[Design Patterns 2](#_Toc361737960)

[Observer Pattern 2](#_Toc361737961)

[Singleton Pattern 5](#_Toc361737962)

[Provider Model for XML and SQL Pattern 5](#_Toc361737963)

[RelayCommand 5](#_Toc361737964)

[Producer Consumer Concepts 6](#_Toc361737965)

[ThreadSafeCircular Queue Concepts 10](#_Toc361737966)

[Dependency Injection 11](#_Toc361737967)

Design Patterns

Observer Pattern

Articles:

1) <http://msdn.microsoft.com/en-us/library/ff648108.aspx>

2) <http://code.msdn.microsoft.com/windowsdesktop/ObserverPublisher-bfa78bbb>

Code Samples from the above articles -

See example E:\04 - Net Technical Material\!Visual Studio 2010 Projects\05-Design Patterns\01 - Observer - Publisher Subsciber\00-Observer\_Samples

|  |  |
| --- | --- |
| **E:\04 - Net Technical Material\!Visual Studio 2010 Projects\05-Design Patterns\02 - Observer - Publisher Subsciber\aa\ClassDiagram1.png** | **C:\Users\John\Desktop\IC142826.gif** |

|  |  |
| --- | --- |
| **E:\04 - Net Technical Material\!Visual Studio 2010 Projects\05-Design Patterns\02 - Observer - Publisher Subsciber\03 - Observable_Ex_3\aClassDiagram.png** | **C:\Users\John\Desktop\observer1.gif** |

Singleton Pattern

Following articles are perfect for this answer –

Article 1 – <http://code.msdn.microsoft.com/Singleton-Pattern-24a1da7f/view/Reviews>  
Article 2 – <http://www.dotnetperls.com/singleton>  
Article 3 - <http://csharpindepth.com/Articles/General/Singleton.aspx>

Code Example –

Issues with Singleton Pattern - <http://msdn.microsoft.com/en-us/magazine/jj991965.aspx>

Provider Model for XML and SQL Pattern

Code Sample - <http://code.msdn.microsoft.com/windowsdesktop/Provider-Model-implementati-1841debb>

RelayCommand

Difference between RoutedCommand and RelayCommand - <http://stackoverflow.com/questions/650010/mvvm-routed-and-relay-command>

ICommand Interface - <http://msdn.microsoft.com/en-us/library/system.windows.input.icommand.aspx>

Producer Consumer Concepts

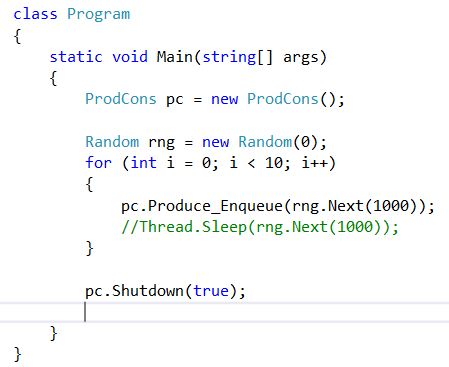
Links - <http://stackoverflow.com/questions/2252259/queuet-dequeue-returns-null>  
<http://stackoverflow.com/questions/3100166/purpose-of-monitor-pulse>  
<http://msdn.microsoft.com/en-us/library/aa645740(v=vs.71).aspx#vcwlkthreadingtutorialexample2synchronizing>  
General Reading - <http://social.msdn.microsoft.com/forums/en-US/csharpgeneral/thread/b32d62c7-7c24-41bc-8f2d-62ce6671a9f8>  
Writing Producer/Consumer using Linked List - <http://csharptest.net/450/writing-a-lockless-queue-for-a-single-producerconsumer-part-1/>

Blockingcollection - <http://www.codethinked.com/blockingcollection-and-iproducerconsumercollection>

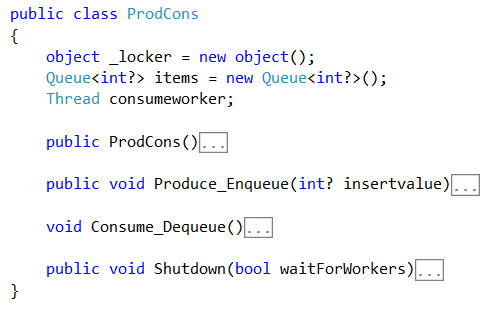
Solution Implementing Producer Consumer leveraging regular Synchronization Methods

|  |
| --- |
| **Output** |
|  |

Main Class

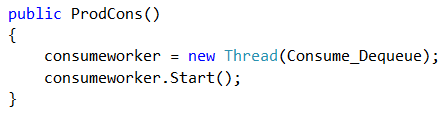


Producer Consumer Class

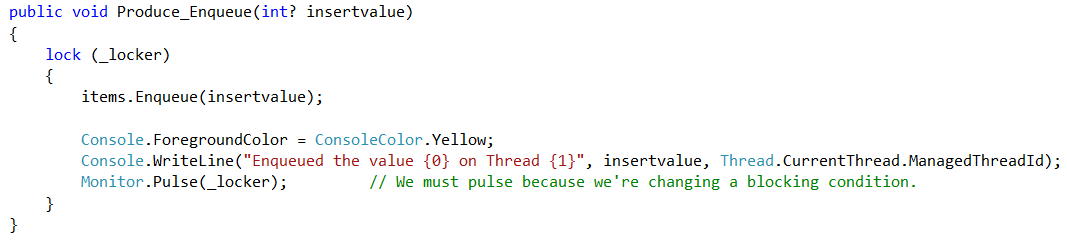


Expanded View

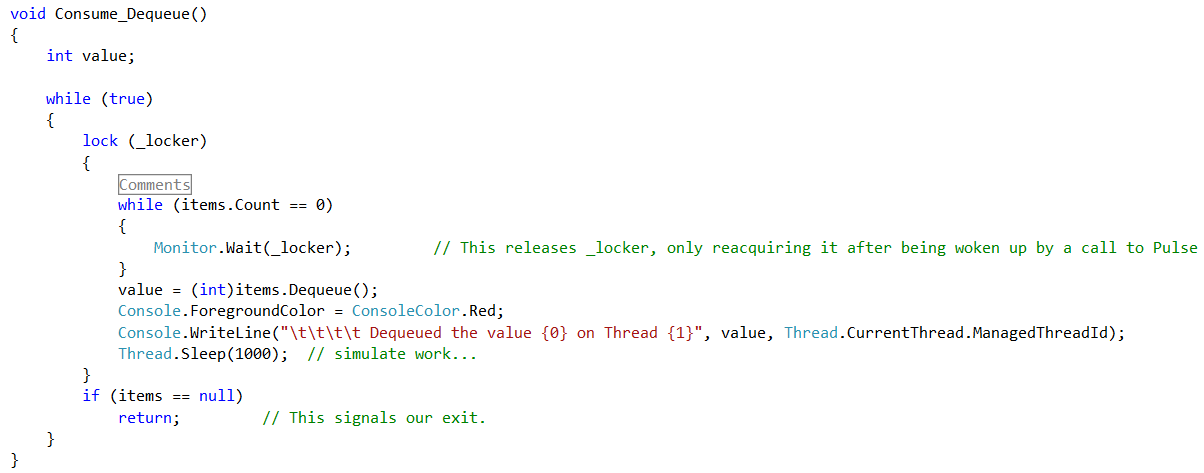
Constructer



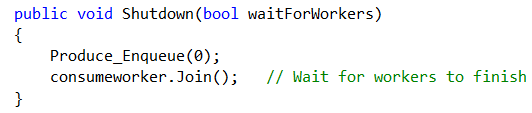
Enqueue Method



Deqeue Method



Shutodown Method



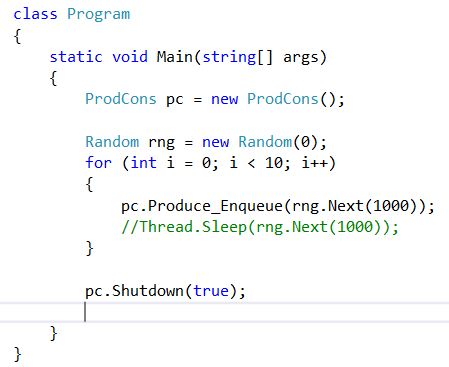
**-----------------------------------------------------------**

**Using .Net Lock Free Data Structure**

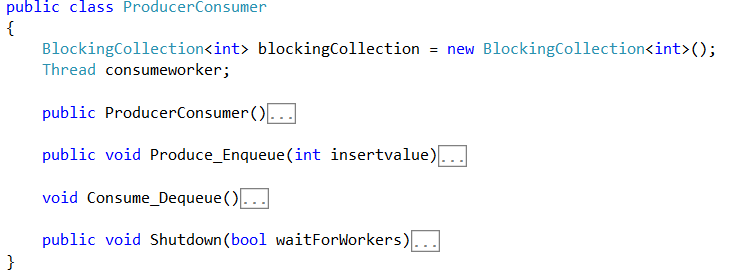
Solution Implementing Producer Consumer leveraging regular Synchronization Methods

|  |
| --- |
| **Output** |
|  |

Main Class

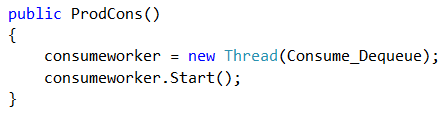


Producer Consumer Class

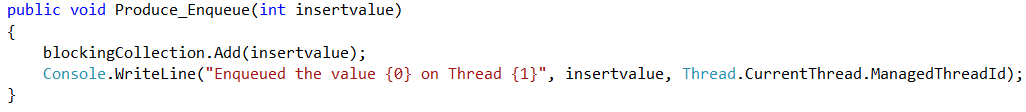


Expanded View (exact same)

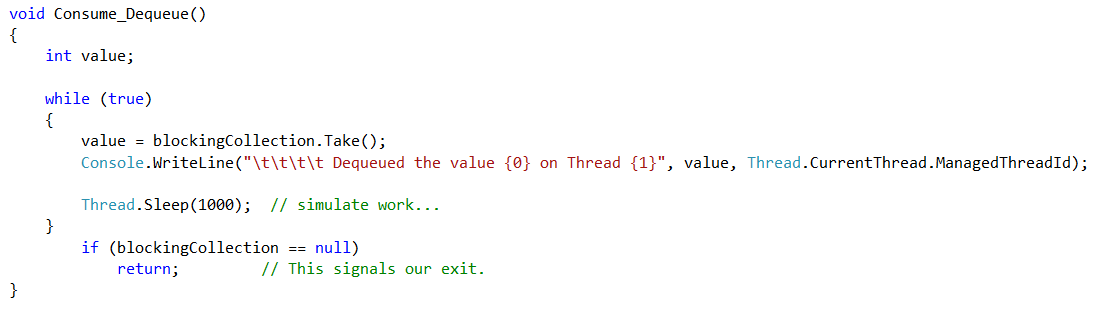
Constructer



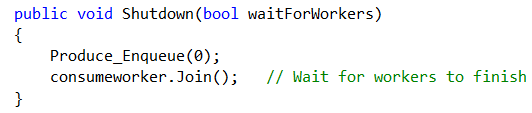
Enqueue Method



Deqeue Method



Shutodown Method (exact same)



ThreadSafeCircular Queue Concepts

* You initialize 4 things -> head pointer, tail pointer, size of int[] and private syncroot object
* Methods are pretty straightforward – main logic will happen in Enqueue and Dequeue methods.

|  |
| --- |
|  |
|  |
|  |
| head  tail |

* Initialize state of variables will be as follows – both head and tail will be -1

Enqueue Method logic checks

Step 1: Check if the queue is full (2 conditions when this can happen)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| tail   |  | | --- | | 4 | | 3 | | 2  head | | 1 | | |  | | --- | | 4 | | 3  head | | 2  tail | | 5 | |

Step 2: (where to move the tail pointer)

Check if tail is at last spot (implicit understanding is also that head is not in [0] position because its either -1 or [2]

If yes : bring tail back to [0] or increment it

tail

|  |
| --- |
| 4  tail |
| 3  tail |
| 2 |
|  |

|  |
| --- |
|  |
| 3 |
| 2 |
| 1 |

Dequeue Method logic checks

Step 1: Check if the queue is empty

|  |
| --- |
|  |
|  |
|  |
| head |

Step 2:

1. Pop the value where head currently resides (use head as index in queue) and assign it to value
2. Set the value over there as 0 (null)

Step 2a: (where to move the head pointer)

If head and tail are at the same position

1. Set both head and tail as -1
2. If head at top position – then bring it back to bottom – index[0]

head

|  |
| --- |
| 4 |
| 3 |
| 2 |
| 5 |

1. Else just move the head pointer

|  |
| --- |
| 4  head |
| 3  head |
| 2 |
| 1 |

Dependency Injection

<http://www.blackwasp.co.uk/DependencyInjection.aspx>

<http://www.dotnet-tricks.com/Tutorial/dependencyinjection/67LX120413-Implementation-of-Dependency-Injection-Pattern-in-C>

<http://blog.efvincent.com/practical-di-101/>