Examination #2

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

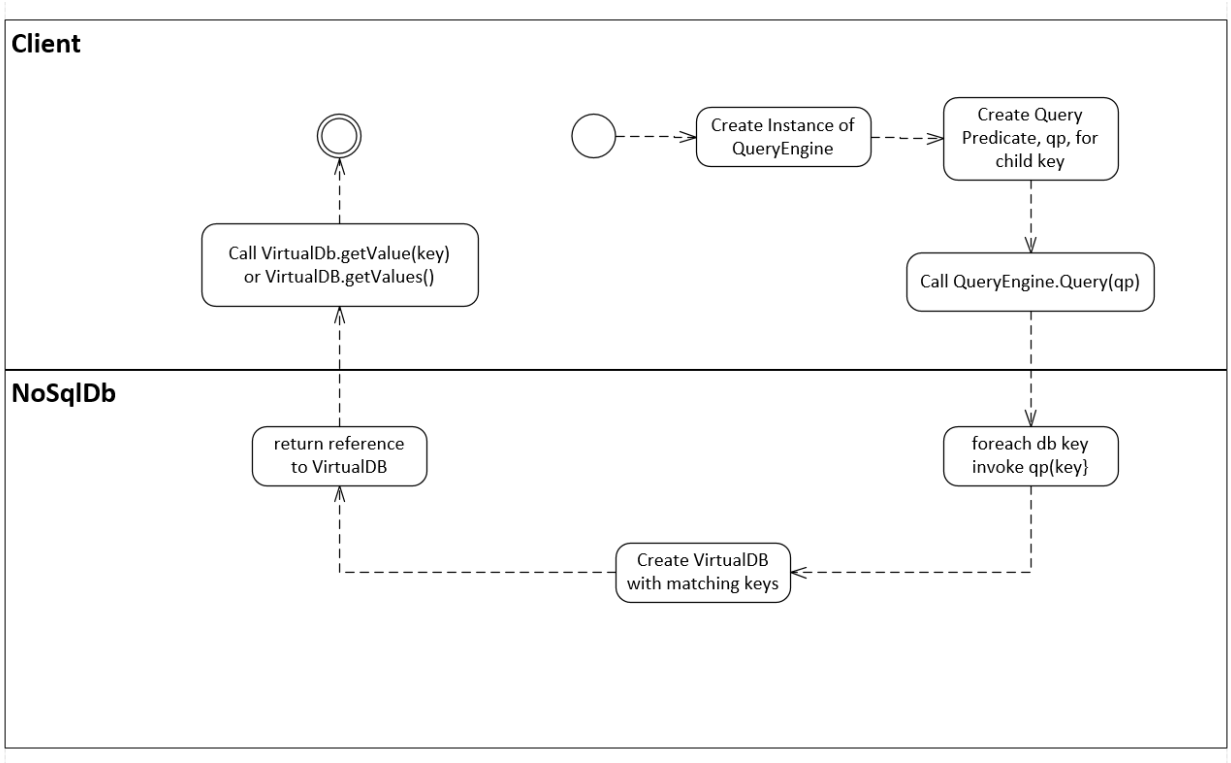
This is a closed book examination. Please place all your books on the floor beside you. You may keep one page of notes on your desktop in addition to this exam package. All examinations will be collected promptly at the end of the class period. Please be prepared to quickly hand in your examination at 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. For Project #2 you where required to implement a NoSqlDb database that holds values defined by instances of a DBElement class. The DBElement class declares metadata that includes a list of child keys and an instance of a payload type. Draw an activity diagram that describes how a client queries the NoSqlDb for values that hold a specified child key and what the NoSqlDb and its associated classes do with that request. You need to be specific about what the code will do.

   Answer:




```
//----< QueryPredicate as answer for MT2Q1, F2015 >-----

Func<string, bool> MakeQueryForSpecifiedChild(string test, DBL pdb)
{
  Func<string, bool> qp = (queryKey) =>
  {
    DBElemL qelem;
    if (!pdb.getValue(queryKey, out qelem))
    {
      return false;
    }
    foreach (string child in qelem.children)
    {
      if (child == test)
        return true;
    }
    return false;
  };
  return qp;
}
```

2. Write all the code to define and use a C# function that takes a single StringBuilder argument and returns void. The function creates a new object of the same type in the caller's scope with the same state as the original. Show how to test for equality of the object before the call with the object after the call. Repeat this demonstration for equality of state.

   Answer:

```
class MT1Q6
{
  static void changeObjectInCallersScope(ref StringBuilder arg)
  {
    string temp = arg.ToString();
    arg = new StringBuilder(temp); // new object with the original object's state
  }
  static void Main(string[] args)
  {
    "MT2Q2 - Function that changes passed object".title('=');

    StringBuilder sb = new StringBuilder("original state");
    StringBuilder orig = sb;

    Console.Write("\n  Original state of StringBuilder is \"{0}\"", sb.ToString());

    MT1Q6.changeObjectInCallersScope(ref sb);
    Console.Write("\n  State of StringBuilder after call is \"{0}\"", sb.ToString());
    Console.WriteLine();

    // show that object has changed

    if (orig == sb)
      Console.Write("\n  object was not changed");
    else
      Console.Write("\n  object was changed");

    if (orig.Equals(sb))
      Console.Write("\n  state of object was not changed");
    else
      Console.Write("\n  state of object was changed");

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

3. Describe any three of the member functions of the System.Object type. Please provide the function signature and describe its operation.

   Answer:

   string System.Object.ToString()
   Provides a string representation of the class that inherits this function. The class may override ToString() to provide something useful, but many of the .Net Framework classes just return the name of the class. Instances of string, StringBuilder, Int32, etc. return more useful information.

   Type System.Object.GetType()
   Returns an instance of the Type class filled with reflection information for the class that inherits this function. When a C# package is compiled into an assembly, the compiler puts all of the type information it developed during compilation into the assembly's metadata, in a tokenized form. The GetType() function parses that information and stores it in an instance of Type which it returns to the caller.

   bool System.Object.Equals(Object)
   The Object class defines Equals(Object) to compare reference handles, e.g., do these two handles refer to the same instance. However, many of the .Net Framework classes, like StringBuilder, overload bool Equals(Object) to provide value comparisons. StringBuilder.Equals(StringBuilder sb) compares the strings that each string builder holds for equality of value.

4. How would you implement a message Receiver for Project #4? You may describe the functioning of the instructor's latest example code or define your own. How do instances of this Receiver support multiple concurrent message senders without needing to provide thread-safety in the NoSqlDb database?

   Answer:

   The Receiver exposes the ICommService to senders, and must aggregate an instance of ServiceHost to do that. ServiceHost is the execution environment for instances of the service, CommService. Instances of CommService accept messages that match the ICommService Data Contract and deposit them into a BlockingQueue shared by all service instances.

   Processes that use an instance of Receiver can access the message queue by instantiating an instance of the service and calling getMessage() on the service instance.

   Alternately the user may create an Action delegate bound to a lambda that does whatever message processing is needed.

   Because concurrent clients place requests into the single thread-safe blocking queue and only one thread dequeues messages, all server activities run on the dequeuing thread and do not have to be locked. That kind of processing is often referred to as a Single Threaded Apartment (STA).

5. Write all the code for a C# class that provides an asynchronous function[1] that will, at the end of its processing, invoke callback processing provided by the client code. Show how the client can provide that callback processing. Please invent simple processing for the asynchronous function and its callback. Writing a Console message will be accepted for that processing.

   Answer:

```
public class AsyncWithCallback
{
  Action doMainTask = null;
  Thread t;

  public AsyncWithCallback()
  {
    doMainTask = () => { Console.Write("\n  I'm busy doing work"); };
  }
  public void asyncRun(Action callback)  // called by client
  {
    ThreadStart ts = () =>
    {
      doMainTask.Invoke();
      callback.Invoke();
    };
    t = new Thread(ts);
    t.Start();
  }
  public void join() { t.Join(); }
}
class MT2Q5
{
  public void myCallbackHandler()
  {
    Console.Write("\n  I'm being called back");
  }
  static void Main(string[] args)
  {
    "MT2Q5 - Asynchronous function that calls back when done".title('=');
    MT2Q5 mt2q5 = new MT2Q5();
    Action callback = () => { mt2q5.myCallbackHandler(); };
    AsyncWithCallback cb = new AsyncWithCallback();
    cb.asyncRun(callback);
    Console.Write("\n  waiting for async to finish");
    cb.join();
    Console.Write("\n\n");
  }
}
```

[1] Asynchronous functions return almost immediately without waiting for the function's processing to complete.

6. What is an extension method? Name two extension methods commonly used with the XDocument class. What do they do?

Answer:

An extension method is a static method of a static class with the signature:

```
public static RT theMethod(this T myType, …)
```

where RT is the return type, perhaps void, and T is the type of the instance to which the method is applied. The invocation has the same syntax as a member of the class, e.g.:

```
T t = new T();
t.theMethod(…);
```

where the ellipsis represents any arguments the method may have, possibly none.

Note that an extension method does not get access to private members of its associated instance. It must use the public interface of the instance to carry out its processing. In the example above, theMethod must use the public interface of t, using the formal parameter, myType.

System.Xml.Linq provides extension methods Elements(…) and Descendents(…) with a couple of overloads that are useful for making Linq queries on XDocument or XElement instances.

XElement.Elements() returns the child elements of the XElement instance to which it is applied. XElement.Descendents() returns a collection of all the descendent elements of the XElement instance to which it is applied.

7. What does the following code do:

```
Action act += () => { Console.Write("\n  Hi there"); };
```

Answer:

It defines a delegate act of the Action type, bound to the lambda processing shown. Note that the lambda is not executed here. It is simply bound to act and will be executed with either of the statements:

```
act() or act.Invoke()
```

by any code that has access to act, perhaps by having it returned in a function call.

We showed in class, using reflection in the LambdaDemos example, that this binding results in an instance of a class that has a method with the code of the lambda and fields for each datum used by the lambda which is embedded in the act Action delegate. In this case there are no data being stored in the internal class.