Deusto
Facultad de Ingeniería
Faculty of Engineering

G-SMS Caso de Estudio
Gestión de Mensajes SMS Message Management para Programas TV
Elaboración del Diagrama de Clases de Diseño de Alto Nivel

Paso 1 – Resultados del Taller de Requisitos

Se desea desarrollar una solución software para la *Gestión Centralizada de Mensajes SMS para Programas Televisivos (G-SMS)*. Esta aplicación tendrá dos áreas bien diferenciadas: una relativa a las labores de administración (creación de concursos, adición de restricciones, cierre de concursos y generación de informes), y otra la relativa al envío de mensajes SMS.

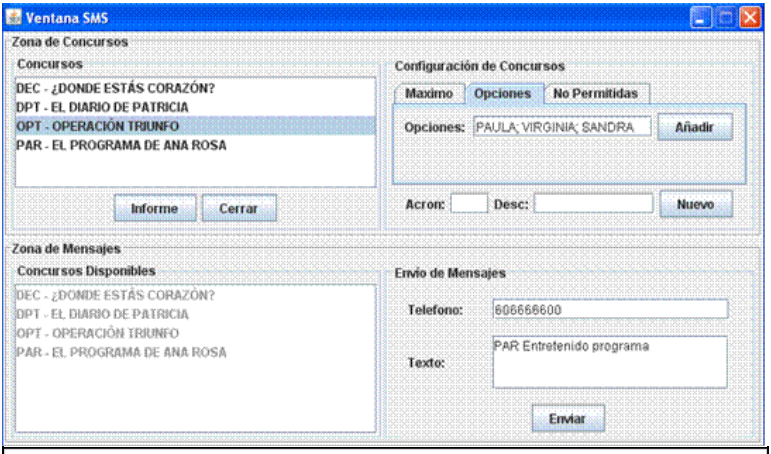
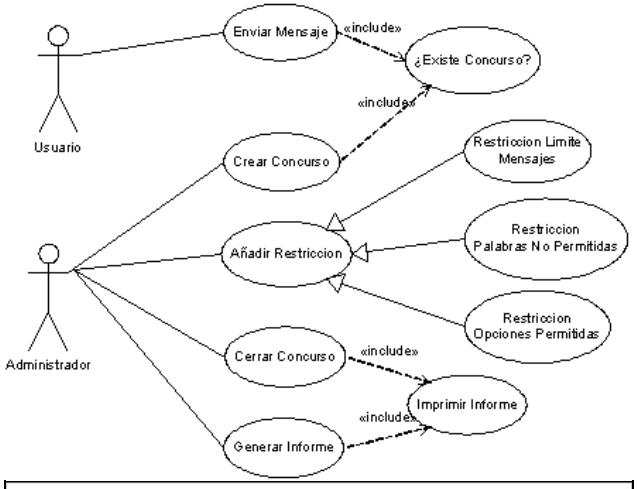


Figura 1. Prototipo de Interfaz: área de Administración y Envío de Mensajes.

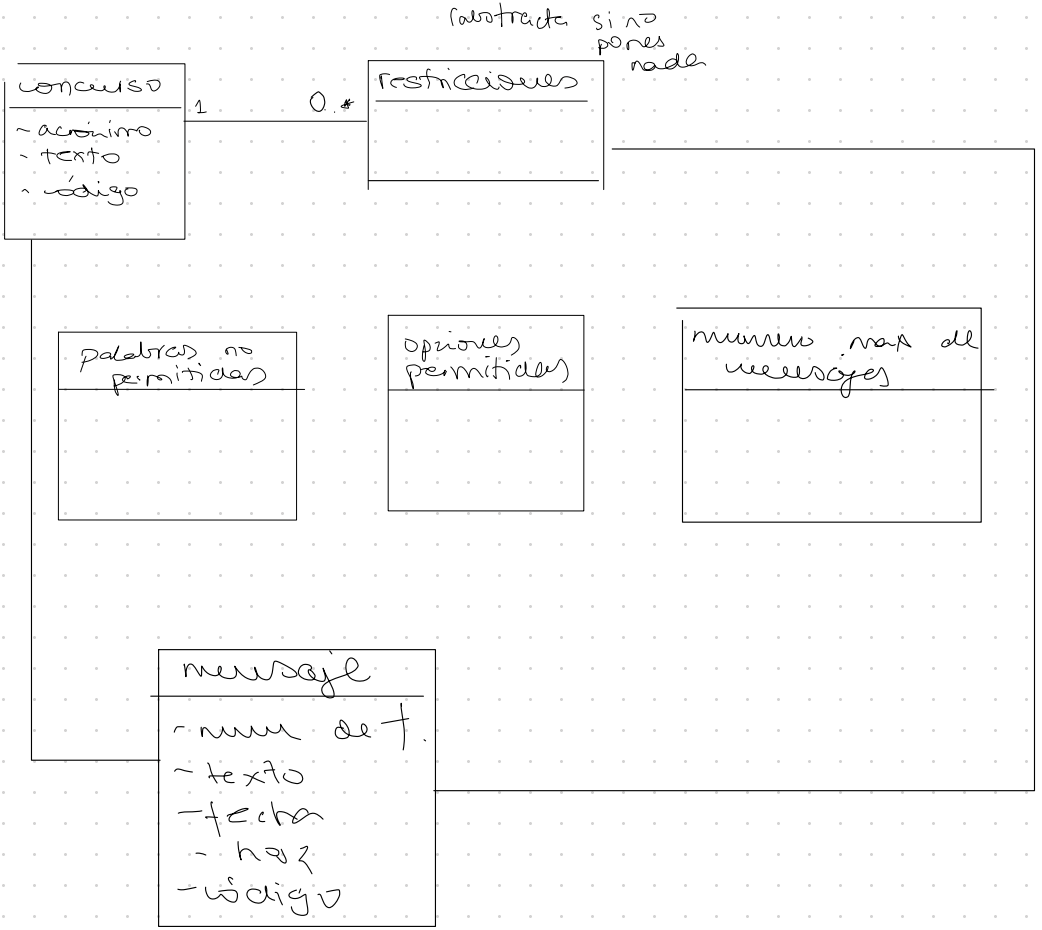
DIAGRAMA DE CASOS DE USO

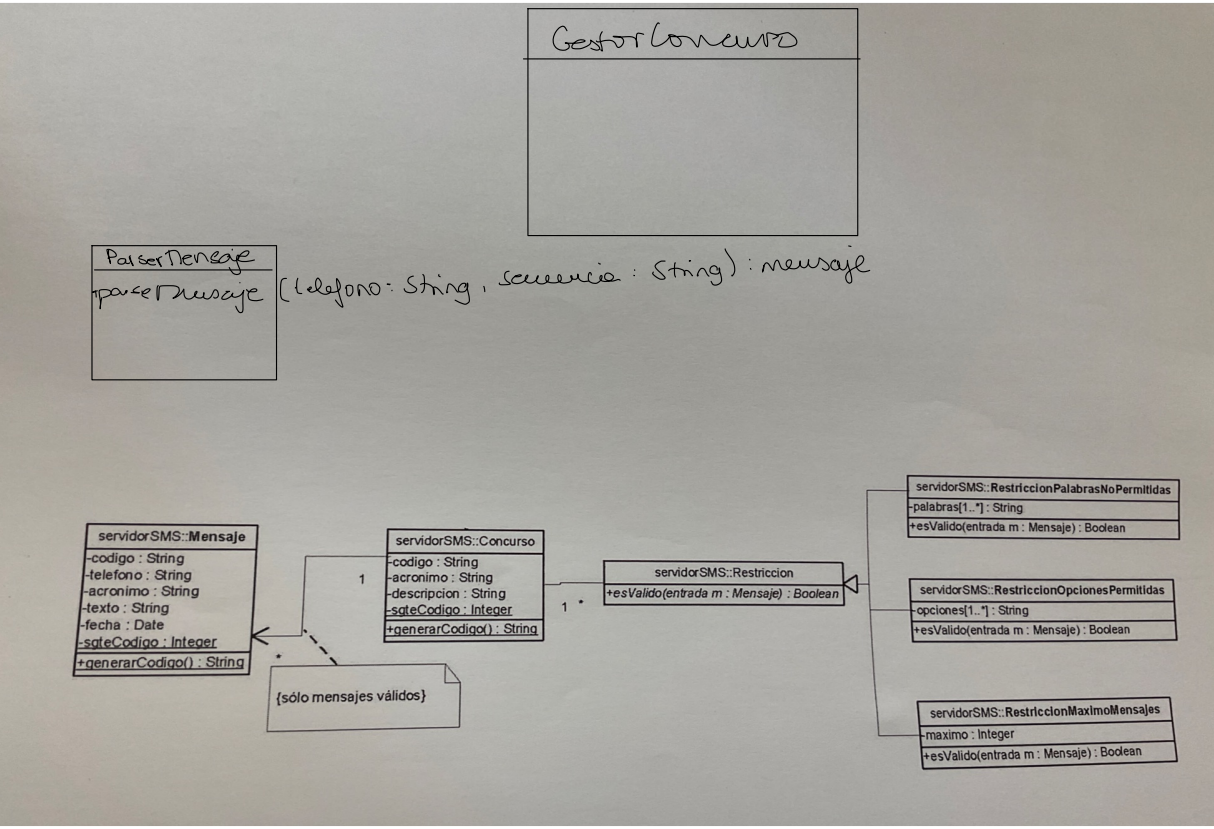


A modo de documentación, en la Figura 1 se ilustra un prototipo de la interfaz. Nótese que para simplificar, en una única pantalla se recogen ambas áreas previamente mencionadas. La parte superior de la pantalla corresponde a Administración y la inferior al Envío de Mensajes. Además se incluye el Diagrama de Casos de Uso.

Descripción Inicial del Problema – DETALLES DEL DOMINIO

Se dispone de la siguiente información para elaborar el modelo de dominio inicial del problema: "*Concurso* es el nombre genérico para cualquier programa de televisión que admita la recepción textos por medio de mensajes SMS. Todos los concursos tienen un acrónimo y un texto descriptivo (véase la Figura 1), y también dispone de un código único. Un concurso puede tener varias *restricciones*, es decir, limitaciones con respecto a la validez de los mensajes. En la actualidad, se consideran tres categorías de restricciones (pero este número podría ampliarse en el futuro). La restricción de 'palabras no permitidas' controla que en el texto de los mensajes no haya ninguna palabra de una lista de palabras no autorizadas. La restricción de 'opciones permitidas' controla que en el texto de los mensajes sólo se encuentre una opción de las presentes en una lista de opciones válidas. La restricción de 'número máximo de mensajes' limita el número máximo de mensajes que se pueden enviar desde el mismo número de teléfono. Independientemente de su tipo, toda restricción está vinculada a un concurso (aunque puede haber concursos que no tengan restricciones) y tiene una responsabilidad: determinar si un mensaje es válido o no,


rabotracta si no pones nada
concurso
- acrónimo
- texto
- código
1
0..*
restricciones
palabras no permitidas
opciones permitidas
numero max de mensajes
mensaje
- num de t.
- texto
- fecha
- hora
- código


GestorConcurso
ParserMensaje
+parseMensaje (telefono: String, secuencia: String): mensaje
servidorSMS::Mensaje
-codigo : String
-telefono : String
-acronimo : String
-texto : String
-fecha : Date
-sgteCodigo : Integer
+generarCodigo() : String
servidorSMS::Concurso
-codigo : String
-acronimo : String
-descripcion : String
-sgteCodigo : Integer
+generarCodigo() : String
1
*
1
*
servidorSMS::Restriccion
+esValido(entrada m : Mensaje) : Boolean
{sólo mensajes válidos}
servidorSMS::RestriccionPalabrasNoPermitidas
-palabras[1..*] : String
+esValido(entrada m : Mensaje) : Boolean
servidorSMS::RestriccionOpcionesPermitidas
-opciones[1..*] : String
+esValido(entrada m : Mensaje) : Boolean
servidorSMS::RestriccionMaximoMensajes
-maximo : Integer
+esValido(entrada m : Mensaje) : Boolean

Deusto
Facultad de Ingeniería
Faculty of Engineering

Caso de Estudio G-SMS
Refinamiento del Diagrama de Clases de Diseño de Alto Nivel

En las siguientes páginas, se adjunta un extracto del código en Java que implementa la simulación del dominio G SMS.

Con la información de las clases, sus características, atributos y operaciones, propiedades de los mismos, relaciones, etc, que aparece en el código:

1. Realiza un Diagrama de Clases de Diseño que puedas comparar con el obtenido previamente de los requisitos.
2. Revisa las asociaciones. ¿Cuáles son bidireccionales y cuáles unidireccionales?
3. Revisa la sección de Cuestiones Interesantes a Tener en Cuenta del documento anterior y reflexiona sobre cómo cada una de ellas afecta al código

Tiempo estimado: 30 minutos

Esquema de Código - Leyenda:

	Atributos modelables de una clase y que por tanto deberían figurar en el diagrama. Generalmente se trata de los atributos 'esenciales' o característicos de la entidad que la clase está modelando.
	Atributos no modelables de la clase. Son atributos que por diversos motivos están presentes en el código, pero que no deben aparecer en el modelo por responder a características de implementación. Mayoritariamente son atributos que implementan relaciones.
	Métodos relevantes de la clase. Se han incluido en este grupo todos los métodos de importancia de la clase, que deberían aparecer en el diagrama de clases y en los de secuencia. Sin embargo algunos no se deberían modelar por ser triviales, tales como constructores, get/set,...

CLASE VENTANASMS

```
package interfaz;

import servidorSMS.*;
import informes.*;

import java.util.List;
import java.util.ArrayList;
import java.util.StringTokenizer;
import javax.swing.*;

public class VentanaSMS extends JFrame {

   // ***SECCIÓN DE ATRIBUTOS***

   [...]

   //¿CÓMO SE MODELA ESTA RELACIÓN?
   private IGestorConcursos gc;

   // ***SECCIÓN DE MÉTODOS***
      [...]
```

INTERFAZ IGESTORCONCURSOS

```
package servidorSMS;
import java.util.List;

   // DECLARACIÓN DE UN CONTRATO DE SERVICIOS

public interface IGestorConcursos {
  public List obtenerConcursos();
  public void nuevoConcurso(String acronimo, String descripcion);
  public void añadirRestriccionMaximoMensajes(int maximo, Concurso concurso);
  public void añadirRestriccionOpcionesPermitidas(List opciones, Concurso concurso);
  public void añadirRestriccionPalabrasNoPermitidas(List palabras, Concurso concurso);
  public void cerrarConcurso(Concurso concurso);
  public void recibirMensaje(String telefono, String texto);

 }
```

CLASE GESTORCONCURSOS

```
package servidorSMS;

import java.util.Collection;
import java.util.Map;
import java.util.List;

public class GestorConcursos implements IGestorConcursos {

   // ***SECCIÓN DE ATRIBUTOS***


   // ATRIBUTOS QUE IMPLEMENTAN ASOCIACIONES.
```


Deusto
Facultad de Ingeniería
Faculty of Engineering
Caso de Estudio G-SMS
Refinamiento del Diagrama de Clases de Diseño de Alto Nivel

```
   // DEFINICIÓN DE ESTRUCTURA QUE SOPORTA BÚSQUEDAS DIRECTAS
   private Map<String,Concurso> concursos;

   // ***SECCIÓN DE MÉTODOS***

      [...]
   }
   // MÉTODO DE BÚSQUEDA DE OBJETOS
   private Concurso buscarConcurso(String acronimo){
      [...]
      }
```

CLASE CONCURSO

```
package servidorSMS;

import java.util.List;
import java.util.ArrayList;

public class Concurso {

   // ***SECCIÓN DE ATRIBUTOS***

   // ATRIBUTOS DE BASE
   private String codigo;
   private String acronimo;
   private String descripcion;

   // ATRIBUTO CON ÁMBITO DE CLASE
   private static int sgteCodigo = 0;

   // ATRIBUTOS QUE IMPLEMENTAN ASOCIACIONES.
   private List<Restriccion> restricciones;
   private List<Mensaje> mensajes;

   // ***SECCIÓN DE MÉTODOS***

   // MÉTODO CON ÁMBITO DE CLASE
   public static String generarCodigo(){
      [...]
   }
```

CLASE RESTRICCION

```
package servidorSMS;

public abstract class Restriccion {

   // ***SECCIÓN DE ATRIBUTOS***

   // ATRIBUTO QUE IMPLEMENTA UNA ASOCIACIÓN
   private Concurso concurso;

   // ***SECCIÓN DE MÉTODOS***
```

```
    // OPERACIÓN ABSTRACTA
    public abstract boolean esValido(Mensaje m);
}
```

CLASE RESTRICCIONMAXIMOMENSAJES

```
package servidorSMS;

public class RestriccionMaximoMensajes extends Restriccion{

    // ***SECCIÓN DE ATRIBUTOS***

    private int maximo;

    // ***SECCIÓN DE MÉTODOS***

    // MÉTODOS DE NEGOCIO
    public boolean esValido(Mensaje m) {
        [...]
    }
}
```

CLASE RESTRICCIONOPCIONESPERMITIDAS

```
package servidorSMS;

import java.util.List;

public class RestriccionOpcionesPermitidas extends Restriccion{

    // ***SECCIÓN DE ATRIBUTOS***

    private List opciones;

    // ***SECCIÓN DE MÉTODOS***

    // MÉTODOS DE NEGOCIO
    public boolean esValido(Mensaje m) {
        [...]
    }
}
```

CLASE RESTRICCIONPALABRASNOPERMITIDAS

```
package servidorSMS;

import java.util.List;

public class RestriccionPalabrasNoPermitidas extends Restriccion{

    // ***SECCIÓN DE ATRIBUTOS***

    private List palabras;
```



```
    // ***SECCIÓN DE MÉTODOS***

    // MÉTODOS DE NEGOCIO
    public boolean esValido(Mensaje m) {
        [...]
    }
}
```

CLASE MENSAJE

```
package servidorSMS;

import java.util.Date;

public class Mensaje {

    // ***SECCIÓN DE ATRIBUTOS***

    // ATRIBUTOS DE BASE
    private String codigo;
    private String telefono;
    private String acronimo;
    private String texto;
    private Date fecha;

    // ATRIBUTO CON ÁMBITO DE CLASE
    private static int sgteCodigo = 0;

    // ***SECCIÓN DE MÉTODOS***

    // MÉTODO CON ÁMBITO DE CLASE
    public static String generarCodigo(){
        [...]
    }
}
```

CLASE PARSERMENSAJES

```
package servidorSMS;

public class ParserMensajes {

    // MÉTODO CON ÁMBITO DE CLASE
    public static Mensaje parsearMensaje(String telf, String texto){
        [...]
    }
}
```

Evidentemente un concepto fundamental ya mencionado es el *mensaje,* cuyas propiedades son número de teléfono, el acrónimo del concurso al que va dirigido el mensaje, el texto que el emisor envía, la fecha y hora del envío y un código único. Un concurso está relacionado con todos sus mensajes válidos; es decir, una vez se dispone del objeto mensaje, debe comprobarse que cumple con las restricciones del concurso al que va dirigido. Sólo en este caso, se vincula con dicho concurso.

Consideraciones de Diseño – Nuevas Clases de Diseño y Operaciones a Añadir

1. Prever la necesidad de una operación en la clase *Concurso* denominada *generarCódigo()* que devuelva un código correlativo de concurso y lleve como prefijo una letra 'C'; es decir, C1, C2, etc. Aplica la misma solución a la clase *Mensaje*.

2. La signatura de la operación de la restricción validar es: *esValido (mensaje): boolean*; ten en cuenta que cada tipo de restricción debe tener su propia implementación de esta operación. Modélalo.

3. La construcción del objeto Mensaje no es trivial. Los datos que se capturan en la interfaz (Figura 1) son un número de teléfono y una secuencia de caracteres en la que se encuentra el acrónimo del concurso al que va dirigido el mensaje y el texto que el emisor envía. Para convertir esta información de entrada en un *mensaje*, se dispone de una clase denominada *ParserMensajes*, que ofrece un servicio: recibe como argumentos el número de teléfono y la secuencia de caracteres recibida y devuelve un objeto mensaje.

4. Un requisito impuesto es la existencia de una clase, denominada *GestorConcursos*, que se encargará de las operaciones de búsqueda de los objetos iniciales (en este caso, buscar un concurso dado un acrónimo) y será el punto de acceso a los servicios de esta solución software Esta clase implementa una interfaz llamada *IGestorConcursos* donde se declaran los servicios públicos que la clase *GestorConcursos* ofrece

5. Para poder acceder a los servicios de este dominio, se representa el prototipo de interfaz de la Figura 1 mediante una clase de presentación/aplicación que se denomina *VentanaSMS*, que tendrá como atributos los datos que se capturan en dicha ventana, además de un vínculo con el objeto *GestorConcursos*.

Paso 2 – Preguntas Interesantes

Piensa las respuestas pero no las escribas

1. ¿Cómo has representado la categorización de restricciones? ¿Qué tiene de especial la clase *Restricción*? Comprueba cómo se cumple Riel 5.10. ¿Qué tiene de especial la operación *esVálido()*? ¿Cómo se denomina este concepto? ¿cómo se modela? Revisa que tu modelo lo contempla de manera adecuada.

2. Atributos y Operaciones de Clase. ¿Los has modelado correctamente?

3. Considerando que la clase de utilidad *ParserMensajes* ofrece un servicio único, no tiene atributos y ni siquiera necesita ser instanciada; es decir, NO hay objetos. ¿Cómo se debe modelar la operación que ofrece? Revisa tu diagrama

4. Solo los mensajes válidos están unidos al Concurso. Los mensajes inválidos se descartan. ¿Cómo puedes modelar en UML esta situación específica?

5. Considerando el uso que hace *GestorConcursos* de la clase de utilidad *ParserMensajes*, ¿cuál es la forma

   correcta de modelar su relación? Revisa tu diagrama

6. *GestorConcursos* tiene una operación de búsqueda. Ponla en tu diagrama y piensa en cómo modelar el atributo de búsqueda.

7. *VentanaSMS* es una clase de presentación/aplicación. Utiliza un estereotipo para añadir esta información a tu modelo.

Paso 3 - TAREAS

1. Con la información proporcionada en los *Detalles del Dominio, genera una versión inicial del Modelo del Dominio* (diagrama de clases inicial). Tiempo estimado: 15 minutos.

Solución

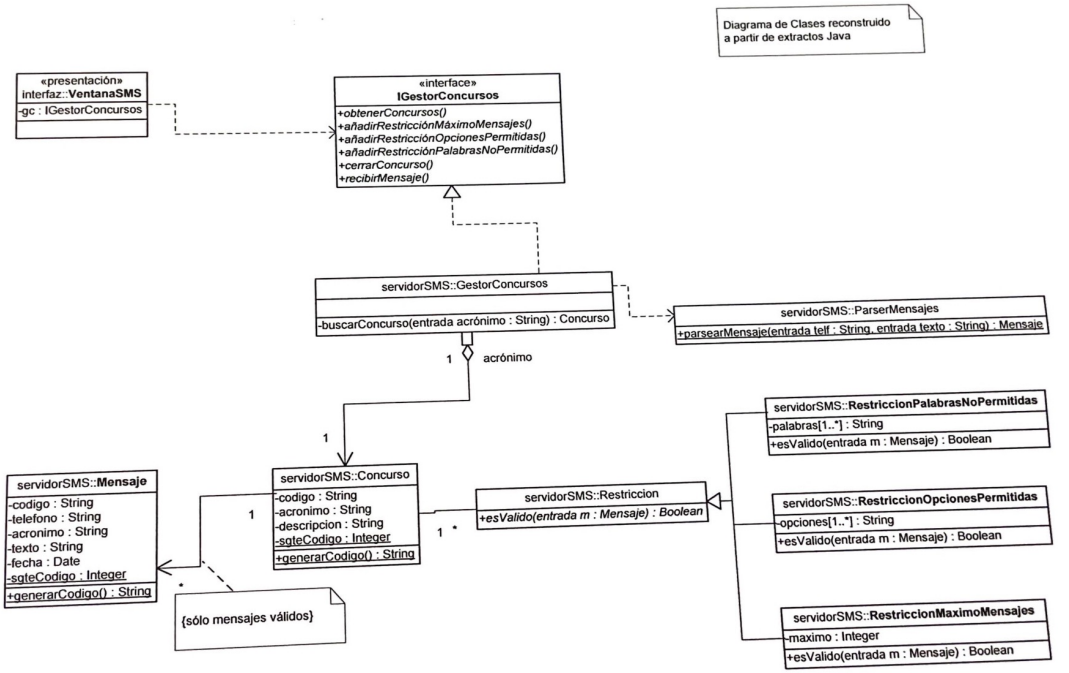


Diagrama de Clases reconstruido
a partir de extractos Java
«presentación»
interfaz::VentanaSMS
-gc : IGestorConcursos
«interface»
IGestorConcursos
+obtenerConcursos()
+añadirRestricciónMáximoMensajes()
+añadirRestricciónOpcionesPermitidas()
+añadirRestricciónPalabrasNoPermitidas()
+cerrarConcurso()
+recibirMensaje()
servidorSMS::GestorConcursos
-buscarConcurso(entrada acrónimo : String) : Concurso
servidorSMS::ParserMensajes
+parsearMensaje(entrada telf : String, entrada texto : String) : Mensaje
1
acrónimo
1
servidorSMS::Mensaje
-codigo : String
-telefono : String
-acronimo : String
-texto : String
-fecha : Date
-sgteCodigo : Integer
+generarCodigo() : String
1
*
servidorSMS::Concurso
-codigo : String
-acronimo : String
-descripcion : String
-sgteCodigo : Integer
+generarCodigo() : String
1
*
servidorSMS::Restriccion
+esValido(entrada m : Mensaje) : Boolean
servidorSMS::RestriccionPalabrasNoPermitidas
-palabras[1..*] : String
+esValido(entrada m : Mensaje) : Boolean
servidorSMS::RestriccionOpcionesPermitidas
-opciones[1..*] : String
+esValido(entrada m : Mensaje) : Boolean
servidorSMS::RestriccionMaximoMensajes
-maximo : Integer
+esValido(entrada m : Mensaje) : Boolean
{sólo mensajes válidos}

2. Usando los puntos numerados en la sección de *Consideraciones del Diseño*, genera el *Diagrama de Clases del Diseño de Alto Nivel* más completo y detallado posible. Revisa a fondo el (paso 2) *Preguntas Interesantes*, ya que deben tenerse en cuenta y pueden ayudarte a refinar tu diagrama de clases. Tiempo estimado: 30 minutos.