

Universidad Tecnológica Nacional

Facultad Regional Avellaneda



Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos

Materia: Laboratorio de Programación II

Apellido:		Fecha:	28/05/2019
Nombre:		Docente ⁽²⁾ :	F. Dávila / D. Boullon
División:	2ºD	Nota ⁽²⁾ :	
Legajo:		Firma ⁽²⁾ :	
Instancia ⁽¹⁾ :	<div style="display: flex; justify-content: space-around; align-items: center;"> <div>PP</div> <div></div> <div>RPP</div> <div>X</div> <div>SP</div> <div></div> <div>RSP</div> <div></div> <div>FIN</div> <div></div> </div>		


(1) Las instancias validas son: 1º Parcial (PP), Recuperatorio 1º Parcial (RPP), 2º Parcial (SP), Recuperatorio 2º Parcial (RSP), Final (FIN). Marque con una cruz.

(2) Campos a ser completados por el docente.

IMPORTANTE:

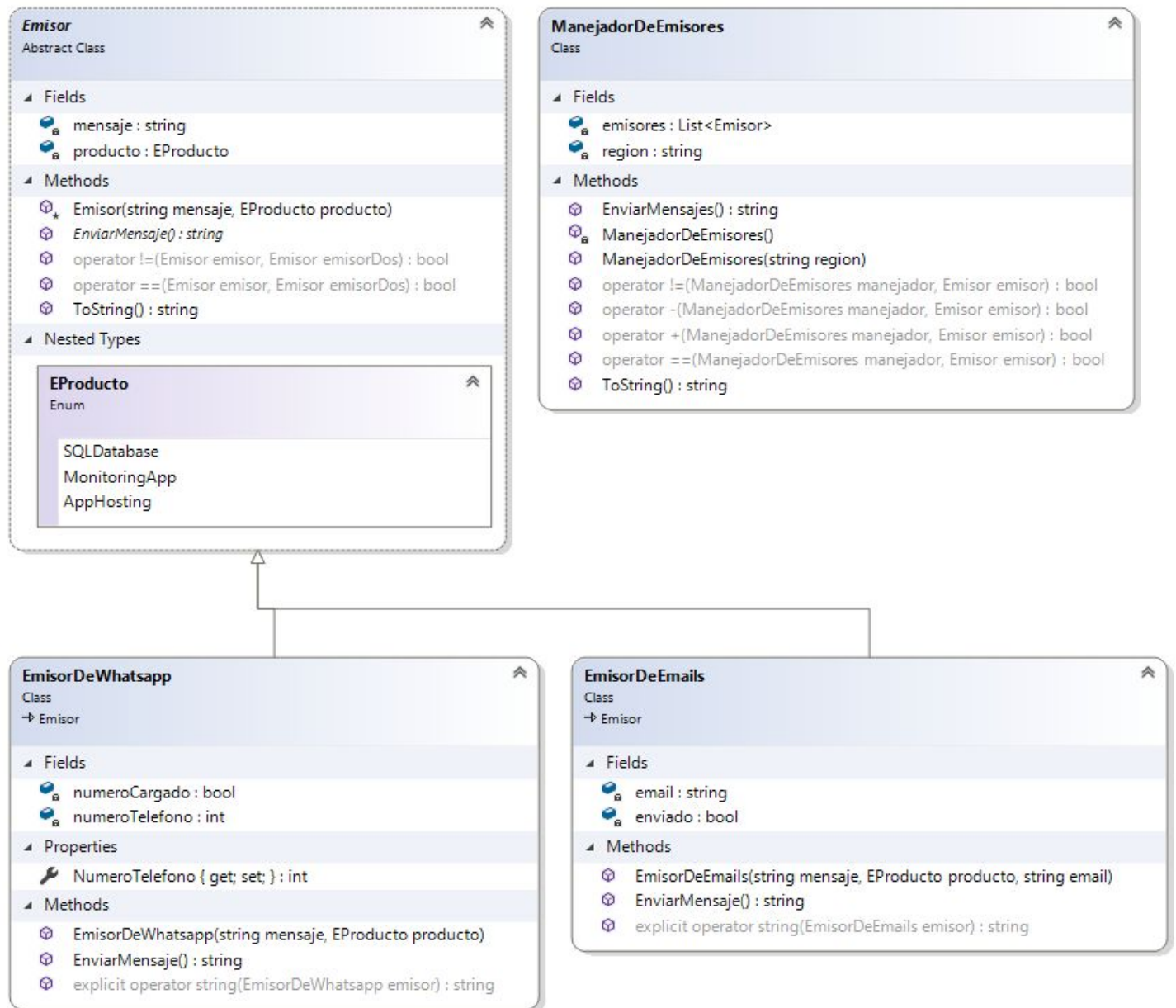
- Guardar el proyecto en el **disco D:**. Ante un corte de energía o problema con el archivo de corrección, el alumno será responsable de que el proyecto sea recuperable.
- **2 (dos) errores en el mismo tema anulan su puntaje.**
- **Errores de conceptos de POO anulan el punto.**
- **Cada tema vale 1 (un) punto (Herencia, Generics, Test Unitarios, etc.). La correcta documentación también será evaluada.**
- **Se deberán tener al menos el 60% bien de los temas a evaluar según la instancia para lograr la aprobación.**
- Colocar sus datos personales en el nombre del proyecto principal, colocando: Apellido.Nombre.AñoCursada. Ej: Pérez.Juan.2018. No se corregirán proyectos que no sea identificable su autor.
- **Salvo que se indique lo contrario, TODAS las clases deberán ir en una Biblioteca de Clases llamada Entidades.**
- No se corregirán exámenes que no compilen.
- **Reutilizar** tanto código como crean necesario.

Al finalizar, colocar la carpeta de la Solución completa en un archivo ZIP que deberá tener como nombre

Apellido.Nombre.AñoCursada.zip y dejar este último en el Escritorio de la máquina. Luego presionar el botón  de la barra superior, **colocar un mensaje** y presionar *Aceptar*. **Aguardar a que el profesor indique que el examen fue copiado de forma correcta.** Luego retirarse del aula.

TIEMPO MÁXIMO PARA RESOLVER EL EXAMEN 90 MINUTOS.

1. Crear una Solución llamada "20190521 – [Apellido].[Nombre]" siendo el Apellido y Nombre del alumno que lo crea.
2. Dentro crear 3 proyectos: uno del tipo *Consola* llamado **RPP**, otro de *Formulario* llamado **Sender** y un último de *Biblioteca de Clases* con el nombre de **Entidades**.
3. Modelar los elementos necesarios para cumplir con el siguiente diagrama de clases:



4. **Emisor**: **clase abstracta** con dos atributos privados.
 - a. Dos emisores serán iguales si tienen el mismo mensaje y producto.
 - b. **EnviarMensaje**: metodo abstracto.
 - c. **ToString**: Sobrecarga que retorna: " \n.{producto} \n. {mensaje}" (usar `String.Format`). Ej:
".MonitoringApp
.3 meses gratis de servicio de monitoreo"
 - d. Constructor protegido que recibe producto y mensaje.
 - e. Definir enum **EProducto** con `SQLDatabase`, `MonitoringApp`, `AppHosting`.
5. **EmisorDeEmails**: hereda de **Emisor** y tiene dos atributos privados.
 - a. Tiene un solo constructor que recibe mensaje, producto y email, y setea el atributo "enviado" en false.
 - b. Conversión implícita a string retornando toda su información utilizando **StringBuilder**.
 - c. **EnviarMensaje** simula el envío del mensaje (no hace nada) y retorna toda su información (en el mismo formato que en el punto anterior) más el texto " \n.Se envia el mensaje" o " \n.Error, el mensaje ya fue enviado" si el mismo ya fue enviado (usar **StringBuilder**). Cambia el valor de **enviado** a **True**
6. **EmisorDeWhatsapp**: hereda de **Emisor**.
 - a. Tiene un solo constructor que recibe mensaje y producto.
 - b. El atributo **numeroCargado** indica si el número fue cargado.

- c. La propiedad **NumeroTelefono** permite cargar un número de teléfono siempre que esté entre 1500000000 y 1599999999, si el valor ingresado es incorrecto no hace nada y el atributo número cargado permanece en **false**.
- d. Tiene una conversión explícita a string que retorna toda su información (**usar StringBuilder**). Si no está cargado el número telefónico en vez del mismo retorna el texto “.No cargado”.
- e. **EnviarMensaje** retorna la información del objeto (igual al punto anterior) más el texto “ Enviando mensaje” o “No se pudo enviar el mensaje” dependiendo del atributo booleano numeroCargado.

7. **ManejadorDeEmisores:**

- a. El constructor privado inicializará la lista.
- b. Tiene un constructor público que recibe la región.
- c. El operador **==** entre un **ManejadorDeEmisores** y **Emisor** retornará true si el elemento ya forma parte del manejador.
- d. El operador **+** agrega un nuevo emisor en el caso de que no lo contenga y retorna un booleano indicando el éxito de la operación.
- e. El operador **-** quita un emisor siempre que se encuentre dentro de la lista de emisores, al igual que **+** retorna un booleano indicando el éxito de la operación.
- f. **ToString** retorna el nombre de la región y la información de todos los emisores (**usar StringBuilder**).
- g. **EnviarMensajes** usa todos sus emisores para tratar de enviar los mensajes. Retorna las respuestas de todos los mensajes junto con el nombre de la región.

8. Agregar en el Main el siguiente código:

```
List<Emisor> emisores = new List<Emisor>();
Emisor emisorEmail = new EmisorDeEmails("Licencia por caducar", Emisor.EProducto.AppHosting, "pepito@pepe.pepe");
emisores.Add(emisorEmail);
emisores.Add(new EmisorDeEmails("3 meses gratis de servicio de monitoreo.", Emisor.EProducto.MonitoringApp,
"pepito@pepe.pepe"));
EmisorDeWhatsapp emisorWhatsapp = new EmisorDeWhatsapp("Usted ha adquirido servicio de Azure SQL Base de Datos",
Emisor.EProducto.SQLDatabase);
emisores.Add(emisorWhatsapp);
Console.ForegroundColor = ConsoleColor.Yellow;
ManejadorDeEmisores manejador = new ManejadorDeEmisores("West Us");
foreach (Emisor emisor in emisores)
{
    if (manejador + emisor)
        Console.WriteLine("Emisor agregado.");
    else
        Console.WriteLine("No se pudo agregar emisor.");
}

Console.ForegroundColor = ConsoleColor.Red;
if (manejador + emisorEmail)
    Console.WriteLine("Se agregó el emisor.");
else
    Console.WriteLine("No se agregó al emisor.");

if (manejador - emisorEmail)
    Console.WriteLine("Se eliminó emisor.");
else
    Console.WriteLine("No se eliminó emisor.");

Console.ForegroundColor = ConsoleColor.Yellow;
Console.WriteLine(manejador.EnviaMensajes());
emisorWhatsapp.NumeroTelefono = 1514134118;

Console.ForegroundColor = ConsoleColor.DarkGreen;
Console.WriteLine(manejador.EnviaMensajes());
Console.ReadKey();
```

Logrando la siguiente salida por pantalla:

```

Emisor agregado.
Emisor agregado.
Emisor agregado.
No se agregó al emisor.
Se eliminó emisor.

Region: West Us

.MonitoringApp
.3 meses gratis de servicio de monitoreo.
.pepito@pepe.pepe
.Se envia el mensaje

.SQLDatabase
.Usted ha adquirido servicio de Azure SQL Base de Datos
.No cargado
.No se pudo enviar el mensaje

Region: West Us

.MonitoringApp
.3 meses gratis de servicio de monitoreo.
.pepito@pepe.pepe
.Error, el mensaje ya estaba enviado

.SQLDatabase
.Usted ha adquirido servicio de Azure SQL Base de Datos
.1514134118
.Enviando mensaje

```

9. Generar el siguiente formulario, logrando la misma funcionalidad:

- Los objetos se instancian en el constructor (**El form solo instancia Whatsapps**).
- La región por default tiene que ser "West Europe".
- El título del formulario serán los datos del alumno.
- Agregar** agrega mensajes al manejador y muestra un MessageBox indicando si se pudo agregar.
- Enviar** trata de enviar todos los mensajes y muestra la información en el rich box.

**importante - Código para cargar comboBox con productos:*

`this.comboBoxProductos.DataSource = Enum.GetNames(typeof(EProducto));`

**importante - Código para leer del databox y parsear a EProducto:*

`EProducto producto;`

`Enum.TryParse(this.comboBoxProductos.SelectedItem.ToString(), out producto);`

The screenshot displays a Windows Forms application with a form titled "Apellido.Nombre.Curso". The form contains a "Sender" section with three input fields: "Mensaje" (a text box), "Producto" (a dropdown menu), and "Email" (a text box). Below these fields are two buttons: "Enviar" and "Agregar". The "Agregar" button is disabled. Below the "Sender" section is a large empty text area.

The Properties window on the right shows the "FrmSender" class. The "Fields" section lists the following properties:

- btnAgregar : Button
- btnEnviar : Button
- comboBoxProductos : ComboBox
- components : IContainer
- groupBox1 : GroupBox
- lblEmail : Label
- lblMensaje : Label
- lblProducto : Label
- manejador : ManejadorDeEmisores
- productos : List<EProducto>
- richMensaje : RichTextBox
- ritchConsola : RichTextBox
- textEmail : TextBox

The "Methods" section lists the following methods:

- BtnAgregar_Click(object sender, EventArgs e) : void
- BtnEnviar_Click(object sender, EventArgs e) : void
- Dispose(bool disposing) : void
- FrmSender()
- InitializeComponent() : void