Let's create a UML design for a simple application that uses the Proxy design pattern and then provide an implementation in C#. The Proxy pattern provides a surrogate or placeholder for another object to control access to it.

**Scenario:** Let's say we have a Document object which can display its content. However, we don't always want to load the content until it's actually needed (to save memory or bandwidth for instance). We'll use a Proxy to control the access to this Document.

**UML:**

1. AbstractDocument: An abstract class.
   * DisplayContent(): Abstract method.
2. RealDocument: Inherits from AbstractDocument.
   * content: Private attribute.
   * DisplayContent(): Override. Displays the content.
3. DocumentProxy: Inherits from AbstractDocument.
   * realDocument: Private attribute (instance of RealDocument).
   * DisplayContent(): Override. Ensures that the realDocument instance is created and then displays the content.

**C# Implementation:**

csharp

// AbstractDocument.cs

public abstract class AbstractDocument

{

public abstract void DisplayContent();

}

// RealDocument.cs

public class RealDocument : AbstractDocument

{

private string content;

public RealDocument(string content)

{

this.content = content;

// Simulating a delay, like loading from a database or a file

System.Threading.Thread.Sleep(2000);

}

public override void DisplayContent()

{

Console.WriteLine(content);

}

}

// DocumentProxy.cs

public class DocumentProxy : AbstractDocument

{

private RealDocument realDocument;

private string content;

public DocumentProxy(string content)

{

this.content = content;

}

public override void DisplayContent()

{

// Lazy instantiation

if (realDocument == null)

{

realDocument = new RealDocument(content);

}

realDocument.DisplayContent();

}

}

// Main program

class Program

{

static void Main()

{

AbstractDocument document = new DocumentProxy("Hello from the Proxy Pattern!");

// The content will be loaded and displayed only when the following line is executed

document.DisplayContent();

}

}

When you run the program, there will be a simulated delay (due to Thread.Sleep) the first time you call DisplayContent on the proxy, representing the time to load the content. The Proxy pattern is used here to lazily instantiate the RealDocument and delay this loading until it's actually necessary.