Name:

Course: Introduction to .NET
Title: Tutorial 8
Instructor: Bill Buchanan

NAPIER UNIVERSITY
EDINBURGH SCOTLAND

[Encapsulation]

Q8.1 An instrument has a given tag, and reads voltage and power. It is also set to a given range, with a time stamp for the time the data was created, such as:

Instrument Tag	Voltage	Power	Range	Time
AB12345	5.6	1.2	mW	01/07/2004 06:17:55
AB12345	5.3	1.21	mW	01/07/2004 06:17:56
AB12345	5.5	0.99	mW	01/07/2004 06:17:57
AB12345	5.0	0.00095	W	02/07/2004 06:17:58
AB12366	5.1	0.0012	W	02/07/2004 06:17:59
AB12366	4.6	0.0014	W	02/07/2004 06:17:60
AB12345	3.6	0.001	W	02/07/2004 06:17:61

Create a class which will encapsulate these measurements, along with a Display() method to display the actual object. An example of the creation of the objects might be:

```
InstrumentMeasurement Data1 = new InstrumentMeasurement();
Data1.Tag="AB12345"; Data1.Voltage=5.6; Data1.Power=1.2;
Data1.Range="mW"; Data1.Time="01/07/2004 06:17:55";
Data1.Display();

InstrumentMeasurement Data2 = new InstrumentMeasurement();
Data2.Tag="AB12345"; Data2.Voltage=5.3; Data2.Power=1.21;
Data2.Range="mW"; Data2.Time="01/07/2004 06:17:56";
Data2.Display();
```

[Arrays with objects]

Q8.2 Enhance the program in Q8.1 by adding the created objects to an array list. Using this program write a program which will display all the objects in the array list.

[Arrays with objects]

Q8.4 Enhance the program in Q8.2 so that the program determines the following:

- Number of instrument samples with the tag "AB12345".
- Number of voltage readings above 5.05.
- Number of power readings above 1mW.
- Number of readings on the 1st July 2004.

[Ordered list]

Q8.3 Enhance the program in Q8.2 so that the program determines the following:

- Show an ordered list for the tags.
- Shows an ordered list for voltages.
- Shows an ordered list for power ratings.

[Inheritance]

Q8.5 Create a derived class of NewInstrumentMeasure, which is inherited from InstrumentMeasurement, and add the following methods:

- `bool IsTagEqualTo(str);`
- `bool IsVoltageLargerThan(val);`
- `bool IsVoltageSmallerThan(val);`
- `bool IsPowerLargerThan(val);`
- `bool IsPowerSmallerThan(val);`

These should return a true or false, depending on the condition. Now modify the previous program, so that these can be called to determine if the conditions are true, or not.

[Abstract classes]

Q8.6 Modify the example in Q8.5 so that base class contains abstract classes for the conditions, and modify the derived classes so that they properly implement them. What happens if you miss one of them out?

[Sealed classes]

Q8.7 Modify the example in Q8.5 so that base class contains sealed classes for the conditions, What happens to the build process if they are implemented in the derived class?

By deriving from a few system classes, determine a few system classes which are sealed. Why might they be sealed?

[Overriding classes]

Q8.8 Modify the example in Section Q8.5 so that the following methods are implemented in the base class (InstrumentMeasurement):

- `bool IsTagEqualTo(str);`
- `bool IsVoltageLargerThan(val);`
- `bool IsVoltageSmallerThan(val);`
- `bool IsPowerLargerThan(val);`
- `bool IsPowerSmallerThan(val);`

Now create a derived class, which overrides the following method:

- `bool IsVoltageLargerThan(val);`

It should display a message of:

```
‘ Is Larger Than is no longer supported on this system’
```

Possible Solutions

```
using System;

namespace ConsoleApplication6
{
   public class InstrumentMeasurement
   {
      string tag, range, time;
      double voltage, power;

      public string Tag { set { tag=value; } }
      public string Range { set { range=value; } }
      public string Time { set { time=value; } }
      public double Voltage { set { voltage=value; } }
      public double Power { set { power=value; } }
      public void Display()
      {
         Console.WriteLine("Tag: " + tag + " Voltage " + voltage);
      }
   }

   class class1
   {

      static void Main(string[] args)
      {
         InstrumentMeasurement Data1 = new InstrumentMeasurement();
         Data1.Tag="AB12345"; Data1.Voltage=5.6; Data1.Power=1.2;
         Data1.Range="mW"; Data1.Time="01/07/2004 06:17:55";
         Data1.Display();

         InstrumentMeasurement Data2 = new InstrumentMeasurement();
         Data2.Tag="AB12345"; Data2.Voltage=5.3; Data2.Power=1.21;
         Data2.Range="mW"; Data2.Time="01/07/2004 06:17:56";
         Data2.Display();
         Console.ReadLine();
      }
   }
}
```

```
using System;
using System.Collections;

namespace ConsoleApplication6
{
   public class InstrumentMeasurement
   {
      string tag, range, time;
      double voltage, power;

      public string Tag { set { tag=value; } }
      public string Range { set { range=value; } }
      public string Time { set { time=value; } }
```

```
      public double Voltage { set { voltage=value; } }
      public double Power { set { power=value; } }
      public void Display()
      {
         Console.WriteLine("Tag: " + tag + " Voltage " + voltage);
      }
   }

   class class1
   {

      static void Main(string[] args)
      {
         ArrayList a1 = new ArrayList();
         InstrumentMeasurement Data1 = new InstrumentMeasurement();
         Data1.Tag="AB12345"; Data1.Voltage=5.6; Data1.Power=1.2;
         Data1.Range="mW"; Data1.Time="01/07/2004 06:17:55";


         a1.Add(Data1);

         InstrumentMeasurement Data2 = new InstrumentMeasurement();
         Data2.Tag="AB12345"; Data2.Voltage=5.6; Data2.Power=1.2;
         Data2.Range="mW"; Data2.Time="01/07/2004 06:17:55";

         a1.Add(Data2);

         for (int i=0;i<a1.Count;i++)
         {
            InstrumentMeasurement obj = (InstrumentMeasurement) a1[i];
            obj.Display();
         }

         Console.ReadLine();
      }
   }
}



         int counter=0;
         for (int i=0;i<a1.Count;i++)
         {
            InstrumentMeasurement obj = (InstrumentMeasurement) a1[i];
            if (obj.Power>0.001) counter++;
            obj.Display();
         }


//
using System;
using System.Collections;

namespace ConsoleApplication6
{
   public class InstrumentMeasurement
   {
      string tag, range, time;
      double voltage, power;
```

```
      public string Tag { set { tag=value; } }
      public string Range { set { range=value; } }
      public string Time { set { time=value; } }
      public double Voltage { set { voltage=value; }get { return voltage; }  }
      public double Power { set { power=value; } get { return power; }  }
      public void Display()
      {
         Console.WriteLine("Tag: " + tag + " Voltage " + voltage);
      }
   }
   class NewInstrumentMeasurement: InstrumentMeasurement
   {
      public bool IsVoltageLarger(double val)
      {
         if (Voltage>5) return (true);
         else return (false);
      }
   }


   class class1
   {

      static void Main(string[] args)
      {
         ArrayList a1 = new ArrayList();
         NewInstrumentMeasurement Data1 = new NewInstrumentMeasurement();
         Data1.Tag="AB12345"; Data1.Voltage=5.6; Data1.Power=1.2;
         Data1.Range="mW"; Data1.Time="01/07/2004 06:17:55";


         a1.Add(Data1);

         NewInstrumentMeasurement Data2 = new NewInstrumentMeasurement();
         Data2.Tag="AB12345"; Data2.Voltage=4.9; Data2.Power=1.2;
         Data2.Range="mW"; Data2.Time="01/07/2004 06:17:55";

         a1.Add(Data2);
         int counter=0;
         for (int i=0;i<a1.Count;i++)
         {
            NewInstrumentMeasurement obj = (NewInstrumentMeasurement) a1[i];
            if (obj.IsVoltageLarger(5)==true) counter++;
            obj.Display();
         }
         Console.WriteLine("Number greater than 5 is " + counter);

         Console.ReadLine();
      }
   }
}
```