```
//=====================================================================
// Developed By:     Natalia M. Requejo
// Date:             10/10/2011
//=====================================================================

//=====================================================================
//Flyweight: elimina o reduce la redundancia cuando tenemos gran
//cantidad de objetos que contienen información idéntica.
//=====================================================================


/// <summary>
/// Structural Flyweight Design Pattern.
/// </summary>
/// mantiene una referencia a objetos flyweights
/// </summary>
       string m_Text;

// Arbitrary extrinsic state
int m_Extrinsicstate = 22;
FlyweightFactory m_FlyweightFactory = new FlyweightFactory();
Flyweight m_Flyweight_x = null;
Flyweight m_Flyweight_y = null;
Flyweight m_Flyweight_z = null;
UnsharedConcreteFlyweight m_Flyweight_u = new UnsharedConcreteFlyweight();

// Work with different flyweight instances
m_Flyweight_x = m_FlyweightFactory.GetFlyweight("X");
m_Text += m_Flyweight_x.Operation(System.Threading.Interlocked.Decrement(ref
m_Extrinsicstate));

m_Flyweight_y = m_FlyweightFactory.GetFlyweight("Y");
m_Text += m_Flyweight_y.Operation(System.Threading.Interlocked.Decrement(ref
m_Extrinsicstate));

m_Flyweight_z = m_FlyweightFactory.GetFlyweight("Z");
m_Text += m_Flyweight_z.Operation(System.Threading.Interlocked.Decrement(ref
m_Extrinsicstate));

m_Text += m_Flyweight_u.Operation(System.Threading.Interlocked.Decrement(ref
m_Extrinsicstate));


/// <summary>
/// The FlyweightFactory class
/// crea y gestiona los objetos Flyweight
/// </summary>
class FlyweightFactory
{
    private Hashtable m_Flyweights = new Hashtable();
```

```
    // Constructor
    public FlyweightFactory()
    {
        m_Flyweights.Add("X", new ConcreteFlyweight());
        m_Flyweights.Add("Y", new ConcreteFlyweight());
        m_Flyweights.Add("Z", new ConcreteFlyweight());
    }

    public Flyweight GetFlyweight(string pKey)
    {
        return ((Flyweight)m_Flyweights[pKey]);
    }
}

/// <summary>
/// The Flyweight abstract class
/// declaran una interface a través de la que flyweights pueden recibir y actuar
sobre estados externos
/// </summary>
abstract class Flyweight
{
    public abstract string Operation(int pExtrinsicstate);
}

/// <summary>
/// The ConcreteFlyweight class
/// Implementa la interfaz del flyweight y añade almacenamiento para el estado
interno
/// </summary>
class ConcreteFlyweight : Flyweight
{

    public override string Operation(int pExtrinsicstate)
    {
        return Constants.vbNewLine + "ConcreteFlyweight: " +
pExtrinsicstate.ToString();
    }
}

/// <summary>
/// The UnsharedConcreteFlyweight class
/// no todas las subclases de Flyweight necesitan ser compartidas
/// </summary>
class UnsharedConcreteFlyweight : Flyweight
{

    public override string Operation(int pExtrinsicstate)
    {
        return Constants.vbNewLine + "UnsharedConcreteFlyweight: " +
pExtrinsicstate.ToString();
```

```
    }
}
```