Examination #2

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. What is a generic type? Pick some .Net Framework generic type and show how it is used to do one simple operation (you get to decide what operation).

   Answer:

   A generic type is a reference type that depends on one or more parameters, specified when an application uses the type.

   Func<U1, U2, …, V> is a generic delegate type that accepts an arbitrary finite number of parameter types for arguments and a final type that specifies the type of the returned object or value.

   Delegates are always bound to functions with a specified signature, and the parameters, above, define that signature.

   Here is an example of the Func delegate used to attempt to open a FileStream for a file with name "filename".

```
Func<string, FileStream> openStream = (string fileName) =>
{
  return new FileStream(fileName, FileMode.Open);
};
```

2. Write all the code for a sender thread that dequeues a message from a BlockingQueue<Message> and sends it, via WCF, to a destination defined by the Message. Please write code for the processing and for starting the thread. You may assume that the thread has access to an instance of a proxy object and is able to create a new instance as needed.

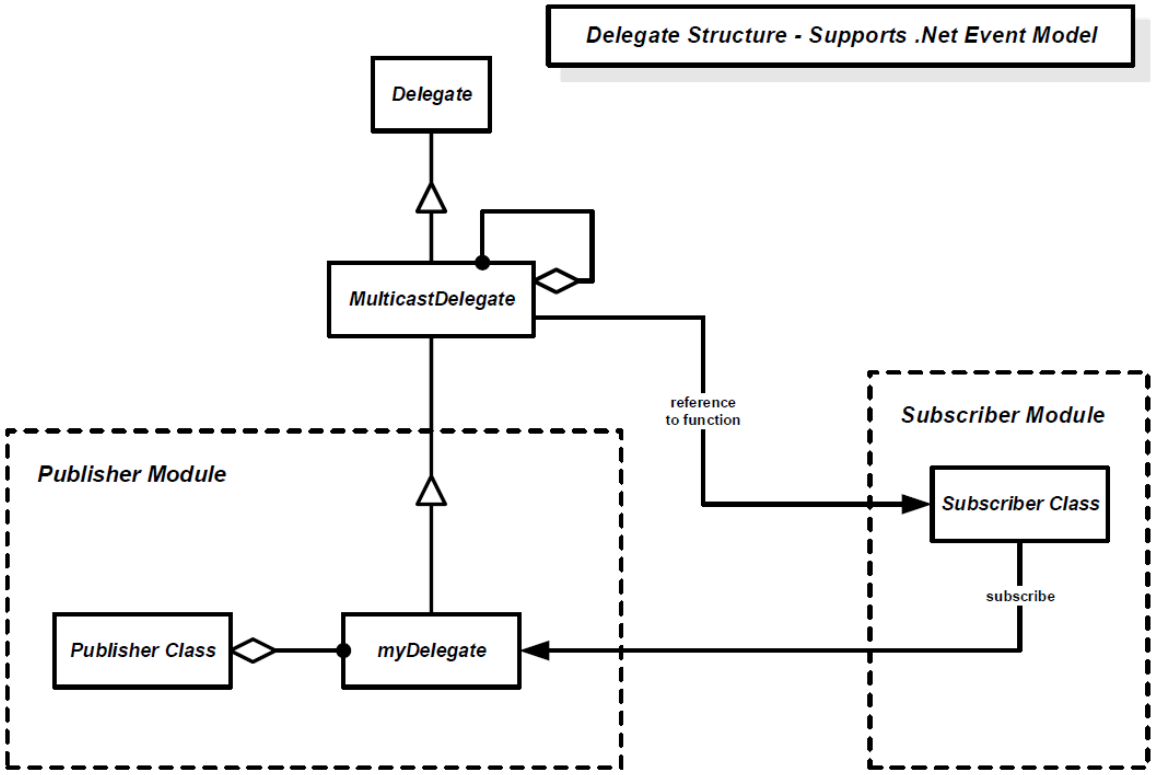
Answer:

```
// Pseudo Code:
//
// ThreadProc() {
//   while(true)
//     deQ msg from BlockingQueue<Message>
//     if(msg.to != lastUrl) {
//       lastUrl = msg.to
//       proxy = CreateProxy(msg.to)
//     }
//     proxy.PostMessage(msg)
//   }
// }
//
// Thread t = new Thread(ThreadProc)
// t.start()
```

```
//----< processing for send thread >---------------------

void ThreadProc()
{
  tryCount = 0;
  while (true)   // loop to collect messages
  {
    Message msg = sndBlockingQ.deQ();
    if(msg.to != currEndpoint)
    {
      currEndpoint = msg.to;
      CreateSendChannel(currEndpoint);
    }
    while (true)   // loop trying to send - this looping not required by question
    {
      try
      {
        channel.PostMessage(msg);
        Console.Write("\n  posted message from {0} to {1}", name, msg.to);
        tryCount = 0;
        break;
      }
      catch (Exception ex)
      {
        Console.Write("\n  connection failed");
        if (++tryCount < MaxCount)
          Thread.Sleep(100);
        else
        {
          Console.Write("\n  {0}", "can't connect\n");
          currEndpoint = "";
          tryCount = 0;
          break;
        }
      }
    }
    if (msg.body == "quit")
      break;
  }
}
//----< initialize Sender >------------------------------------

public Sender()
{
  sndBlockingQ = new BlockingQueue<Message>();
  sndThrd = new Thread(ThreadProc);
  sndThrd.IsBackground = true;
  sndThrd.Start();
}
```

3. Describe the structure[1] of a delegate you might use in an application. Show how to declare the delegate type, an instance of the delegate, how to subscribe for delegate notifications, and how to invoke the delegate.

Answer:

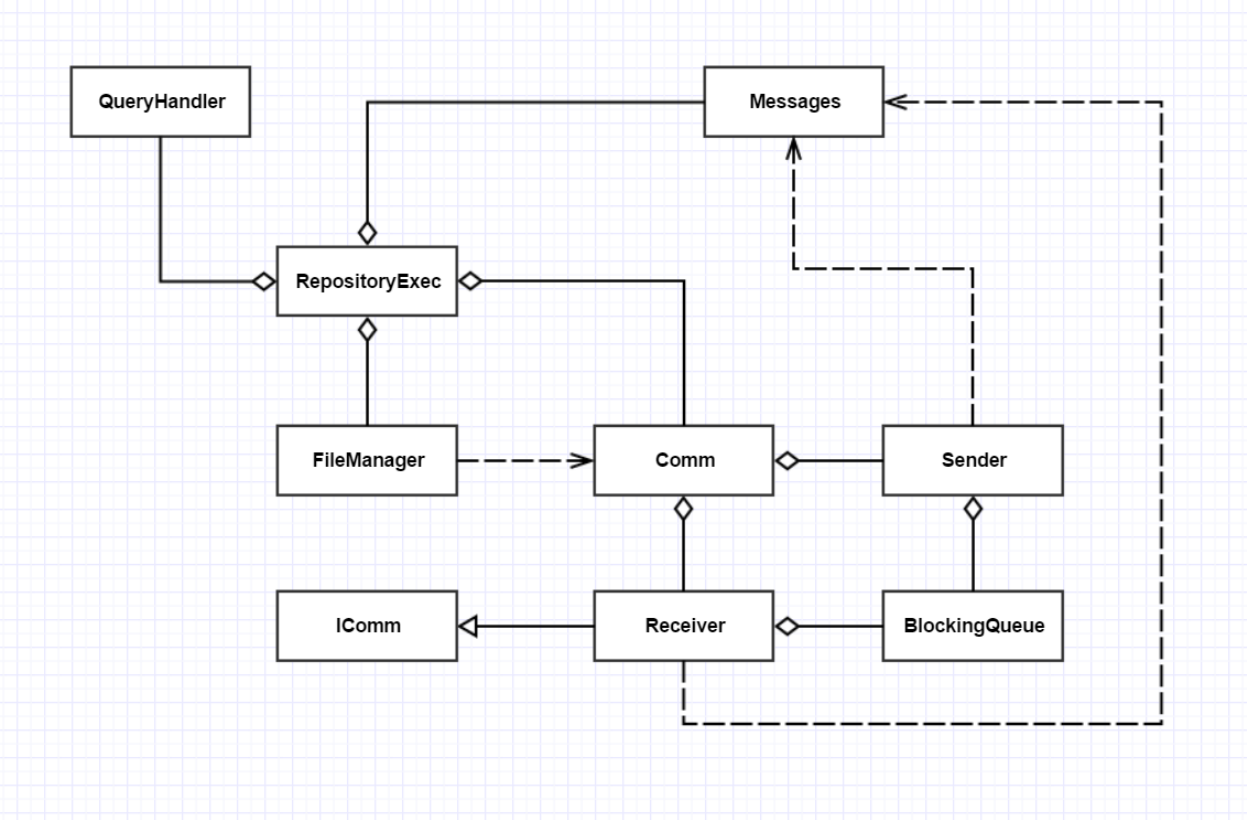



```
delegate void SimpleTraditional(string msg);  // decl delegate Type

void handler(string msg)
{
  Console.Write("\n  {0}",msg);
}

static void Main(string[] args)
{
  DemoDelegates pr = new DemoDelegates();
  SimpleTraditional trad = new SimpleTraditional(pr.handler);  // decl inst and subsc
  trad += new SimpleTraditional(pr.handler);  // subsc again
  trad += new SimpleTraditional(pr.handler);  // subsc again
  trad.Invoke("traditional message");         // invocation
}
```

[1] By structure we mean the classes, used to define the delegate, and their relationships.

4. Draw a class diagram for your design of Project #4 Repository.

   Answer:



I've included ServiceHost in the Receiver because that's the way my CommChannelDemo was configured. Project #4 didn't require you to have a logger, so that's not included here either. You didn't lose points for having them.

Note:
- You should not include .net Framework classes, as the class diagram describes your design.
- Every class except the Executive should have an owner.
- It's important to show using relationships, e.g., interactions with classes not owned by the user.

5. Define the Uses and Users for Project #4, as you would for an Operational Concept Document (OCD)? Please fit your entire answer on this page.

Answer:
There are two kinds of answers. One assumes that the TestHarness is used in the context of this course, e.g., a learning tool and a means to evaluate student progress. The other assumes that the TestHarness will be used in the context of a Software Development Federation. I'll give both kinds of answer:

Student Project Users of Repository:

a. **Student** in CSE681 – Software Modeling and Analysis.
Uses the project to learn how to structure complex code, handle threading problems, and use the WCF framework to pass messages and WPF framework to develop a GUI.
Impact on Design:
Will provide a console along with the GUI window and in the TestHarness to demonstrate requirements. Will design a message catalog with the smallest number of messages needed to meet all the requirements. Will structure the design based on demonstration code provided in Project4HelpF16 (if they want to finish on time).

b. **TA grader:**
Uses the project to evaluate student performance, e.g., check requirements, evaluate structure and design, and analyze code metrics.
Impact on Design:
Students pay particular attention to demonstration and code structure.

c. **Instructor:**
Will examine code structure, look over demonstrations as recorded by TA grader, and look at code documentation and structure.
Impact on Design Process:
Students come to help sessions and ask questions. They test the submission in a new empty directory. They examine each package for proper documentation.

Federation Users of Repository:

a. **Developer:**
Uses TestHarness to verify code as part of check-in process and defect analysis.
Impact on Design:
This work is in the main flow of software development and it is very important to make it as easy to use and productive as is practical. We do this by focusing on the Client interactions with a well designed GUI, by making the TestHarness as efficient as we can, and by automating generation of messages by selecting items on the client GUI.

b. **Quality Assurance:**
QA personnel focus on the testing process in preparation for acceptance testing and on maintaining code quality.
Impact on Design:
QA will be submitting test request messages for large bodies of code. It is important that the collection of dependent code packages be automated and that the TestHarness operates as efficiently as possible. It is also important that we be able to scale out by adding new TestHarness servers without a lot of rework to the existing Federation infrastructure.

c. **Managers:**
Managers will need summary level information about testing.
Impact on Design:
It is important to build in easy to use query mechanisms that can select results to summarize testing and to build effective displays of that information.

d. **Client, Repository, Build Servers:**
Federated servers need to collaborate with other software.
Impact on Design:
Need common communication protocols and message structures. Error handling must not interrupt information flow.

6. Show how, using reflection, you can discover if an instance of some type implements a specified interface.

   Answer:

```
Type[] types = assem.GetExportedTypes();

foreach (Type t in types)
{
  if (t.IsClass && typeof(ITest).IsAssignableFrom(t))  // this is the most important
  // does this type derive from ITest ?                // part of the answer
  {
    try
    {
      testDriverName = file;
      tdr = (ITest)Activator.CreateInstance(t);
      Console.Write(
        "\n    TID" + Thread.CurrentThread.ManagedThreadId + ": "
        + testDriverName + " implements ITest interface - Req #4"
      );
    }
    catch
    {
      continue;
    }
  }
}
```

I normally don't ask a question that requires you to remember some detailed syntax. However, this is at the core of everything we are doing in the TestHarness and we have looked at this code several times in my various demos.

7. When writing code for a Windows Presentation Foundation (WPF) application, why might you need to use System.Threading.Dispatcher. Write a code fragment that shows how you would do that. You may need to define a simple function with at least one argument for this code fragment.

   Answer:

   A Window class will often create a child thread to do a long-running task, to avoid missing UI events. The child thread is not allowed to directly insert it's results into the Window controls, a listbox for example. In order to do the insertion of results, the child thread has to dispatch, to the main UI thread, a delegate bound to a function that does the required insertion.

   Here's how you do that:

```
void addFile(string file)
{
  listBox1.Items.Insert(0,file);
}

string[] files = System.IO.Directory.GetFiles(path, pattern);
foreach (string file in files)
{
  if (Dispatcher.CheckAccess())
    addFile(file);
  else
    Dispatcher.Invoke(
      new Action<string>(addFile),
      System.Windows.Threading.DispatcherPriority.Background,
      new string[] { file }
    );
}
```