Universidad de la Habana, Ciencia de la Computación

Programación Examen Final Curso 2006-2007

PROBLEMA 1 Filtrando con delegates y serializando

En `Delegate.dll` se encuentra definido el tipo delegate siguiente

```
namespace ExamenProgramacion
{
  public delegate bool MenorIgual(object a, object b);
  //Retorna true si a y b son del mismo tipo y a es menor o igual que b
  //false en caso contrario
}
```

En `Tipos.dll` están definidos los tipos (note que son serializables)

```
nameespace ExamenProgramacion
{
      [Serializable]
      public class Fecha
      {
        public Fecha(int dia, int mes, int año){...}

        //Devuelve true si fecha1 y fecha2 son de tipo Fecha y fecha1 <= fecha2
        //false en caso contrario
        public static bool FechaMI(object fecha1, object fecha2){...}
        public override string ToString(){...}
        ...
      }
      [Serializable]
      public class Circulo
      {
        public Circulo(int x, int y, int radio){...}

        //Devuelve true si circ1 y circ2 son de tipo Circulo y radio de circ1
        //es <= que radio de circ2, false en caso contrario
        public static bool CirculoMI(object circ1, object circ2){...}
        public override string ToString(){...}
        ...
      }
      [Serializable]
      public class Racional
      {
         public Racional(int num, int den){...}

         //Devuelve true si a y b son de tipo Racional y a <= b
         //false en caso contrario
         public static bool RacionalMI(object a, object b){...}
         public override string ToString(){...}
         ...
      }
```

```
}
```

Usted debe implementar una dll `Utils.dll` en la que esté definida la clase

```
namespace ExamenProgramacion
{
  class Util
  {
    public static IEnumerable Intervalo(IEnumerable items,
                                        MenorIgual func,
                                        object inf, object sup)
    { ... }
  }
}
```

El parámetro `items` es un `IEnumerable` que nos da una secuencia de `object`, `func` es un delegate `MenorIgual` para comparar dos objetos y devolver un `bool`. El método `Intervalo` debe devolver un `IEnumerable` con todos aquellos objetos de `items` que según el delegate `func` estén comprendidos entre `inf` y `sup.`

NOTA La implementación correcta de este método `Intervalo` no debe depender en nada de los tipos definidos en `Tipos.dll`

Usted debe implementar también , además de `Utils.dll`, **un proyecto** **`Problema1`** que produzca un ejecutable de Consola **`Problema1.exe`** con una clase

```
namespace ExamenProgramacion
{
  public class Problema1
  {
    ...
    public static void Main(string[] args)
    {
      ...
    }
  }
}
```

NOTA

Póngale explícitamente nombre `Problema1` y la especificación `public` a la clase y al método `Main` (recuerde que cdo crea un proyecto Console Visual Studio no hace esto by default)

Este ejecutable `Problema1` debe usar `Delegate.dll`, `Utils.dll` y `Tipos.dll` y leer de un fichero `Secuencia.datos` (este fichero se le entrega junto con este documento y Ud. lo debe ubicar en la misma carpeta que su ejecutable **`Problema1.exe`**).

El fichero `Secuencia.datos` contiene un objeto de tipo `IEnumerable` que ha sido serializado. A partir de este fichero se deben producir tres ficheros `Fechas.datos`, `Circulos.datos` y `Racionales.datos` (que deben quedar en la misma carpeta de su ejecutable). En cada uno de estos ficheros debe quedar lo siguiente:

`Fechas.datos` debe ser un `IEnumerable` serializado con los objetos de tipo `Fecha` contenidos `Secuencia.datos` que corresponden a fechas de este año.

`Circulos.datos` debe ser un `IEnumerable` serializado con los objetos de tipo `Circulo` contenidos `Secuencia.datos` que estén entre los radios `30` y `60`.

`Racionales.datos` debe ser un `IEnumerable` serializado con los objetos de tipo `Racional` contenidos `Secuencia.datos` que estén en `Secuencia.datos` y que estén entre los racionales `1/2` y `1.`

NOTA

Recuerde que la comparación por menor igual de dos objetos de tipo diferente es `false`

Recuerde que los tipos y métodos necesarios para hacer la entrada y salida y la serialización están en los namespaces `System.IO` y `System.Runtime.Serialization.Formatters.Binary`.

Recuerde que los arrays y el tipo `ArrayList` son serializables.

Si lo desea para hacer sus pruebas, su programa **`Problema1`** podrá hacer salida por consola (**PERO CUIDE DE QUE EL EJECUTABLE QUE ENTREGUE NO REQUIERA DE NINGUNA ENTRADA POR CONSOLA**).

Lo único que se evaluará es si su aplicación crea bien los ficheros `Fechas.datos`, `Circulos.datos` y `Racionales.datos` con los datos correspondientes y usando `Utils.dll`.

Para ayudarlo a probar se le suministra un fichero `Secuencia.datos` que contiene fechas, círculos y racionales, pero a la hora de evaluarle su aplicación el fichero `Secuencia.datos` que se usará no tiene por qué ser el mismo que se le suministra aquí.