Universidad Tecnológica Nacional Facultad Regional Avellaneda										
Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos										
Materia: Laboratorio de Programación II										
Apellido:			Fecha:		30/10/2018					
Nombre:			Docente(2):		F. Dávila/D.Boullon					
División:	2ºD		Nota(2):							
Legajo:			Firma(2):							
Instancia(1):	**PP**		**RPP**	X	**SP**		**RSP**		**FIN**	

(1) Las instancias validas son: 1er Parcial (**PP**), Recuperatorio 1er Parcial (**RPP**), 2do Parcial (**SP**), Recuperatorio 2do Parcial (**RSP**), Final (**FIN**). Marque con una cruz.
(2) Campos a ser completados por el docente.

IMPORTANTE:

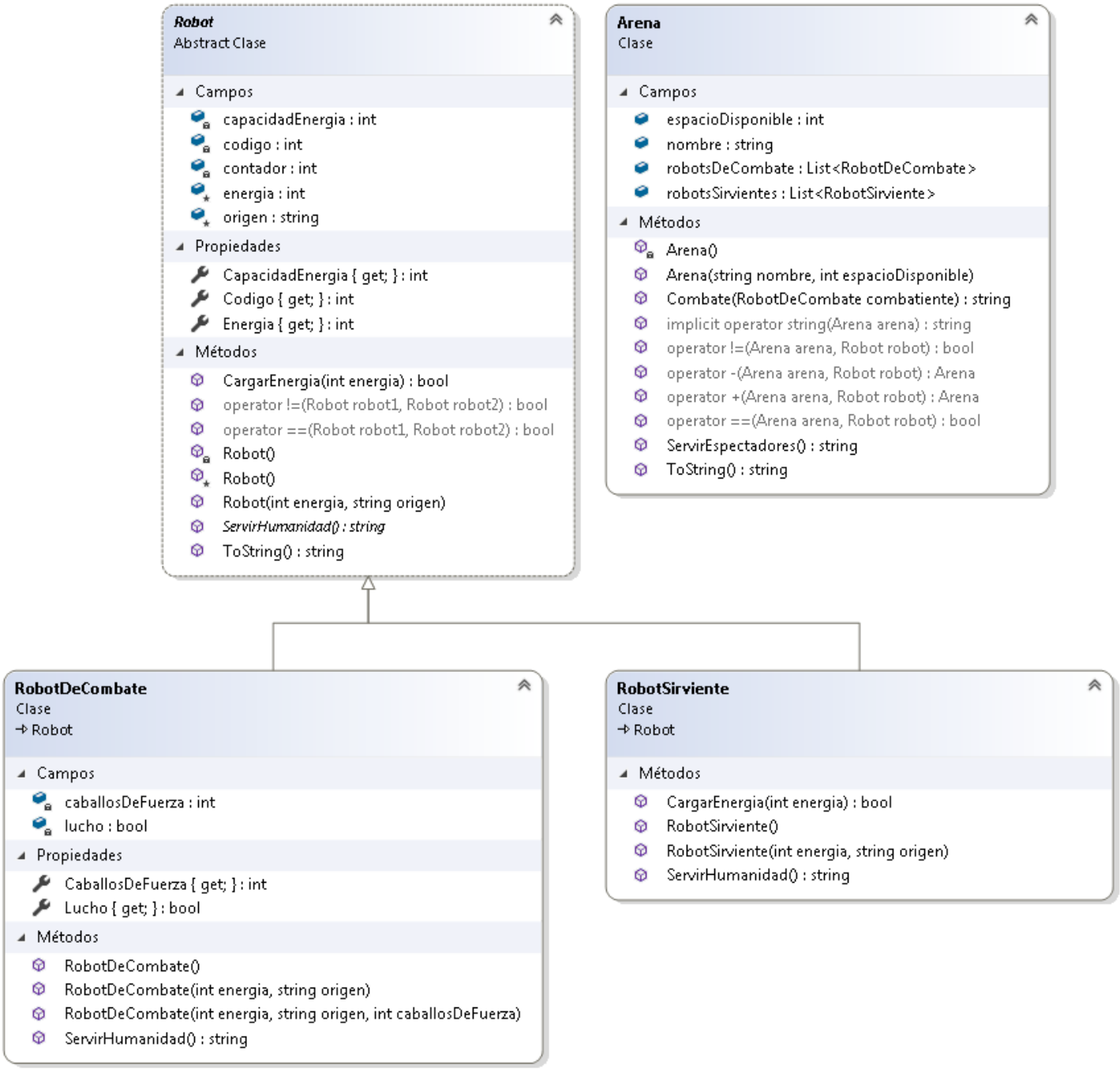
- ***2 (dos) errores en el mismo tema anulan su puntaje.***
- **La correcta documentación y reglas de estilo de la cátedra serán evaluadas.**
- Colocar sus datos personales en el nombre del proyecto principal, colocando: Apellido.Nombre.Departamento. Ej: Pérez.Juan.2D. No sé corregirán proyectos que no sea identificable su autor.
- **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.
- Colocar nombre de la clase (en estáticos), **this** o **base** en todos los casos que corresponda.

---

*TIEMPO MÁXIMO PARA RESOLVER EL EXAMEN **90** MINUTOS.*

- Crear una solución con el nombre en el siguiente formato: [APELLIDO].[NOMBRE]
- Dentro crear 3 proyectos: *Entidades* (Class Library), *VistaConsola* (Console) y *VistaForm* (WindowsForms).

El ejercicio consiste en armar algunas entidades para una Arena de lucha de robots. Dentro de la biblioteca Entidades, diagramar las siguientes clases:



Clase Abstracta Robot

- **Atributos:**
  - Atributo protegido (ESTÁTICO) **capacidadEnergia** (int).
  - Atributo privado (ESTÁTICO) **contador**(int).
  - Atributo privado **codigo** (int).
  - Atributo protegido **energia** (int).
  - Atributo protegido de tipo **origen**(string).
- **Métodos:**
  - Método público virtual **CargarEnergia** que recibirá una cantidad de energía y validará que sea mayor a cero y menor a la **capacidadEnergia** y lo asignará. Si puede asignarlo, retornará true.
  - Método abstracto ServirHumanidad que no recibe parámetros y retorna un booleano.
- Sobrecarga del operador **==** comparará dos robots por el atributo **codigo**.

- **Constructores:**
  - Constructor de clase que cargará los atributos estáticos de capacidadEnergía en 50 y contador en 0.
  - Constructor de instancia sin parámetros que cargará origen como "Coreano", energía en 10 y aumentará en 1 al contador y le asignará dicho valor al atributo **codigo**. (El código es único e irrepetible para cada robot).
  - Constructor de instancia que recibirá dos parámetros: energía y origen.

RoborDeCombate

- **Atributos:**
  - Atributo privado **lucho** (bool), que indicará si luchó o no contra otro robot. Tendrá una propiedad de solo lectura.
  - Atributo privado de tipo **caballosDeFuerza** (int). Tendrá una propiedad de solo lectura.
- **Métodos:**
  - El método **ServirHumanidad**, en caso de que tenga energía, descontará en uno la misma, y retornará un mensaje "Robot De combate [codigo] - Disparando misiles..." caso contrario retornará "Robot De combate [codigo] - Sin energía".
- **Constructores:**
  - **Constructor de instancia** que no recibe parámetros y inicializa el atributo lucho en false;
  - **Constructor de instancia** que recibe energía y origen, y recibe los mismos parámetros. Pero este inicializa **caballosDeFuerza** en 10 y **lucho** en false;
  - **Constructor de instancia que agrega el parámetro caballosDeFuerza.**

RobotSirviente

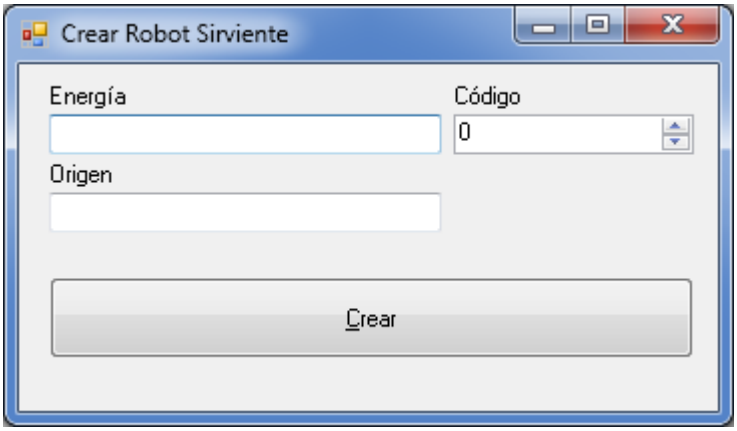
- **Métodos:**
  - Método público **CargarEnergia (que sobrescriba el método de robot)** que recibirá una cantidad de energía y **sólo cargará energía si la misma está en cero**.
  - **ServirHumanidad**, en caso de tener energía, descontará en uno la misma, retornará un mensaje con el código y el mensaje "Haciendo masajes..."; caso contrario el código y "Sin energía..." .
- **Constructores:**
  - Tiene un constructor de instancia que no recibe parámetros (reutilizar código).
  - Tiene un constructor de instancia que recibe energía y origen.

Arena

- **Métodos**:
  - **Combate** en caso de que no sea null y que tenga energía:
    - busca el primer **robotDeCombate** de la lista (con energía) para combatir con el mismo.
    - Ambos usan el método **ServirHumanidad**, armando un string con la información de ambos y declarando como ganador al que tiene más caballos de fuerza.
    - En caso de que sean iguales, debe retornar la información de ambos y declarar un empate.
    - Si no pudieron combatir debe retornar *"No se encontró oponente"*.
  - **ServirEspectadores** buscará el primer **robotSirviente** que tenga energía y utilizará el método **ServirHumanidad**. Sino retornará un vacio ("").
  - **ToString()** retorna un string con el nombre de la Arena y una lista de Robots de combates y otra de robots sirvientes, mostrando la información de su método ServirHumanidad().
  - **implicit** retorna un string con el nombre de la Arena y una lista de Robots de combates y otra de robots sirvientes.
- **Operadores:**
  - El operador == recibe una Arena y un Robot, retorna true si la Arena contiene al robot en alguna de sus dos lisas.
  - El operador + recibe una Arena y un Robot y agrega al Robot (Combate o Sirviente) en la lista que corresponda (Sirviente o Combate) en caso de que no se encuentre en su lista.
  - El operador - recibe una Arena y un Robot y retira al Robot en caso de que se encuentre en alguna de las dos listas.

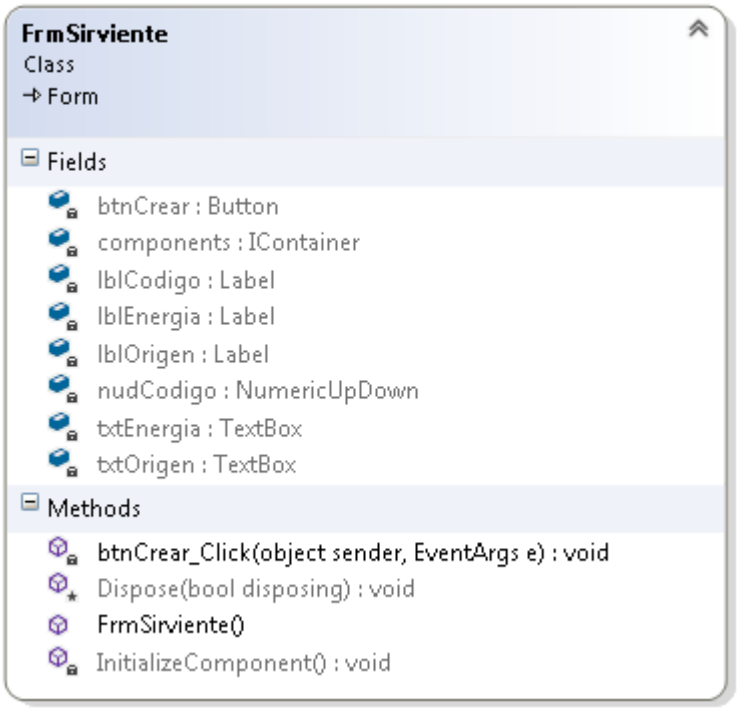
Por último, generar el siguiente formulario dentro de *VistaForm*:
Siendo los elementos a utilizar Button, Label, NumericUpDown y TextBox.

- Crear el siguiente formulario:



- Botón **Crear** se deberá generar un nuevo objeto **RobotSirviente**, mostrar su información en un MessageBox.
- Asociar el evento del botón al método ya creado en el formulario.
- Al ejecutar el formulario éste deberá aparecer centrado en la pantalla.

El diagrama de clases deberá lucir exactamente como este:



Al finalizar, colocar la carpeta de la Solución completa en un archivo ZIP que deberá tener como nombre Apellido.Nombre.division.zip y dejar este último en el Escritorio de la máquina.

Luego presionar el botón de la barra superior, colocar un mensaje y apretar **Aceptar.**
Finalmente retirarse del aula y aguardar por la corrección.