Decisiones Importantes de Diseño:

**Clase Analyisis**

**1.PhraseType enum**El atributo PhraseType es de tipo Type, un enum que toma los valores {positive,negative,neutral}

**2.Constructor con parámetro Phrase.**Carga en null el atributo Entity, carga *neutral*el Type, y clona la phrase en el atributo Phrase.

**Clase AnalyisisLogic**

**1.SearchEntity y SearchFeeling**  
Reciben un array de Entity y de Feeling (para evitar dar acceso real a las listas del sistema) y los recorren.

Nota: Resolver problema de acceso a Entity y Feeling. Quizás hay que quitar las properties de acceso

**2. SearchEntity return null**  
Si no encuentra un Feeling en Phrase, retorna null.

**3. SearchEntity return Entity**Si encuentra una Entity la clona al atributo Entity de Analysis.

**4. Alias Type**  
Se utilizó un alias Type = BusinessLogic.Analysis.Type

**Clase Feeling**

**1.Feeling Type**Se utiliza un atributo booleano para definir el Type positivo(true)/negativo(false)

**Clase Alarma**

**1.TimeBack**Se toma el tiempo de evaluación en unidad horas, con un atributo TimeBack

**2.Type y State**  
Se utiliza un atributo booleano para definir el Type positivo(true)/negativo(false) de la alarma  
Se utiliza un atributo booleano para definir el State positivo(true)/negativo(false) actual de la alarma

**3.Servicios**Se permite resetear el Counter al Sistema con ResetCounter(), y aumentar el Counter de a uno con IncreaseCounter(). IncreaseCounter chequea el estado de la alarma cada vez que se ejecuta.

**Clase FeelingAnalyzer**

**1.Metodos de verificación add-delete de listas**Se crean diferentes métodos de verificación para cada lista, aunque tengan código en común. Esto se hace debido a que en un futuro la lógica del programa puede cambiar.  
Ej: La validación de un sentimiento no es la misma que la de una frase.