**CS345 Test #3 Sample Test**

* 1. The interface IComparable is defined as

interface IComparable<T>

{

int CompareTo(T rhs);

}

int CompareTo(rhs) returns 0 if the this object is equal to the rhs, a negative number if this is less than rhs and a positive number if this is greater than rhs. Write a class called Person that contains an age and a name. Implement IComparable so that it compares Person names.

public class Person : IComparable<Person>

{

public Person(string name, int age)

{

Name = name;

Age = age;

}

public int CompareTo(Person rhs)

{

// return (age – rhs.age); (if I wanted to compare ages)

return String.Compare(Name, rhs.Name);

}

public string Name {get; protected set;}

public int Age {get; protected set;}

}

* 1. Write an example that sorts and array of instances of the Person class.

Person[] people = new Person[] { new Person("Fred", 32), new Person("Wilma", 28), new Person("Barney", 30),

new Person("Betty", 26)};

Array.Sort(people);

1. Why does C# have interfaces? How would you do something similar in C++?
2. What is the purpose of this call?

Application.Run(new Form1());

1. What are some advantages and disadvantages of using Windows Forms vs. WPF?
   1. Advantages
      1. Can separate program development from artistic/design development
      2. External tools can be used to create XAML
      3. Some additional capabilities in WPF
         1. 3d graphics and streaming video
         2. Additional components
      4. Easy to add additional styles/skins
      5. WPF remembers what has been drawn to the screen – the programmer is not responsible for redrawing content.
   2. Disadvantages
      1. Windows forms offer a simpler/lighter weight library
      2. Division of XAML and C# make reading the design a little more difficult
2. Why doesn’t WPF use a Paint handler?
3. Suppose that you wanted to map the entire client window to world coordinates such that the world coordinate system ran from 0 to 100 in the x-axis, 0 to 200 in the y-axis, and was rotated 45 degrees. Write the code to set the world coordinate system for an instance of the graphics class.

graphics.ScaleTransform(ClientRectangle.Width / 100.0f,

ClientRectangle.Height / 200.0f);

graphics.RotateTransform(45.0f);

1. Suppose that you wanted the client window to world coordinates such that the world coordinate system ran from -100 to +100 in the x-axis (left to right) and 100 to -100 (top-to-bottom) in the y-axis. Write the code to set the world coordinate system for an instance of the graphics class.

// The range [-100,100] is 200 wide

const float worldWidth = 200.0f;

// The range [-100,100] is 200 high

const float worldHeight = 200.0f;

// The ScaleTransform tells how many device coordinates there are

// per world coordinate. We want the y-coordinates to increase as we

// travel up the graph, so we multiply the y-coordinate by -1 to

// flip the axis.

graphics.ScaleTransform(

ClientRectangle.Width / worldWidth,

-ClientRectangle.Height / worldHeight);

// We want the left-most points on the graph to have

// x-coordinate -100, rather than 0, so we

// slide the x-coordinate system 100 units to the right.

// We want the top-most point to be 100, so we slide the

// y-coordinate system down 100.

// One way to approach this is to ask:

// 1.) What do I want the top-left coordinate to be?

// In our case it is (-100, 100)

// 2.) What do I have to add to those values to get (0,0)

// In our case it is (100, -100)

graphics.TranslateTransform(100, -100);

1. Given a set of points in device coordinates, how would I translate them to world coordinates?
   1. Create an array of Points
   2. Set a Matrix equal to a copy of the graphics.Transform
   3. Invert the matrix
   4. Use matrix.TransformPoints to perform the conversion.
2. Describe the basic operations necessary to implement a simple animation.
   1. Create an instance of class Timer
   2. Set the timer interval to the interval at which you want to update your animation.
   3. Add an event handler to the timer that will be invoked at the specified interval.
   4. Cause the event handler to update the animation and cause it to be redrawn via an Invalidate().
3. If an XAML description is like a tree, how would you describe the difference between the leaves and branches?
4. What are margins and padding in WPF?
5. How does a StackPanel differ from a Grid? Why would you use one rather than the other?
6. Suppose that I wanted to add a button to my WPF form. Would I use XAML or C#? Explain.
7. What is a delegate? How would we do something like a delegate in C++?
8. Definitions:

Explain and give an example of each of the following terms.

* 1. Double-buffered video display
  2. Resources