Two Generic parameters

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

public interface IDocument {
string Title {
get;
}

string Content {
get;
}
}

public class Document : IDocument {
private string title;
public string Title {
get {
return title;
}
}

private string content;
public string Content {
get {
return content;
}
}

public Document(string title, string content) {
this.title = title;
this.content = content;
}
}
```

```
public class ProcessDocuments
where T : IDocument
where U : IDocumentManager {
public static void Start(U dm) {
new Thread(new ThreadStart(new
ProcessDocuments(dm).Run)).Start();
}

protected ProcessDocuments(U dm) {
documentManager = dm;
}

private U documentManager;

protected void Run() {
while (true) {
if (documentManager.IsDocumentAvailable) {
T doc = documentManager.GetDocument();
Console.WriteLine(“Processing document {0}”, doc.Title);
}
Thread.Sleep(new Random().Next(20));
}
}
}

public interface IDocumentManager {
void AddDocument(T doc);
T GetDocument();
bool IsDocumentAvailable {
get;
}
}

public class DocumentManager : IDocumentManager {
private readonly Queue documentQueue = new Queue();

public void AddDocument(T doc) {
lock (this) {
documentQueue.Enqueue(doc);
```

```
}
}

public T GetDocument() {
T doc = default(T);
lock (this) {
doc = documentQueue.Dequeue();
}
return doc;
}

public bool IsDocumentAvailable {
get {
return (documentQueue.Count > 0) ? true : false;
}
}
}
class Program {

static void Main(string[] args) {
DocumentManager dm = new DocumentManager();
ProcessDocuments.Start(dm);
for (int i = 0; i < 1000; i++) { Document doc = new Document("Doc " + i.ToString(), "content"); dm.AddDocument(doc); Console.WriteLine("added document {0}", doc.Title); Thread.Sleep(new Random().Next(20)); } } }
```
[/csharp]