

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. What is meant by the terms "Publish and Subscribe". How would you implement publish and subscribe in a C# program?

Answer:

Publish and Subscribe is a Design Pattern for event notification. The Publisher, usually part of a library or framework, knows when some event of interest occurs and will notify all Subscribers, usually part of an application's code of the occurrence of the event. The publisher maintains a list of subscriber event handler functions and notifies them of events by invoking their handler functions.

C# provides the delegate syntax to create a MultiCast Delegate type. Once declared, the Delegate type is used to create an instance of the delegate. Usually a Publisher provides one delegate for each different kind of event it uses.

2. You've been asked to describe the software architecture of your Project #4. What will you discuss?

Answer:

You need to discuss all the expected parts of an OCD for Project #4, e.g., uses and user groups, project package structure, responsibilities and interactions of each of the packages, and any critical issues faced by the development team.

Often you will sketch user interfaces, discuss any interesting data structures and the sharing of data across the application, and document any algorithms or unique design ideas that will make the developed code flexible and robust. The WPF support for drag and drop is a good example.

Specific to project #4 you should discuss the latency issues for remote collaborative drawing, locking strategies for UIElements so that multiple users cannot concurrently edit shapes and their positions. Another issue is the difficulty of designing and implementing a graceful and easy to use interface.

3. Describe the syntax and semantics of passing a C# value type by reference.

Answer:

C# value types reside on the stackframe of the function in which they are defined. When passed to a function by reference a reference is created in the functions stackframe referring to the value type in the caller's stackframe. The syntaxes for declaration and invocation are shown below.

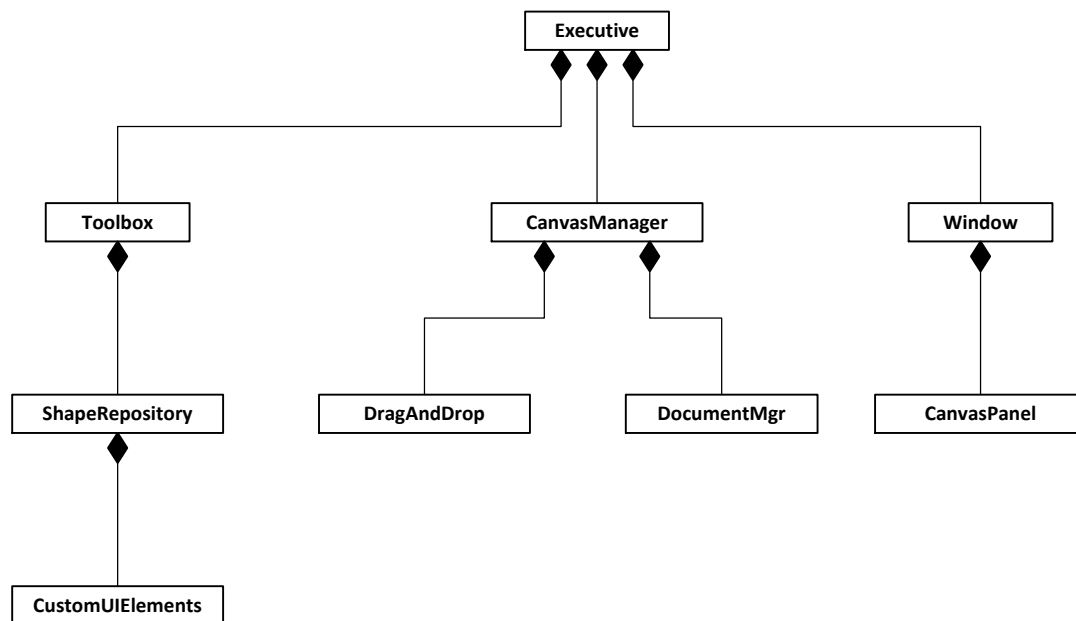
```
double d = 2.5;
void someFun(ref double) { ... }
someFun(ref d);
```

When the function changes the state of the value type it does so through the reference to the caller's value and so the caller also sees the change of state.

4. Draw a class diagram representing a design you may use for your Project #4 implementation.

Answer:

This answer ignores remoting, as that can optionally be deferred to a Project #5 prototype. The main functions that need to be supported are operations on the drawing canvas, maintenance of a toolbox and representations of custom UIElements, supporting drag-and-drop, and managing drawing documents. All of these are represented by classes in the diagram below.



5. A graph is a data structure holding a collection of vertices. Each vertex may have one or more references to other vertices in the graph. We say that the vertex holding the reference is a parent and a referenced vertex is a child of that parent¹. Write all the C# code for a class or classes to hold a Graph data structure.

Answer:

See code MT4Q4 for Main function

```
public class Vertex<T>
{
    T t_;
    List<Vertex<T>> children;

    public Vertex()
    {
        children = new List<Vertex<T>>();
    }
    public Vertex(T t)
    {
        t_ = t;
        children = new List<Vertex<T>>();
    }
    public void addChild(Vertex<T> v)
    {
        children.Add(v);
    }
    public Vertex<T> this[int i]
    {
        get { return children[i]; }
    }
    public T value
    {
        get { return t_; }
        set { t_ = value; }
    }
}
public class Graph<T>
{
    List<Vertex<T>> vertices;

    public Graph()
    {
        vertices = new List<Vertex<T>>();
    }
    public void addVertex(Vertex<T> v)
    {
        vertices.Add(v);
    }
    public Vertex<T> this[int i]
    {
        get { return vertices[i]; }
        set { vertices[i] = value; }
    }
}
```

¹ Graphs may be used to describe dependency relationships among files. Each vertex represents a file and each edge represents a dependency of the parent on that child.

6. Describe the tasks a program must carry out to display a graph in a window provided by Windows Presentation Foundation (WPF). You can assume the graph information is encoded in an XML file. Are there any tasks that seem to present interesting design issues?

Answer:

Assuming that graph means an instance of the graph class of the previous question, here is a sensible task list:

Start the WPF application to create a Window with a Canvas panel.

Load the graph's XML into an XDocument and query for its vertices, including connections and values.

Select a layout strategy. For small graphs you could simply plot each vertex equidistant on a vertical line, then draw arcs to represent vertex connectivity. Even for this simple case developing this task takes some significant thought and trials. If the graph is large this becomes an interesting research problem.

Support drag-and-drop of vertex locations. Any layout strategy is likely to get a fairly good looking diagram for small graphs but even those may need some manual adjustments.

Support labeling vertices and providing other text annotations and titles.

7. Write an XML representation for an instance of your graph class. Write LINQ queries to access that information from an XML file.

Answer:

```
<?xml version="1.0" encoding="utf-8" ?>
<graph>
  <vertex>
    <id>1</id>
    <value>v1</value>
    <child>2</child>
  </vertex>
  <vertex>
    <id>2</id>
    <value>v2</value>
    <child>3</child>
  </vertex>
  <vertex>
    <id>3</id>
    <value>v3</value>
    <child>1</child>
  </vertex>
</graph>
```

```
namespace MT4Q6
```

```
{
  class Program
  {
    static void Main(string[] args)
    {
      XDocument doc = XDocument.Load(@"..\..\Graph.xml");
      Console.Write(doc.ToString());
      Console.WriteLine("\n\n");
      var verts = from e in
                    doc.Elements("graph")
                      .Elements("vertex")
                    select e;

      foreach (var vert in verts)
      {
        var q2 = from e in vert.Elements("id") select e;
        Console.WriteLine("\n vertex id is {0}", q2.ElementAt(0).Value);
        var q3 = from e in vert.Elements("value") select e;
        Console.WriteLine("\n vertex value is {0}", q3.ElementAt(0).Value);
        var q4 = from e in vert.Elements("child") select e;
        int numChildren = q4.Count();
        for (int i = 0; i < numChildren; ++i)
        {
          Console.WriteLine("\n child is {0}", q4.ElementAt(i).Value);
        }
      }
      Console.WriteLine("\n\n");
    }
  }
}
```