# EXAMEN FINAL DE PROGRAMACIÓN

# CURSO 2011-2012

# BUZÓN DE CORREOS

El problema consiste en implementar una clase `Buzon`, que contiene la "funcionalidad" de un buzón de correo electrónico. Los mensajes que maneja este buzón son representados por instancias de la siguiente clase `Mensaje`:

```
public class Mensaje

{

 public Mensaje(string remitente, DateTime fecha, string asunto, string cuerpo){…}

 public DateTime Fecha { get; set; }

 public string Remitente { get; set; }

 public string Asunto { get; set; }

 public string Cuerpo { get; set; }

}
```

El `Buzon`, recibe mensajes de correo a través de la ejecución del siguiente método:

```
public bool Recibe(Mensaje mensaje);
```

El cual retorna valor `true` si el mensaje es aceptado y `false` si es rechazado. Un mensaje es rechazado si su tamaño excede un límite. Consideraremos como tamaño de un mensaje a la suma del total de palabras de su asunto y cuerpo. El tamaño máximo de un mensaje en cada buzón está dado por la propiedad `MaximoDePalabrasPorMensaje`:

```
public int MaximoDePalabrasPorMensaje { get; set; }
```

Por una palabra se entiende una secuencia continua de caracteres que no contenga espacios en blanco, símbolos de puntuación (`','`, `'.'`, `';'`, `':'`), tabulaciones (`'\t'`) o cambios de línea (`'\n'`). Por ejemplo, el siguiente texto contiene 11 palabras (las hemos subrayado para mayor claridad):

```
"Hola, Juan:\n\tTe  escribo con gusto estas 11  palabras.
\nSaludos,\n\t\tPedro"
```

Note que entre las palabras puede haber cualquier cantidad de espacios en blanco, tabulaciones, cambios de línea o símbolos.

Los mensajes se reciben en un buzón sin un orden específico.

Los mensajes aceptados por el buzón pueden ser catalogados según las categorías que contenga la lista que retorne la propiedad `Categorias`:

```
public List<ICategoria> Categorias {get;}
```

Garantice que al crearse una instancia de la clase `Buzon`, la lista que retorne esta propiedad sea vacía pero no `null`. Las categorías se agregan al buzón, adicionando elementos a esta lista de categorías.

Una categoría está representada por una instancia de una clase que implemente la interfaz `ICategoria`:

```
public interface ICategoria

{

  string Nombre { get; }

  bool Incluye(Mensaje mensaje);

}
```

Un mensaje estará incluido en una categoría si el método `Incluye` aplicado al mensaje evalúa `true`. Note que un mensaje aceptado puede estar incluido en varias categorías a la vez y también pudiera no estar incluido en ninguna. El nombre de una categoría NO puede ser la cadena vacía.

La clase buzón debe tener un método:

```
public IEnumerable<Mensaje> MensajesEnBuzon();
```

que devuelve todos los mensajes ordenados en orden creciente según su fecha.

El método:

```
public IEnumerable<Mensaje> MensajesEnCategoria(string categoria);
```

Devuelve todos los mensajes que han sido incluidos en la categoría cuyo nombre se pasa como parámetro. Si no hay mensajes en un buzón pertenecientes a la categoría o si el buzón no contempla la categoría en cuestión, el resultado no debe contener ningún mensaje.

Los métodos anteriores tienen las sobrecargas:

```
public IEnumerable<Mensaje> MensajesEnBuzon(IComparer<Mensaje> comparador);

public IEnumerable<Mensaje> MensajesEnCategoria(string categoria,
IComparer<Mensaje> comparador);
```

Que tienen un comportamiento similar a los dos anteriores, el primero retorna todos los mensajes del buzón pero ordenados de menor a mayor según `comparador`, mientras que el segundo devuelve los mensajes pertenecientes a la categoría cuyo nombre se pasa como parámetro, ordenados según `comparador`.

La clase `Buzon` debe tener un constructor sin parámetros. Usted debe implementar los miembros públicos que se han indicado aquí, para completar el ejercicio (empleando para ello la plantilla que se le ha proporcionado junto con esta orientación).

Por ejemplo, para crear un buzón y ponerle un límite de palabras a cada correo se debe escribir:

```
Buzon buzon = new Buzon();
buzon.MaximoDePalabrasPorMensaje = 10;
```

Luego, la siguiente expresión evalúa a `false`, pues el correo tiene más de 10 palabras entre asunto y cuerpo:

```
buzon.Recibe(new Mensaje("pedro@yahoo.com",
                         new DateTime(2012, 5, 12),
                         "Saludos",
                         "Hola,  Juan:\n\tTe    escribo  con  gusto  estas  11
palabras. \nSaludos,\n\t\tPedro"))
```

Sin embargo, las siguientes evalúan `true`:

```
buzon.Recibe(new Mensaje("maria@yahoo.com",
                          new DateTime(2011, 3, 3),
                          "Recordatorio",
                          "Recuerda pasar      por la casa."))

buzon.Recibe(new Mensaje("jose@matcom.uh.cu",
                         new DateTime(2012, 6, 1),
                         "Estamos a la espera",
                         "Trae  los documentos \n\tJose.."))

buzon.Recibe(new Mensaje("jose@yahoo.com",
                         new DateTime(2010, 2, 10),
                         "Problema",
                         "...No puedo ir..."))

buzon.Recibe(new Mensaje("jose@matcom.uh.cu",
                         new DateTime(2012, 6, 2),
                         "Ausencia",
                         "No viniste\n\nJose"))

buzon.Recibe(new Mensaje("maria@matcom.uh.cu",
                         new DateTime(2012, 6, 1),
                         "Informe",
                         "Puse -el-informe- en el escritorio."))
```

Si después de haber agregado los correos anteriores se ejecuta el código:

```
foreach (Mensaje m in buzon.MensajesEnBuzon())
  Console.WriteLine("{0} - {1}",m.Remitente, m.Asunto);
```

Entonces se imprime en consola:

```
jose@yahoo.com - Problema
maria@yahoo.com - Recordatorio
maria@matcom.uh.cu - Informe
jose@matcom.uh.cu - Estamos a la espera
jose@matcom.uh.cu – Ausencia
```

Si se tiene una clase `CorreosMatcom` que implementa `ICategoria`, que incluye aquellos correos en los que el remitente contenga la subcadena `"matcom.uh"` y cuyo nombre es `"Matcom"`, al ejecutar el siguiente código:

```
ICategoria c = new CorreosMatcom(); // c.Nombre devuelve "Matcom"
buzon.Categorias.Add(c);
foreach (Mensaje m in buzon.MensajesEnCategoria("Matcom"))
  Console.WriteLine(m.Remitente);
```

Se imprime:

```
maria@matcom.uh.cu
jose@matcom.uh.cu
jose@matcom.uh.cu
```

Si se tiene una clase `ComparadorPorAsunto` que implementa `IComparer<Mensaje>` y que compara dos mensajes según el orden alfabético del asunto, al ejecutar el código siguiente:

```
foreach (Mensaje m in buzon.MensajesEnBuzon(new ComparadorPorAsunto()))
  Console.WriteLine("{0} - {1}",m.Remitente, m.Asunto);
```

Se imprime:

```
jose@matcom.uh.cu - Ausencia
jose@matcom.uh.cu - Estamos a la espera
maria@matcom.uh.cu - Informe
jose@yahoo.com - Problema
maria@yahoo.com - Recordatorio
```

Si finalmente, se ejecuta el siguiente código:

```
foreach (Mensaje m in buzon.MensajesEnCategoria("Matcom",
                                                new ComparadorPorAsunto()))

  Console.WriteLine("{0} - {1}",m.Remitente, m.Asunto);
```

Se imprime:

```
jose@matcom.uh.cu - Ausencia
jose@matcom.uh.cu - Estamos a la espera
maria@matcom.uh.cu - Informe
```