Examination #4

Name:_____ **Instructor's Solution** _____ SUID:_____

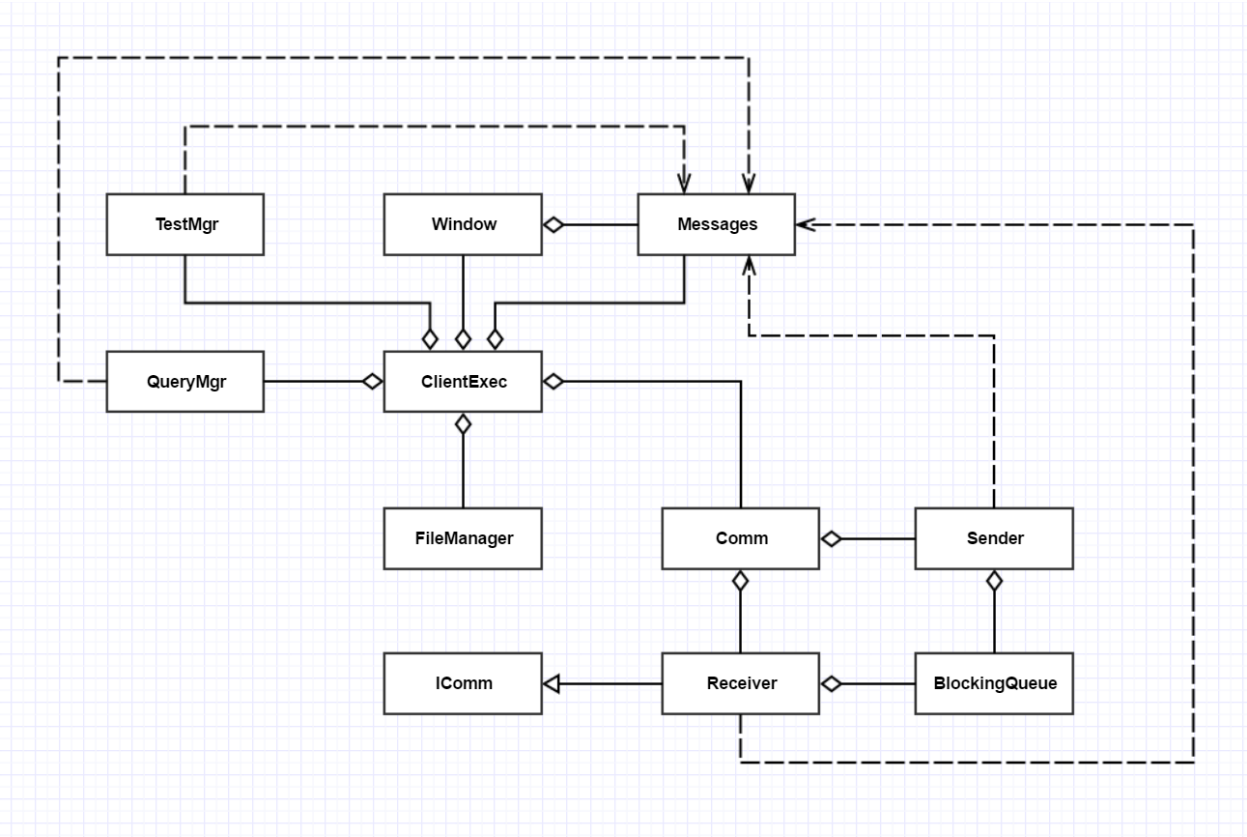
This is a closed book examination. Please place all your books on the floor beside you. You may keep one page of notes on your desktop in addition to this exam package. All examinations will be collected promptly at the end of the class period. Please be prepared to quickly hand in your examination at that time.

If you have any questions, please do not leave your seat. Raise your hand and I will come to your desk to discuss your question. I will answer all questions about the meaning of the wording of any question. I may choose not to answer other questions.

You will find it helpful to review all questions before beginning. All questions are given equal weight for grading, but not all questions have the same difficulty. Therefore, it is very much to your advantage to answer first those questions you believe to be easiest.

1. Draw a class diagram for the Client of Project #4.

   Answer:



2. Write all the code[1] for a Message class that has Name, MessageType, TimeStamp, DestinationUrl, SourceUrl, and body. These may all be strings except for the TimeStamp property, which should be a DateTime type. Show the XML you will use to pass a list of files in the body of the message. Finally, show how to construct the XML using the XDocument class.

   Answer:

```
public class Message
{
  public string Name { get; set; }
  public string MessageType { get; set; }
  public DateTime TimeStamp { get; set; }
  public string DestinationUrl { get; set; }
  public string SourceUrl { get; set; }
  public string body { get; set; }
}

class MT4Q2
{
  public string fileListBody(List<string> fileNames)
  {
    XDocument doc = new XDocument();
    XElement root = new XElement("files");
    foreach (string name in fileNames)
    {
      XElement fileElem = new XElement("file");
      fileElem.Value = name;
      root.Add(fileElem);
    }
    doc.Add(root);
    return doc.ToString();
  }

  static void Main(string[] args)
  {
    Console.Write("\n  MT4Q2 - Message Class");
    Console.Write("\n ======================\n");

    MT4Q2 demo = new MT4Q2();
    Message msg = new Message();
    msg.Name = "Test Message";
    msg.MessageType = "files";
    msg.TimeStamp = DateTime.Now;
    msg.DestinationUrl = "http://localhost:8080/ISomeContract";
    msg.SourceUrl = "http://localhost:8081/ISomeContract";
    msg.body = demo.fileListBody(new List<string> { "file1", "file2", "file3" });

    msg.showMessage();

    Console.Write("\n");
  }
}
```

```
<files>
  <file>file1</file>
  <file>file2</file>
  <file>file3</file>
</files>
```

[1] You don't need to write show or ToString functions.

3. When and why would you choose to test code in a Test Harness instead of simply using Visual Studio? How does the code have to be configured to test in our Project #4 TestHarness?

   Answer:

   A TestHarness is a tool that automates test execution in an environment containing a lot of baseline code. Building the test is a manual effort, done once, while test execution may happen many times.

   Using a TestHarness requires:
   - building a testdriver that inherits from the ITest interface
   - building a TestRequest message that cites the test driver and the code to be tested
   - loading code libraries from a Repository for all dependencies
   - using a client and the TestHarness server

   It will always take more time to define and execute a test once in a TestHarness than in an IDE like Visual Studio. You would not do that unless the same test had to be executed many times.

   This is exactly what happens when code, that your tested code depends on, changes. Every time a dependency changes you must retest the depending code. That will happen many times during development of a big project.

   With a test harness you just send an existing TestRequest message, re-executing the existing test to see if lower level change(s) broke your code's operations. This assumes that you saved the TestRequest message in the Repository when it was first constructed.

4. Write a code fragment to define a lambda and bind to a delegate, where the lambda captures a string message and writes that, when invoked, to the System.Console.

   Answer:

```
class MT4Q4
{
  static void Main(string[] args)
  {
    Console.Write("\n  MT4Q4 - Define Lambda");
    Console.Write("\n ======================\n");

     Action<string> act = (string msg) => {
       Console.Write("\n  a message: \"{0}\"", msg);
    };
    act.Invoke("Hello");

    Console.Write("\n\n");
  }
}
```

5. What is the purpose of an Operational Concept Document (OCD) and what are its principal parts?

Answer:

The purpose of an OCD is to disclose a Project's concept and guide its developers during design and implementation. OCDs are often used to inform stakeholders of the project goals and risks in order to acquire resources to continue development.

It's most important role is to provide an invariant description of the essential features and parts of the system in an environment where detailed requirements and implementation are subject to change, and to focus attention on important risks.

The concept should define:

- The intended users and their uses of the system, defining the benefits of each intended use and impact on system design to support that use.
- The structure[2] of the system in terms of its packages and their responsibilities and communication flows and activities.
- Critical issues and explore their consequences and proposed solutions.

Note that the OCD does not attempt to capture the detailed requirements of the Project. It is concerned with concept, not the details of implementation and testing.

[2] The emphasis here is on structure, not on the types of diagrams used to disclose the structure. Those are usually package and activity diagrams, but could also be class and state diagrams, and other kinds of ad-hoc diagrams.

6. WCF allows the construction of a proxy for a service that may not yet be running. When the first call is made to the service, using the proxy, it fails with an exception if the service is not available. Write a wrapper class for the proxy that retries the first call a finite number of times with brief waits between calls if the call initially fails.

   Answer:

   Main issue here is that service methods may have different signatures, so must use a lambda:

```
class ReTry  // answer starts here
{
  public int numRetries { get; set; } = 10;
  public int waitMilliSec { get; set; } = 50;

  public void doWithReTry(Action action)
  {
    for (int i = 0; i < numRetries; ++i)
    {
      try
      {
        action.Invoke();
        break;
      }
      catch
      {
        Thread.Sleep(waitMilliSec);
        Console.Write("\n  proxy call failed");
        Console.Write("\n  retrying");
      }
    }
  }
}   // answer ends here
public class Proxy
{
  static int count = 0;
  public void someProxyMethod(string msg)
  {
    if (++count < 4)
      throw new Exception("proxy call failed");
    Console.Write("\n  proxy call succeeded with message \"{0}\"", msg);
  }
}
class MT4Q6
{
  static void Main(string[] args)
  {
    Console.Write("\n  MT4Q4 - Retry Proxy");
    Console.Write("\n ====================\n");
    ReTry reTry = new ReTry();
    Proxy proxy = new Proxy();
    string msg = "Hello from MT4Q4";
    reTry.doWithReTry(() => { proxy.someProxyMethod(msg); });

    Console.Write("\n\n");
  }
}
```

7. What is the purpose of the System.Type class? Write a small fragment of code to show one way to use it with an object. Repeat for use with a C# class.

Answer:

The Type class is a container for information gathered about an object or class using reflection.

Use with object:

```
MT3Q3 mt3Q3 = new MT3Q3();
Type t = mt3Q3.GetType();
Console.Write("\n  name of this class is {0}", t.Name);
Console.Write("\n  namesapce is {0}", t.Namespace);
```

Use with class:

```
Type t = typeof(MT3Q3);
Console.Write("\n  name of this class is {0}", t.Name);
Console.Write("\n  namesapce is {0}", t.Namespace);
```

See MT3Q3.cs for code using these fragments.