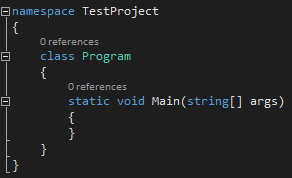
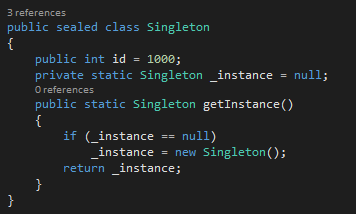
**Main program**



**Factory Pattern :**The *factory method pattern* is a [creational pattern](https://en.wikipedia.org/wiki/Creational_pattern) that uses factory methods to deal with the problem of [creating objects](https://en.wikipedia.org/wiki/Object_creation) without having to specify the exact [class](https://en.wikipedia.org/wiki/Class_(computer_programming)) of the object that will be created. This is done by creating objects by calling a factory method—either specified in an interface and implemented by child classes, or implemented in a base class and optionally overridden by derived classes—rather than by calling a [constructor](https://en.wikipedia.org/wiki/Constructor_(object-oriented_programming))."Define an interface for creating an object, but let subclasses decide which class to instantiate. The Factory method lets a class defer instantiation it uses to subclasses."

**Command Pattern :**   
In [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming), the **command pattern** is a [behavioral](https://en.wikipedia.org/wiki/Behavioral_Pattern) [design pattern](https://en.wikipedia.org/wiki/Design_pattern_(computer_science)) in which an object is used to [encapsulate](https://en.wikipedia.org/wiki/Information_Hiding) all information needed to perform an action or trigger an event at a later time.

**Façade :** Design pattern that provides an interface to a larger body of complicated code.

**Singleton :** Restricts the instantiation of the class to one object.   


**Adapter :** when you have a class that needs to utilize a particular interface, and you have a library that includes the functionality you need, but it doesn't use the interface that you require. You can achieve the reuse of that library's code by creating an Adapter class that sits between your client code, and the code that's in this library, and adapts one interface to the other. Adapters are also commonly known as **wrappers**