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. Write all the code to declare a WCF service contract, useful for Project #4:

   Answer:

```
using Command = String;             // Command is key for message dispatching
using EndPoint = String;            // (ip address or machine name):(port number)
using Argument = String;
using ErrorMessage = String;

[ServiceContract(Namespace = "MessagePassingComm")]
public interface IMessagePassingComm
{
  /*----< support for message passing >--------------------------*/

  [OperationContract(IsOneWay = true)]
  void postMessage(CommMessage msg);

  // private to receiver so not an OperationContract
  CommMessage getMessage();

  /*----< support for sending file in blocks >-------------------*/

  [OperationContract]
  bool openFileForWrite(string name);

  [OperationContract]
  bool writeFileBlock(byte[] block);

  [OperationContract(IsOneWay = true)]
  void closeFile();
}
```

2. Define the syntax and semantics for passing a value type by reference to a C# method. Please consider all relevant cases.

Answer:

```
class MT4Q2
{
  public struct X
  {
    public string name;
    public int count;
  }
  public class Y
  {
    public void someFun(ref X x)  // call passing value type X by reference
    {
      x.name = "X";  // caller sees change of state
      x.count = 1;   // caller sees change of state
    }
  }
  static void Main(string[] args)
  {
    Console.Write("\n  MT4Q2 - passing value type by reference");
    Console.Write("\n ========================================");

    X x;
    x.name = "JoeStruct";
    x.count = 0;
    Console.Write(
      "\n  before call: x.name = {0} and x.count = {1}", x.name, x.count
    );
    Y y = new Y();
    y.someFun(ref x);
    // caller sees change of value
    Console.Write(
      "\n   after call: x.name = {0} and x.count = {1}\n\n", x.name, x.count
    );
  }
```

3. Write a well-formatted list of messages you will need for Project #4. Please include name, source and destination, purpose, and contents. Attempt to include every message you will need for that project.

   Answer:

Name	Source	Destination	Purpose	Contents
FileRequest	Client	Repository	Get file list	Command
FileList	Repository	Client	Return file list	File list
BuildRequest	Client	Repository	Send XML string	BuildRequest
SendBuildRequest	Client	Repository	Command	Command
BuildRequest	Repository	BuildServer	Send XML string	BuildRequest
BuildRequest	BuildServer	ChildBuilder	Send XML string	BuildRequest
Ready	ChildBuilder	BuildServer	Notification	Ready status
FileRequest	ChildBuilder	Repository	Get file	File name
File	Repository	ChildBuilder	Send file	File contents
BuildLog	ChildBuilder	Repository	Send build log	Log string
TestRequest	ChildBuilder	TestHarness	Send XML string	TestRequest
FileRequest	TestHarness	ChildBuilder	Get file	File name
File	ChildBuilder	TestHarness	Send file	File contents
TestResult	TestHarness	Client	Notification	Test status
TestLog	TestHarness	Repository	Send test log	Log string
Complete	TestHarness	ChildBuilder	Notification	Complete status

4. Write all the code to define, create, and use a main thread and child thread that share a string[1]. Please write the code to safely have the main thread append "m" to the string and the child append "c" to the string.

   Answer:

```
class Shared
{
  public object synch { get; set; } = new object();
  public StringBuilder test { get; set; } = new StringBuilder();
}
class MT4Q4
{
  void threadProc(object sharedObj)
  {
    for (int i = 0; i < 25; ++i)
    {
      Shared shared = (Shared)sharedObj;
      lock (shared.synch)
      {
        shared.test.Append('c');
      }
      Thread.Sleep(10);
    }
  }
  static void Main(string[] args)
  {
    Console.Write("\n  MT4Q4 - shared string");
    Console.Write("\n ======================");

    MT4Q4 p = new MT4Q4();
    Shared shared = new Shared();
    Thread t = new Thread(p.threadProc);
    t.Start(shared);
    for(int i=0; i<25; ++i)
    {
      lock(shared.synch)
      {
        shared.test.Append('m');
      }
      Thread.Sleep(20);
      Console.Write("\n  {0}", shared.test);
    }
    t.Join();
```

[1] You may wish to consider using a StringBuilder to hold the shared string.

5. Suppose you want to build a FileMgr class that finds all the files and directories on a stated path. How would you avoid putting application detail in the fileMgr so that it is reusable, but allows an application to define what happens when a file or directory is discovered? What would an application need to do to use the FileMgr?

   Answer:

   The FileMgr class declares public delegate types NewFileHandler and NewDirHandler and instances newFileHandler and newDirHandler. Client code defines the onFile and onDir methods for application specific processing and registers with the FileMgr's delegates.

```
public class Navigate
{
  public delegate void NewDirHandler(string dir);
  public event NewDirHandler newDir;
  public delegate void NewFileHandler(string file);
  public event NewFileHandler newFile;

  public Navigate()
  {
  }

  public void go(string path)
  {
    if(newDir != null)
      newDir(path);

    string [] files = Directory.GetFiles(path,"*.*");
    foreach(string file in files)
    {
      if(newFile != null)
        newFile(file);
    }
    string[] dirs = Directory.GetDirectories(path);
    foreach(string dir in dirs)
      go(dir);
  }
}
```

   Client code needs to register its handlers with the Navigator's delegates.

```
void Register(Navigate nav)
{
  nav.newDir += new Navigate.NewDirHandler(OnDir);
  nav.newFile += new Navigate.NewFileHandler(OnFile);
}
```

   Hear, `OnDir` and `OnFile` have been defined by the application.

6. Write all the code for a message dispatcher[2] that, given a CommMessage.command, invokes processing to handle the command. Please include a simple example of message processing for one command, and show how the dispatcher uses that code.

Answer:

```
using Msg = CommMessage;

class CommMessage
{
  public enum MsgType { request, reply };
  public enum MsgCommand {  echo, allother };
  public MsgType type { get; set; } = MsgType.request;
  public MsgCommand command { get; set; } = MsgCommand.allother;
  public string body { get; set; } = "";
}

class MT4Q6
{
  public Dictionary<Msg.MsgCommand, Func<Msg, Msg>> msgDispatcher { get; set; }
    = new Dictionary<Msg.MsgCommand, Func<Msg, Msg>>();

  Func<Msg, Msg> echo = (Msg msg) => {
    Console.Write("\n  {0}", msg.body);
    msg.type = Msg.MsgType.reply;
    return msg;
  };

  MT4Q6()
  {
    msgDispatcher[Msg.MsgCommand.echo] = this.echo;
  }
```

Here's the dispatch:

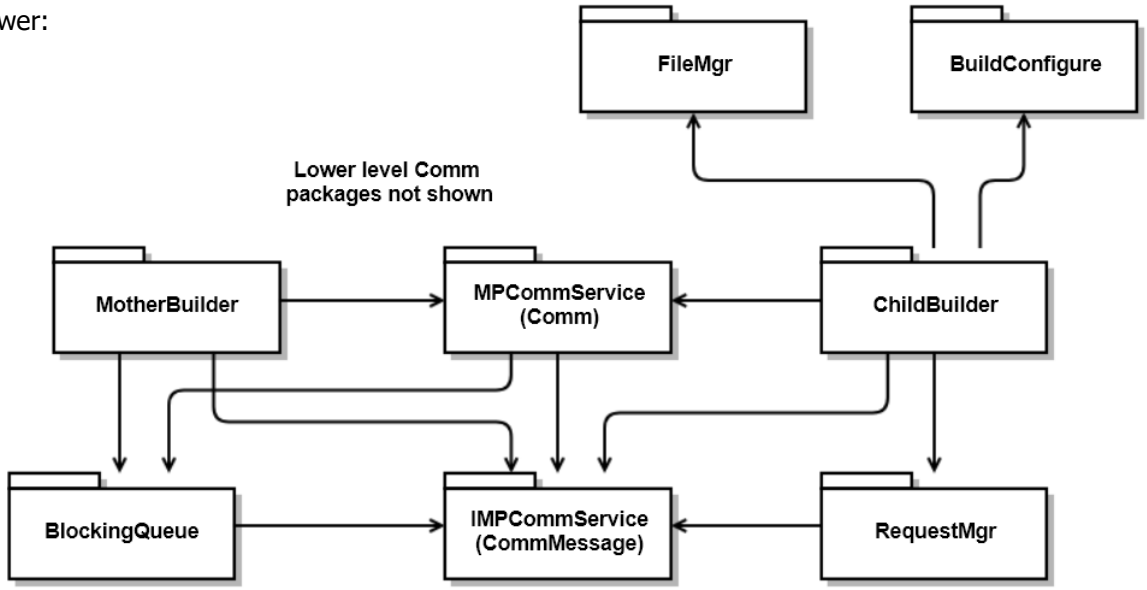
```
Msg reply = p.msgDispatcher[request.command](request);
```

where request and reply are CommMessages.

[2] A message dispatcher invokes all the processing needed to handle each type of message, based on its command property.

7. Draw a package diagram for your design of Project #3's process pool. Please include all packages you would need to meet requirements. Each package should have a single responsibility. For each of your packages, state its responsibility. To save time, you may represent Comm with a single package.

Answer:



MotherBuilder manages ChildBuilder processes

ChildBuilder builds test libraries and sends to TestHarness

BuildConfigure sets the environment and provides access to build tool chain

FileMgr creates and deletes temporary build directories and provides access to files in those directories.

RequestMgr parses BuildRequest messages and creates TestRequest messages

MPCommService provides communication services used by MotherBuilder and ChildBuilders

IMPCommService defines the CommMessage structure used by communicators.