Universidad Tecnológica Nacional
Facultad Regional Avellaneda

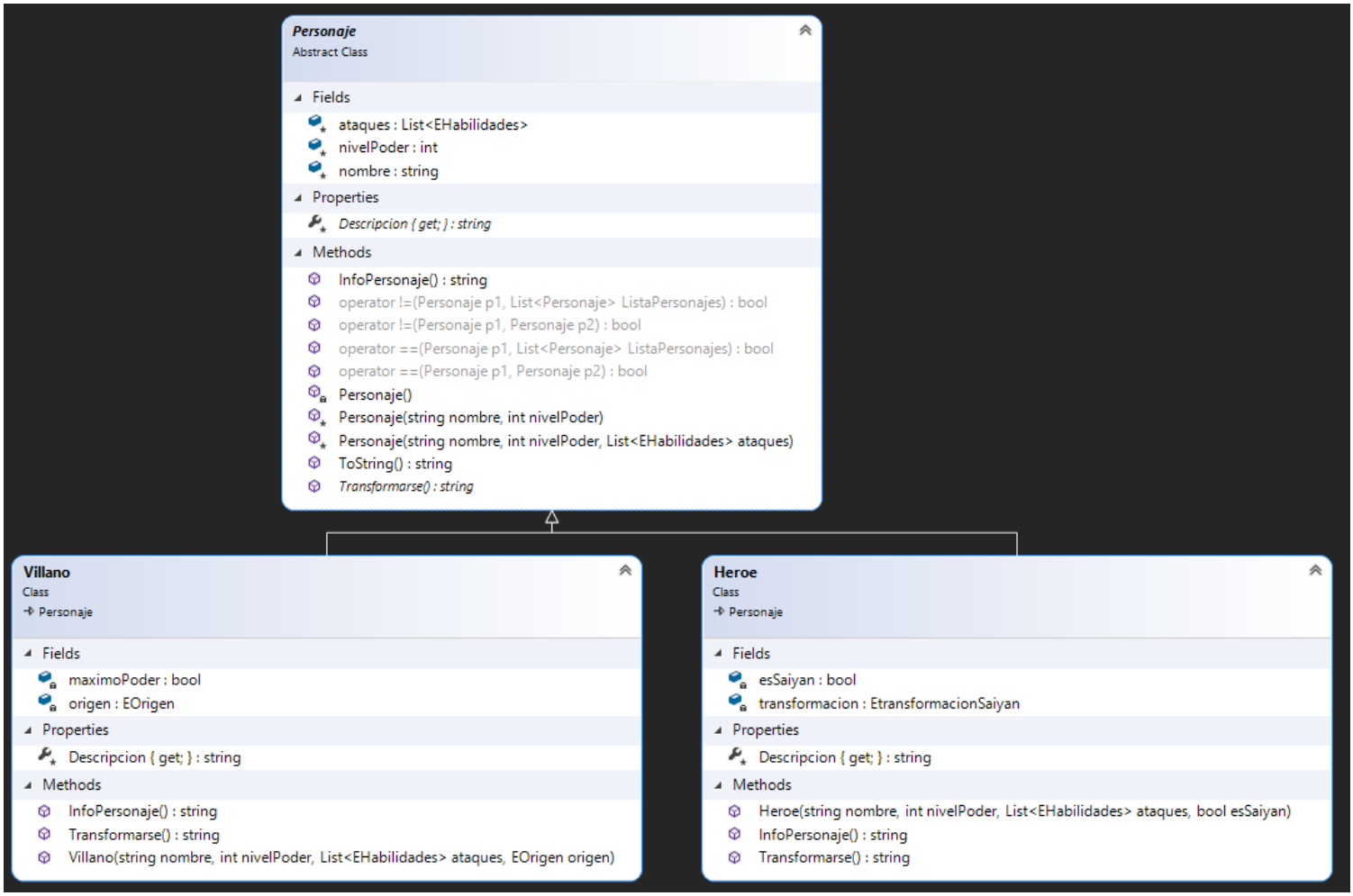
Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos										
Materia: Laboratorio de Programación II										
Apellido:					Fecha:		03-12-2020			
Nombre:					Docente(2):		Dávila-Oggioni-Rodríguez			
División:	2°C – 2°D				Nota(2):					
Legajo:					Firma(2):					
Instancia(1):	**PP**		**RPP**	X	**SP**		**RSP**		**FIN**	

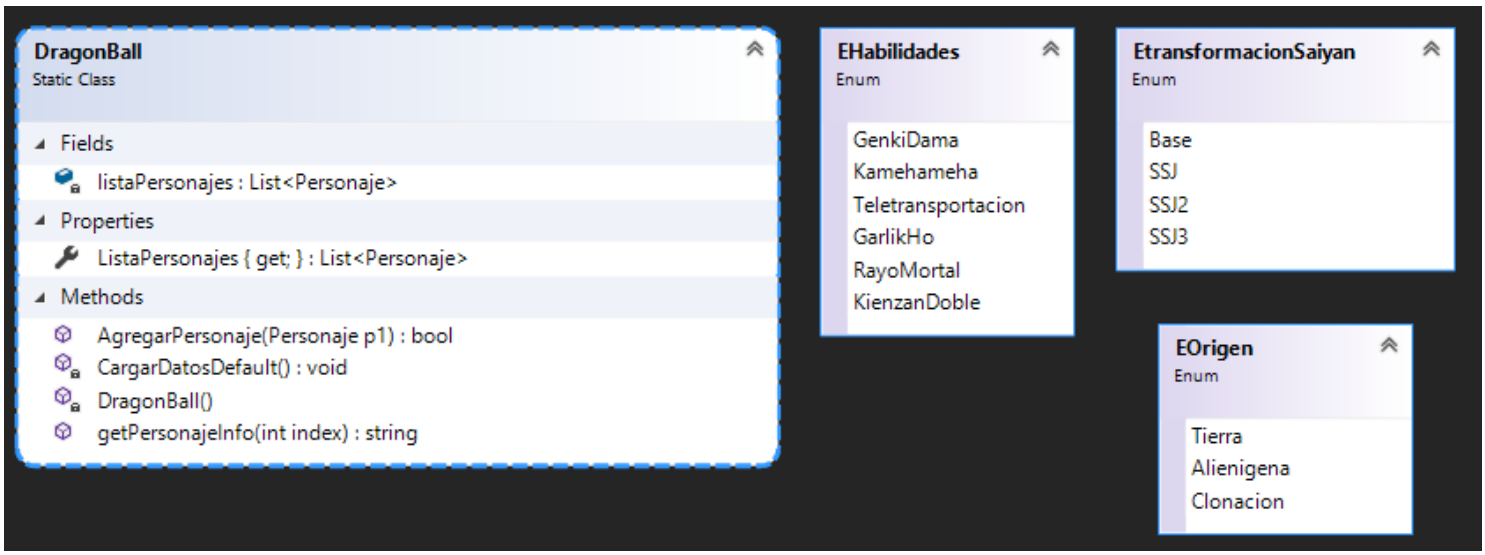
(1) Las instancias validas son: 1er Parcial (**PP**), Recuperatorio 1er Parcial (**RPP**), 2do Parcial (**SP**), Recuperatorio 2do Parcial (**RSP**), Final (**FIN**). Marque con una cruz.
(2) Campos a ser completados por el docente.

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 de la carpeta principal y la solución: Apellido.Nombre.Div. Ej: Pérez.Juan.2D. No sé corregirán proyectos que no sea identificable su autor.
- 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.
- Aplicar los principios de los 4 pilares de la POO.

1. Crear un proyecto del tipo Biblioteca de Clases y colocar el siguiente esquema de clases:


Personaje
Abstract Class
Fields
ataques : List<EHabilidades>
nivelPoder : int
nombre : string
Properties
Descripcion { get; } : string
Methods
InfoPersonaje() : string
operator !=(Personaje p1, List<Personaje> ListaPersonajes) : bool
operator !=(Personaje p1, Personaje p2) : bool
operator ==(Personaje p1, List<Personaje> ListaPersonajes) : bool
operator ==(Personaje p1, Personaje p2) : bool
Personaje()
Personaje(string nombre, int nivelPoder)
Personaje(string nombre, int nivelPoder, List<EHabilidades> ataques)
ToString() : string
Transformarse() : string
Villano
Class
Personaje
Fields
maximoPoder : bool
origen : EOrigen
Properties
Descripcion { get; } : string
Methods
InfoPersonaje() : string
Transformarse() : string
Villano(string nombre, int nivelPoder, List<EHabilidades> ataques, EOrigen origen)
Heroe
Class
Personaje
Fields
esSaiyan : bool
transformacion : EtransformacionSaiyan
Properties
Descripcion { get; } : string
Methods
Heroe(string nombre, int nivelPoder, List<EHabilidades> ataques, bool esSaiyan)
InfoPersonaje() : string
Transformarse() : string


DragonBall
Static Class
Fields
listaPersonajes : List<Personaje>
Properties
ListaPersonajes { get; } : List<Personaje>
Methods
AgregarPersonaje(Personaje p1) : bool
CargarDatosDefault() : void
DragonBall()
getPersonajeInfo(int index) : string
EHabilidades
Enum
GenkiDama
Kamehameha
Teletransportacion
GarlikHo
RayoMortal
KienzanDoble
EtransformacionSaiyan
Enum
Base
SSJ
SSJ2
SSJ3
EOrigen
Enum
Tierra
Alienigena
Clonacion

2. Clase **Personaje**:
    a. Los métodos y propiedades marcados en cursiva son abstractos.
    b. InfoPersonaje: retornará los datos del personaje utilizando StringBuilder y String.Format.
        i. Debera retornar:
            - Nombre
            - Lista de ataques
            - Nivel de poder
            - Descripción (es una propiedad, ver más abajo).

    c. Los operadores deberán:
        i. Verificar si un personaje existe en la lista de personajes
        ii. Verificar si un personaje es igual al otro comparando por tipo de dato (GetType) y nombre
    d. ToString:Solo devolverá el nombre del personaje.

3. Clase **Villano**:
    a. Descripción: Es una propiedad de tipo string que solo devolverá:
    "Soy Malísimo. Diabólico. Así como los profes de labo de la noche";
    b. Transformarse: Este método, en caso de que la variable maximoPoder sea falsa, aumentara el valor de nivelPoder en un 80%, cambiará el valor de la variable maximoPoder y devolverá el mensaje
    "Poder aumentado al máximo".
    Caso contrario, devolverá "El poder ya está al límite".
    c. InfoPersonaje: Sumará los datos que falten al InfoPersonaje de la clase base, y retornará la información.

4. Clase **Heroe**:
    a. Descripción: Es una propiedad de tipo string que devolverá:
        i. Si es saiyan: "Disfruta los combates. Su poder no tiene limite"
        ii. Si no lo es: "Siempre pelea junto a un Saiyan.Fiel amigo"

    b. Transformarse: Este método, en caso de que la variable esSaiyan sea verdadera, se capturará la transformación actual en la que está el personaje, y se subirá a la siguiente. Es decir, si el personaje es Goku, y está en SSJ, deberá subir a SSJ2.
    El orden de las transformaciones es siguiente:
        1. Base: setea el nivelPoder en 100.
        2. SSJ: multiplica el nivel de poder por 10.
        3. SSJ2: multiplica el nivel de poder por 20.
        4. SSJ3: multiplica el nivel de poder por 30.

    Nota: Si un personaje quiere volver a transformarse estando en nivel SSJ3, volverá a estado base-.
    c. InfoPersonaje: Sumará los datos que falten al InfoPersonaje de la clase base, y retornará la información.

5. Clase **Static DragonBall**:
    a. La lista se inicializará en el constructor privado.
    b. El Tipo de campo será estático.
    c. AgregarPersonaje: Permitirá agregar un personaje siempre y cuando no exista en la lista previamente. Utilizará las sobrecargas de operadores de la clase Personaje.
    d. CargarDatos: Tendrá allí algunos Personajes hardcodeados. Asegurarse de usarlo antes de levantar la aplicación. No mediante botón. La carga tiene que ser automática.
    e. GetInfoPersonaje: Recibirá el índice del personaje seleccionado en el ListBox, y dentro retornará la información completa de ese personaje.

6. El formulario principal estará adjunto en junto con el parcial. En el deberán modificar:
    a. Título del formulario con los datos que pide el mismo.
    b. Deberán darle la funcionalidad al botón AgregarPersonaje para que puedan agregarse más personajes mediante un form.