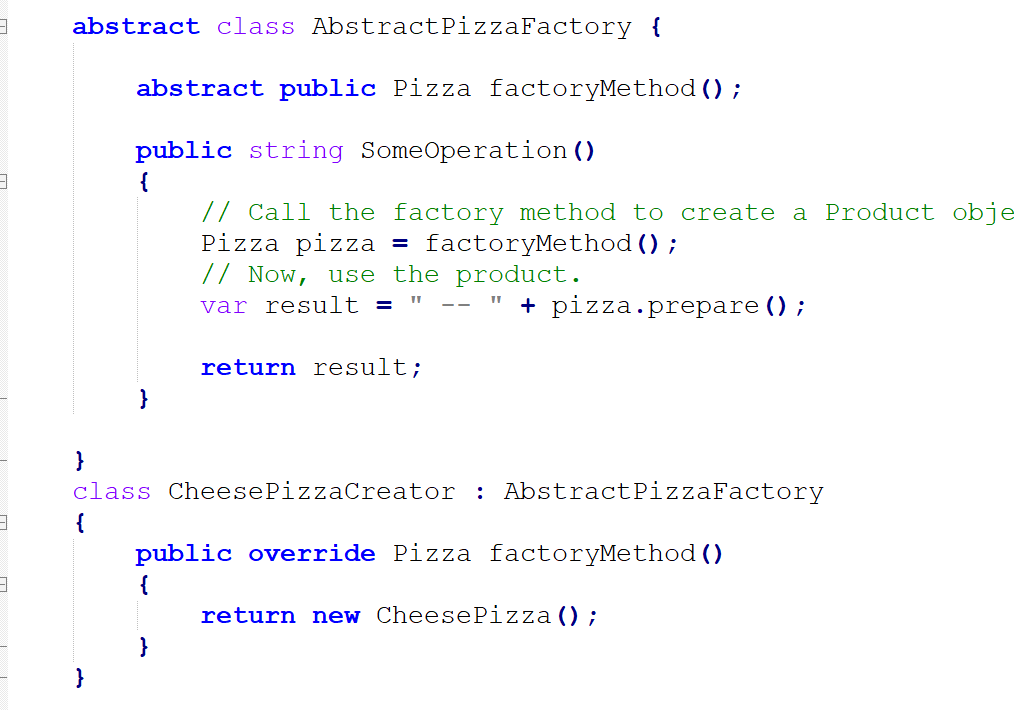
1. What is the most important thing you learnt in this module?

The most important thing I learned was these five design patterns, Singleton, Factory, Abstract Factory, Façade, Observer. I understand the usage scenarios of these five design patterns, I am familiar with the use of these five design patterns, and at the same time, my programming ability has been further improved.

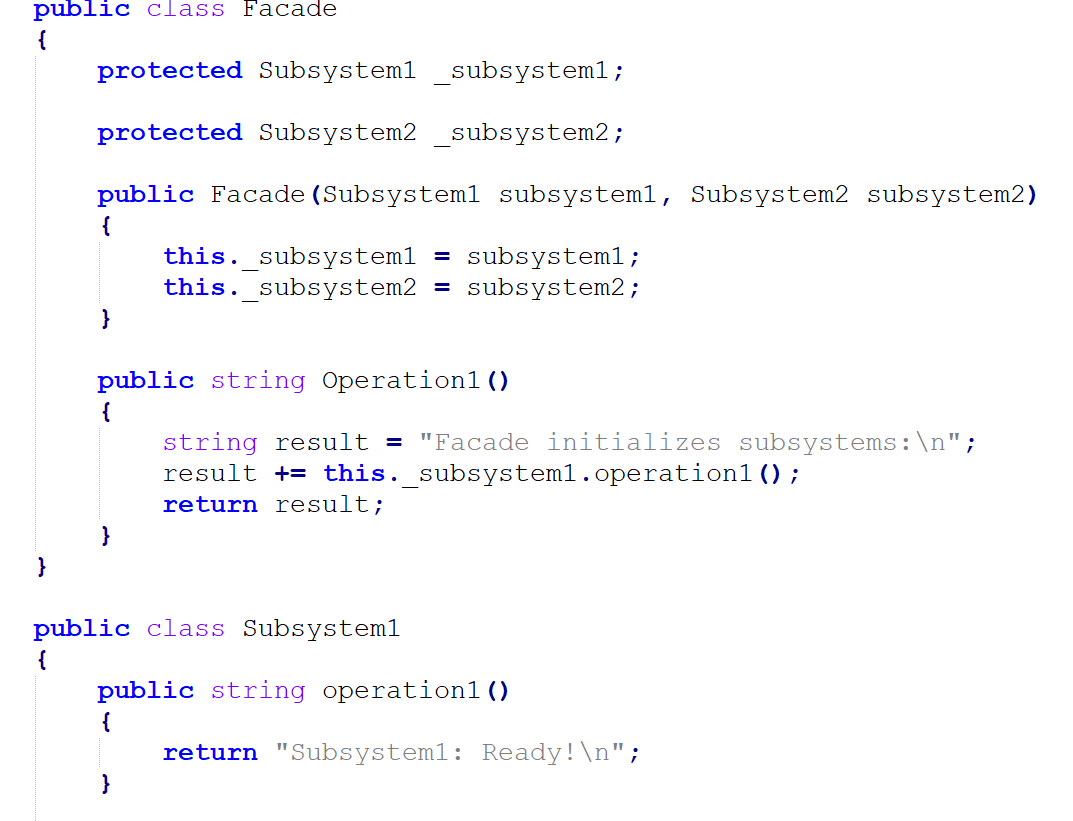
1. How does this relate to what you already know?
2. Using the Factor Menthod Pattern, you can create many classes with different requirements, and classes are the basis of object-oriented programming, so you can think more about the factory pattern when constructing classes in the future.
3. Using the singleton pattern can effectively solve the problem of global variables. When programming, you can use the singleton pattern when you need to define global variables.
4. If you want to hide the complexity of the system and provide the client with an interface through which the client can access the system, you can use the Facade pattern
5. To let some specific people receive the message, you can use the observer pattern, in which the publisher can send the message to the subscribers in the message queue.
6. Reflect on the code that was given to you in the lab. You can take the screen shot of your python code and add image or just provide the code as text in your report. A good reflection includes:



Simple factory, the class needs to be customized according to the user's input, and when the function of the class changes, the function needs to be refactored, which is more troublesome



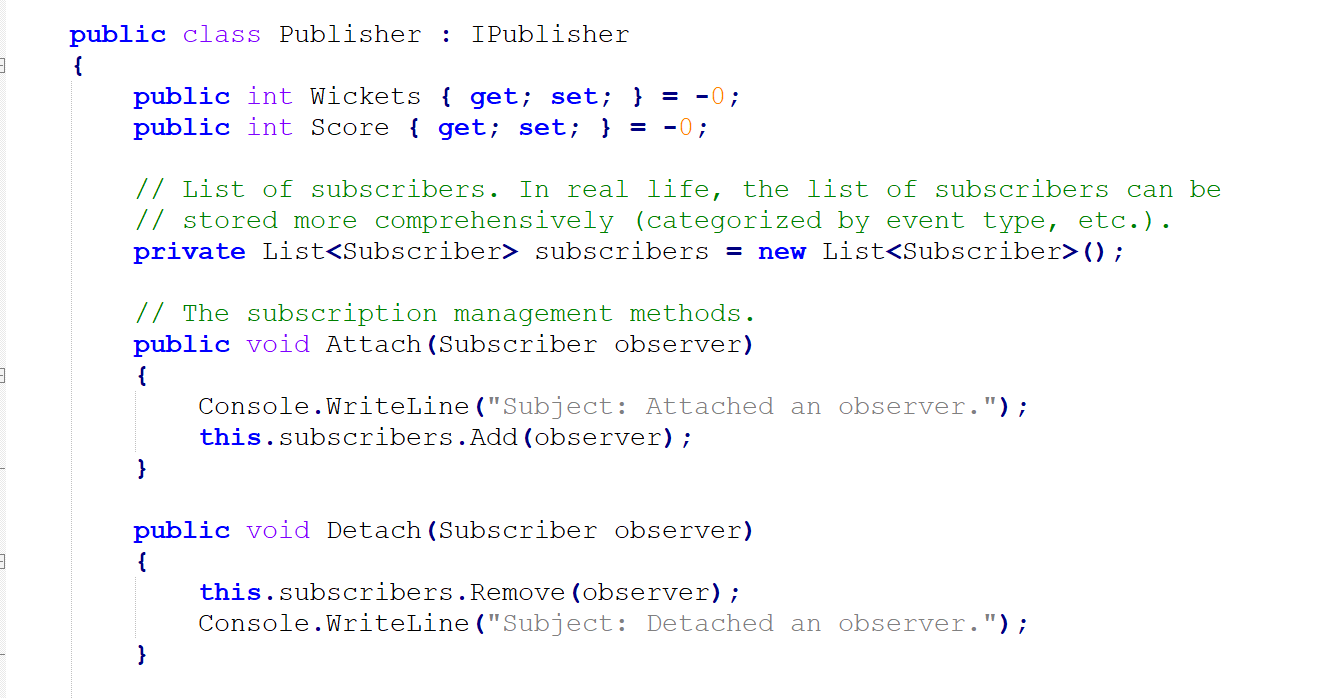
Abstract factory class, abstract classes only need to define abstract methods, which can be implemented by subclasses. The advantage is that when the number of classes increases, it is only necessary to define the increased classes and implement the methods of the parent class, and no code refactoring is required.



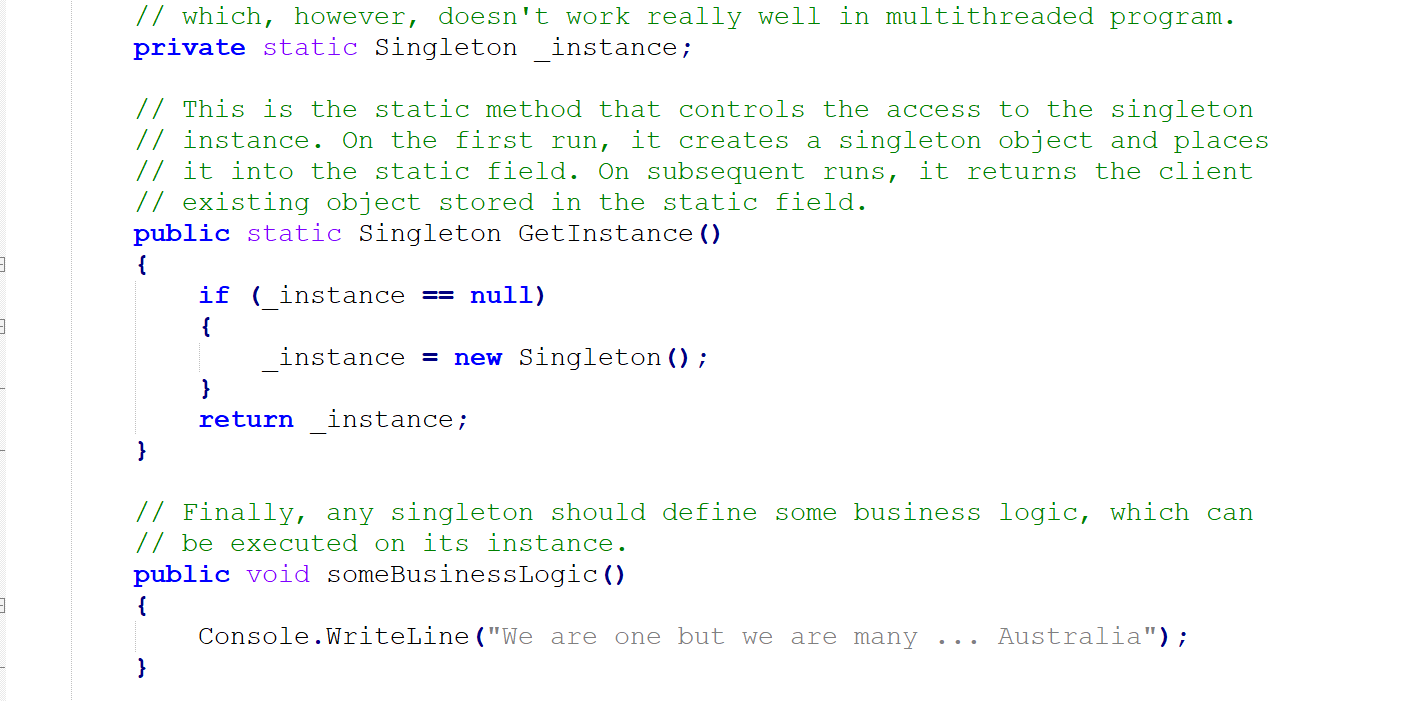
Hide complex internal operations, and open a simple interface for users to use.



Define different subscribers, which are mainly used to receive information.



Define the message sender, where the Attach function completes the addition of message receivers, the Detach function is used to delete the message receiver, and the Notify function is used to promote the message,



In the singleton mode, the most important thing is that a class can only instantiate one object. If the class is called after it is instantiated, then the last instantiated object is returned, which can be used as a global variable.

Activity 1

Discuss your understanding of design pattern by using Iterator design pattern.

Iterative mode is used to sequentially access the elements of a collection object without needing to know the underlying representation of the collection object. The iterator pattern is a behavioral pattern.

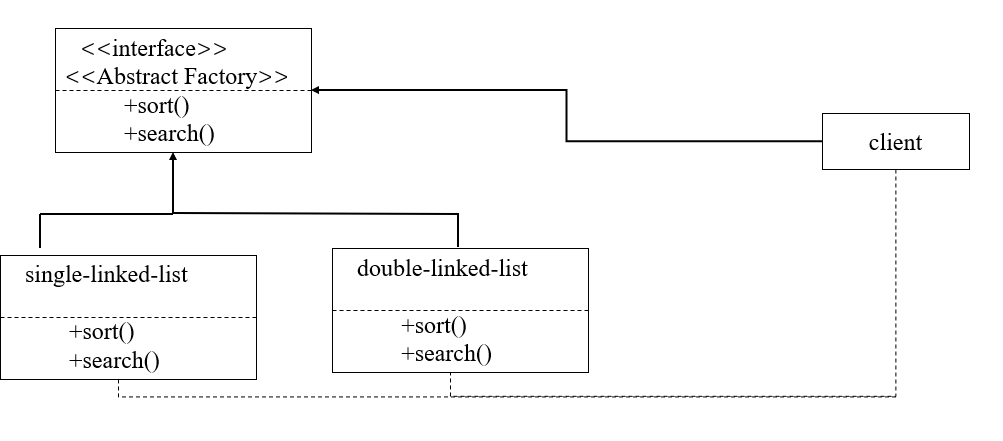
Its advantages are:

1. It supports traversing an aggregate object in different ways.
2. Iterators simplify aggregation classes.
3. There can be multiple traversals on the same aggregate.
4. In the iterator mode, it is very convenient to add new aggregate classes and iterator classes without modifying the original code.

Its disadvantages are:

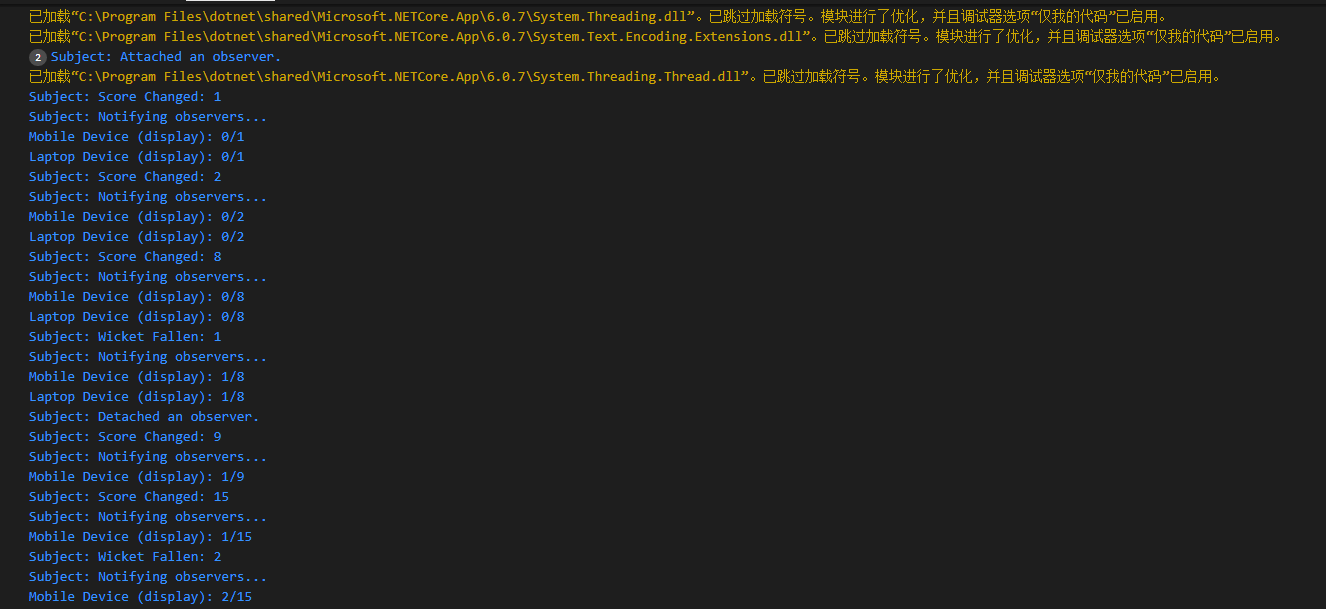
1. Access the contents of an aggregate object without exposing its internal representation.
2. You need to provide multiple ways of traversing aggregate objects.
3. You need to provide multiple ways of traversing aggregate objects.

Activity 2



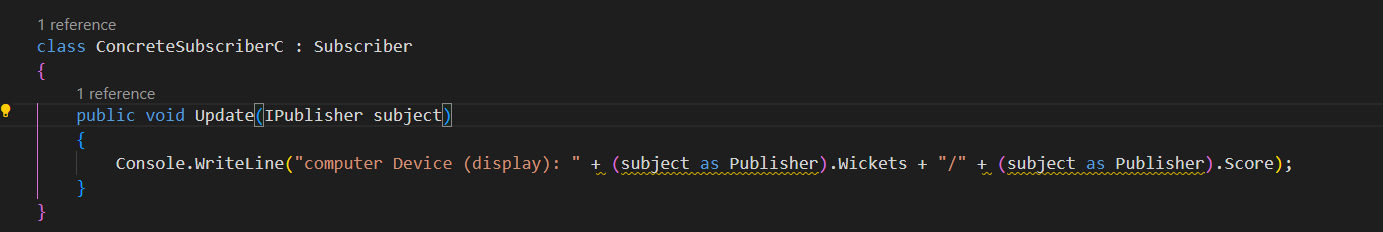
Activity 5

Have a look at the associated C# code for the observer design pattern. Demonstrate your understanding of the code by (doing some of the following activities):

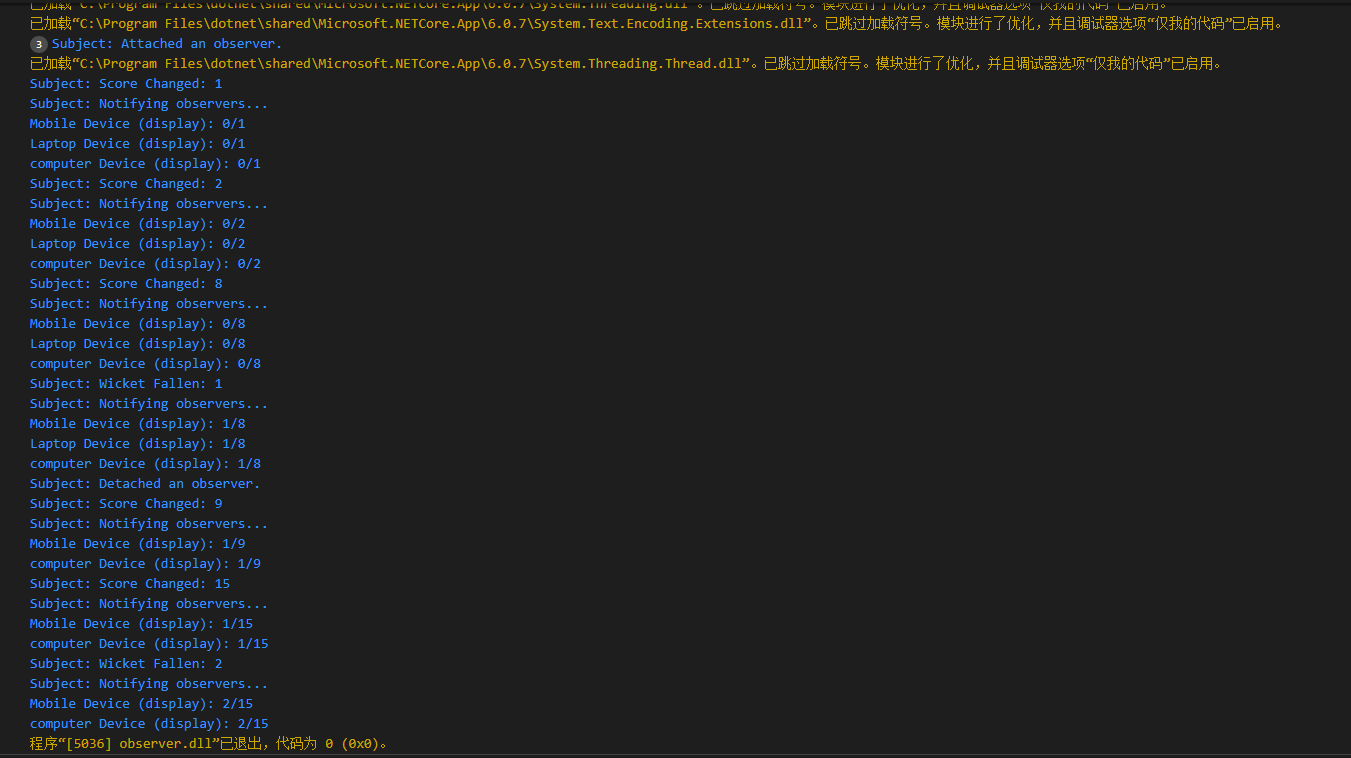


The result of running the initial code is shown in the figure. There are two subscribers and one sender in the code, where Score and Wicket represent the information that needs to be modified.

Here we add another subscriber, the code is as shown in the figure:



The result of running is as follows:



Our newly added class is computer, as can be seen from the above figure, there are three objects: Mobile , Laptop , computer.