Examination #3

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. Explain the differences between a C# reference type and a C# value type.

   Answer:

   Value types are copy-able and assign-able. They live in the memory allocated to instances of the type where they are defined or in the stackframe of a method defining them locally. Value types become invalid when their container is disposed.

   Value types are primitive .Net types, e.g., ints, doubles, … and also structs.

   Reference types have copy-able and assign-able handles, but the instances, which live on the managed heap, are not copy-able nor assign-able.

   Reference types are all Library and User-defined types created with classes or delegates using new keyword. They remain valid until all references are no longer in scope or set to null, or they are disposed.

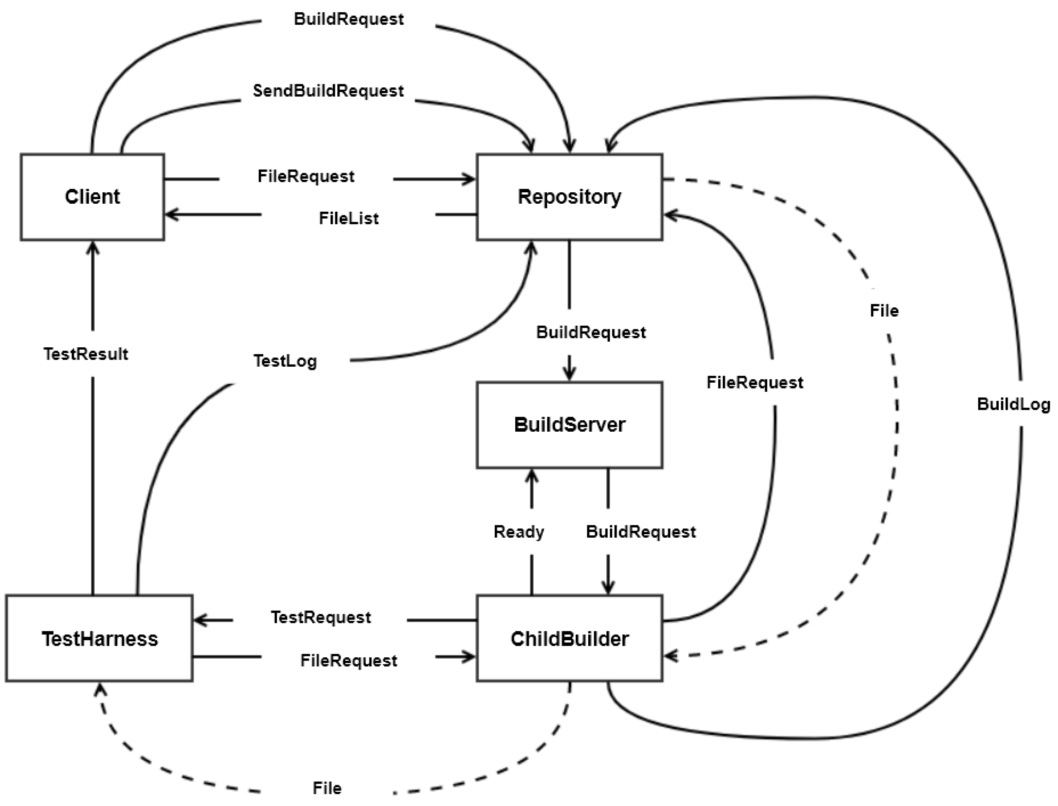
2. Write all the code for a Repository thread that dequeues a Client's request message for a list of files, and processes the message then returns a reply message with the file list requested. You may assume an instance of Comm with the same interface you used for Project #3 and the same CommMessage structure.

Answer:

```
PartialMockRepo()
{
  Thread t = new Thread(threadProc);
  t.IsBackground = true;
  t.Start();
}
void threadProc()
{
  while (true)
  {
    CommMessage msg = comm.getMessage();
    if(msg.type == CommMessage.MsgType.files)
    {
      msg.files.Clear();
      string[] files = Directory.GetFiles("../../Storage");
      foreach(string file in files)
      {
        string filename = Path.GetFileName(file);
        msg.files.Add(filename);
      }
      EndPoint temp = msg.to;
      msg.to = msg.from;
      msg.from = temp;
      comm.postMessage(msg);
    }
   // process other message types here
  }
}
```

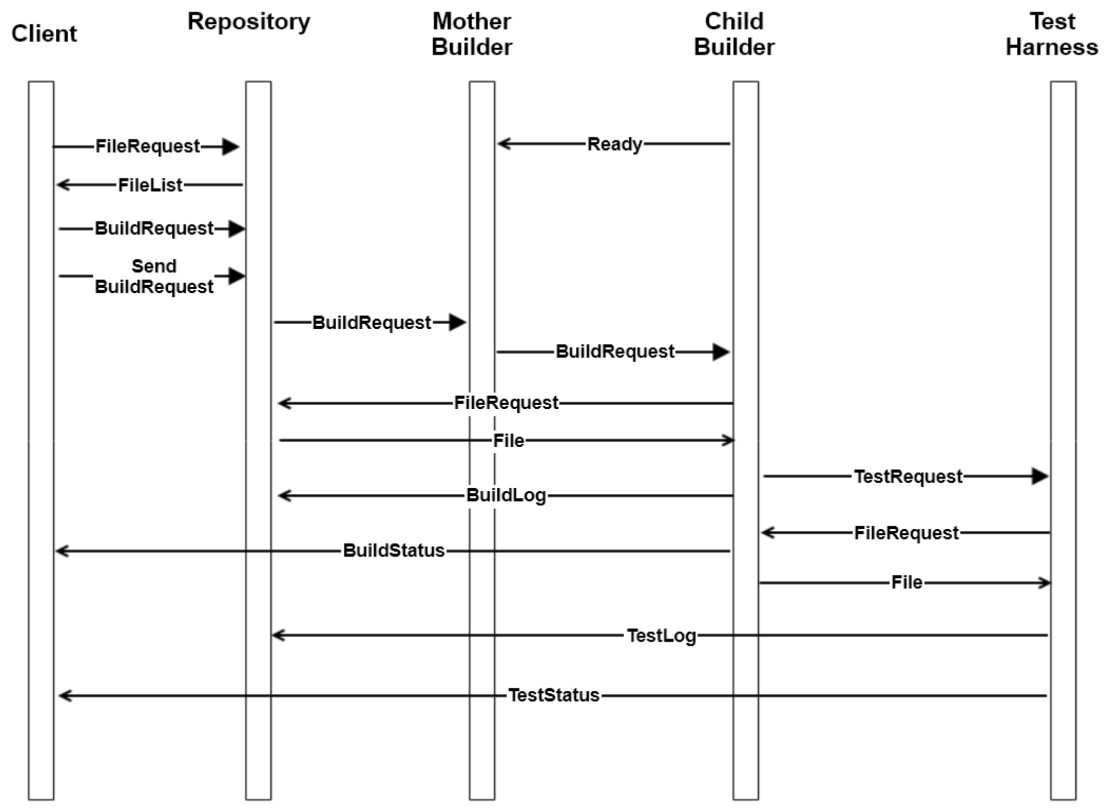
3. Draw a diagram representing message flow in Project #4. Identify each of the messages and the information it carries.

Answer:



BuildRequest – XML string	SendBuildRequest – command
FileRequest – command	FileList – list of strings
BuildLog – text string	File – binary
Ready – command (status)	TestRequest – XML string
TestResult – command (status)	TestLog – text string

Alternate Message-Flow Diagram



4. What is a .Net delegate? Show how a publisher would use a delegate to allow subscribers to start handling events with a method structure:

```
void handler(object sender, string message).
```

Please also show how the subscriber starts to receive event notifications.

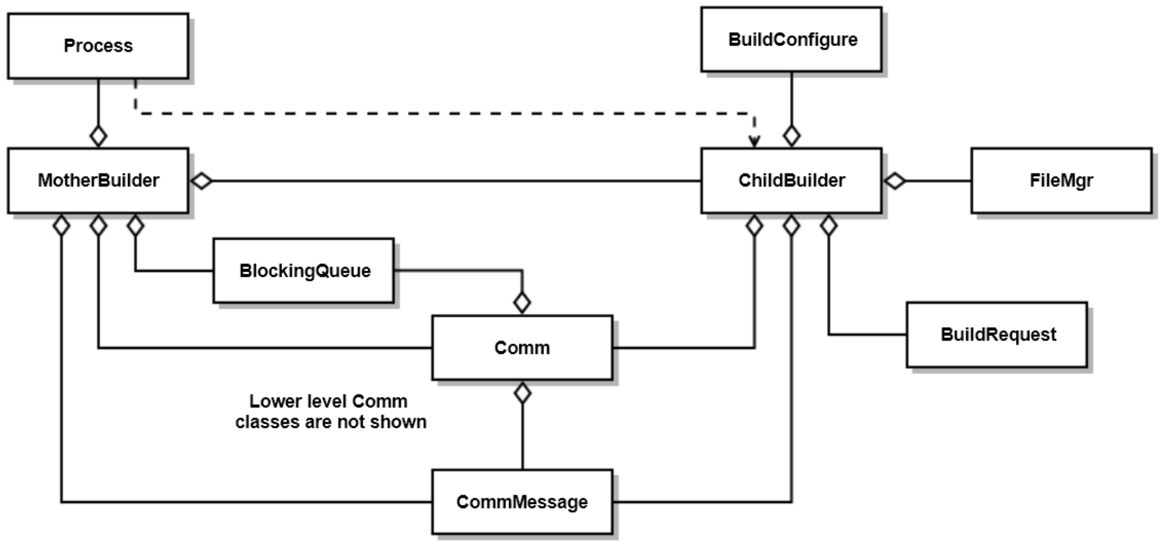
Answer:

A delegate is a .Net reference type that maintains and executes a linked list of event handler references. Delegates are also used to bind to, and transport, lambdas.

```
public class Publisher
{
  public delegate void PubDelType(object sender, string msg);
  public PubDelType pubEvent { get; set; }
  public string name { get; set; } = "JoePub";

  public void doEvents()
  {
    for(int i=0; i<5; ++i)
    {
      if(pubEvent != null)
      {
        pubEvent.Invoke(this, "event msg #" + i.ToString());
      }
    }
  }
}
public class Subscriber
{
  public string name { get; set; } = "JaneSub";
  public void pubEventHandler(object pub, string msg)
  {
    Console.Write(
      "\n  {0} received: \"{1}\" from {2}", name, msg, (pub as Publisher).name
    );
  }
}
class MT3Q4
{
  static void Main(string[] args)
  {
    Console.Write("\n  MT3Q4 - Publish and Subscribe");
    Console.Write("\n ===============================");

    Publisher JoePub = new Publisher();
    JoePub.name = "JoePub";
    Subscriber JaneSub = new Subscriber();

    JoePub.pubEvent += new Publisher.PubDelType(JaneSub.pubEventHandler);
    JoePub.doEvents();
    Console.Write("\n\n");
  }
}
```

5. Draw a class diagram for your design of Project #3's process pool. Please include all classes you would need to meet requirements. Each class should have a single responsibility. For each of your classes, state its responsibility.

Answer:



- MotherBuilder manages ChildBuilders, passing them BuildRequests when ready
- ChildBuilder loads files, matching BuildRequest, from Repository and builds them into libraries and sends them to the TestHarness
- BuildConfigure sets the environment and paths to toolchain to prepare for build
- FileMgr is used to create temp directory, find the libraries for Comm.sendFile, and delete directory.
- System.Diagnostics.Process is used to support starting childBuilders and tracking their exit events.
- Comm and CommMessage support communication between MotherBuilder, ChildBuilders, Repo, and TestHarness
- BuildRequest parses BuildRequest message for source code file names

6. Write all the code for a method that attempts to open a file, and, if open fails, sleeps for a brief time then attempts to open again. That try and wait process repeats for a maximum of Max times. You may assume that the open operation will throw an exception if the attempt fails.

Answer:

```
FileStream openWithRetry(string fileSpec, int MaxTries, int SleepTime)
{
  int count = 0;
  FileStream fs = null;
  while(true)
  {
    try
    {
      count++;
      fs = File.Open(fileSpec, FileMode.Open);
      break;
    }
    catch
    {
      if (count < MaxTries)
      {
        Console.Write("\n  open failed - retrying");
        Thread.Sleep(SleepTime);
      }
      else
      {
        Console.Write("\n  open failed");
        return null;
      }
    }
  }
  return fs;
}
```

7. What is C# reflection? Show how you would determine, using reflection, whether the type of some instance implements an interface, ITest.

Answer:

C# reflection is the process of extracting type information from the metadata of an assembly, defined at the time the assembly is compiled. Here's an example:

```
public interface ITest { bool test(); }
public class Tester : ITest
{
  public bool test() { return false; }
}
class MT3Q7
{
  public static void showImplements<T>(string interfaceName)
  {
    Type t = typeof(T);                            // these two lines are
    Type interf = t.GetInterface(interfaceName);   // a correct answer
    if (interf != null)
    {
      Console.Write("\n  {0} implements {1}", t.Name, interfaceName);
    }
    else
    {
      Console.Write("\n  {0} does not implement {1}", t.Name, interfaceName);
    }
  }

  public static void alternateShowImplements(object obj, string interfaceName)
  {
    Type t = obj.GetType();                        // these two lines are
    Type interf = t.GetInterface(interfaceName);   // a correct answer
    if (interf != null)
    {
      Console.Write("\n  {0} implements {1}", t.Name, interfaceName);
    }
    else
    {
      Console.Write("\n  {0} does not implement {1}", t.Name, interfaceName);
    }
  }
  static void Main(string[] args)
  {
    showImplements<Tester>("ITest");
    showImplements<Tester>("IFoobar");

    alternateShowImplements(new Tester(), "ITest");
    alternateShowImplements(new Tester(), "IFoobar");

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