

**Universidad Tecnológica Nacional
Facultad Regional Avellaneda**



Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos

Materia: Laboratorio de Programación II

Apellido:		Fecha:	05/10/2017
Nombre:		Docente ⁽²⁾ :	F. Dávila
División:	2°C	Nota ⁽²⁾ :	
Legajo:		Firma ⁽²⁾ :	
Instancia ⁽¹⁾ :	<div style="display: flex; justify-content: space-around;"> PP X RPP </div>	<div style="display: flex; justify-content: space-around;"> SP RSP FIN </div>	

(1) Las instancias validas son: 1^{er} Parcial (**PP**), Recuperatorio 1^{er} Parcial (**RPP**), 2^{do} Parcial (**SP**), Recuperatorio 2^{do} Parcial (**RSP**), Final (**FIN**). Marque con una cruz.

(2) Campos a ser completados por el docente.

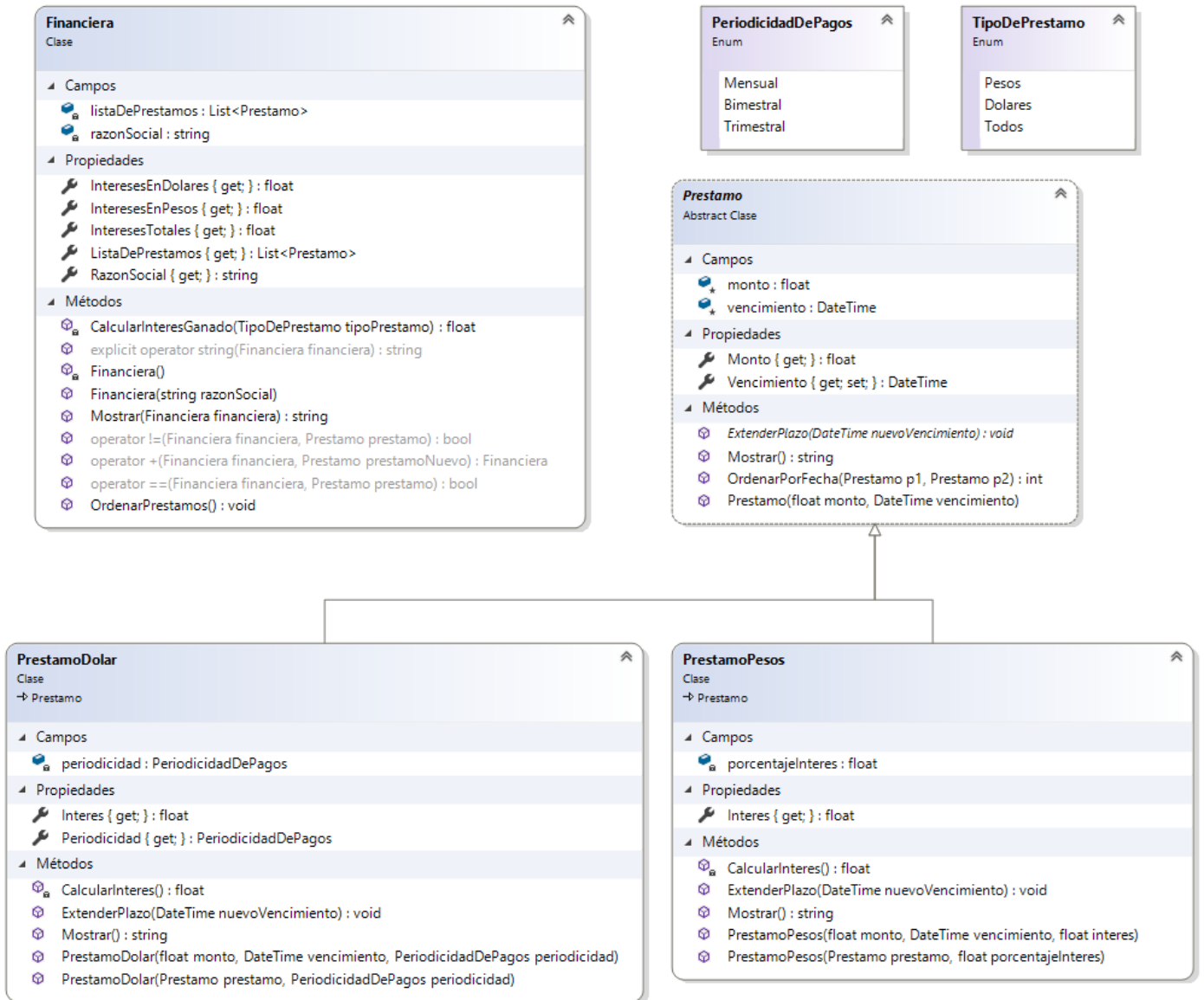
Pautas del examen, leer antes de iniciar el Visual Studio:

- Crear una nueva Solución con el nombre: apellido.nombre.división. Por ejemplo Pérez.Juan.2E.
- Crear un nuevo proyecto del tipo Biblioteca de Clases llamado Entidades.
- Crear un nuevo proyecto del tipo Consola llamado Test dónde sólo estará la clase Program.
- Los proyectos que no sean identificables, no serán corregidos.**
- Los alumnos que no entreguen o su parcial no sea identificable serán desaprobados.**
- Sólo se corregirá lo que el alumno entregue de la siguiente forma:
 - Al finalizar, colocar la carpeta de la Solución completa en un archivo ZIP y dejar este último en el Escritorio de la máquina. Luego presionar el botón de la barra superior, cargar un mensaje y presionar Aceptar. La barra superior deberá cambiar de color.
- En todos los casos que sea posible, reutilizar código.
- Los proyectos deberán compilar.

Prestamos Financieros

El objetivo de la aplicación será llevar un control de los préstamos otorgados por una entidad Financiera y sus ganancias obtenidas.

- Crear una biblioteca de clases llamada EntidadFinanciera que contenga un proyecto de consola respetando el siguiente diagrama de clases:



Prestamo:

2. Crear la clase dentro de un NameSpace llamado **PrestamosPersonales**
3. Los atributos de la clase Prestamo serán protected y deberán existir propiedades de tipo lectura/escritura para acceder a los mismos.
4. La propiedad Vencimiento deberá validar que el vencimiento sea posterior a la fecha actual antes de cargar el valor, caso contrario cargarlo con la fecha actual.
5. La clase Prestamo contendrá los siguientes métodos:
 - a. Constructor: recibirá monto y vencimiento como parámetros.
 - b. **OrdenarPorFecha**: Método estático que recibirá como parámetros 2 préstamos y retornará un entero (Será utilizado en el método Sort de la lista de préstamos en la clase Financiera).
 - c. **ExtenderPlazo**: Método **abstracto** que recibirá un parámetro de tipo DateTime.
 - d. **Mostrar**: Método de instancia y virtual que deberá devolver una cadena que contenga los atributos de la clase.
6. Dentro del mismo NameSpace de la clase Préstamo, se deberá agregar el enumerado **TipoDePrestamo** (Pesos, Dolares, Todos) y el enumerado **PeriodicidadDePago**.

PrestamoPesos y PrestamoDolar:

7. Las clases PrestamoPesos y PrestamoDolar, también pertenecientes al NameSpace

PrestamosPersonales, serán clases derivadas de la clase Préstamo. Sus atributos serán privados y deberán tener propiedades de lectura/escritura para cada uno de ellos.

8. Ambas clases contendrán los siguientes métodos:
 - a. **CalcularInteres**: Método privado de instancia que calculará y retornará el total del préstamo (cantidad de dinero prestado + intereses). Este método será accedido a través de una propiedad de lectura llamada **Interes** cuyo valor será utilizado en la clase Financiera dentro del método **CalcularInteresGanado**.
 - i. En el caso de los préstamos en pesos, el interés se calculara a partir del monto y el porcentaje definido.
 - ii. Para los préstamos en dólares el porcentaje de interés estará dado por la periodicidad de pago (Mensual: 25, Bimestral: 35, Trimestral: 40).
 - b. **ExtenderPlazo**: Deberá implementarse el método abstracto definido en la clase base.
 - i. En el caso de los préstamos en pesos, se deberá aplicar un incremento del 0.25% al interés original por cada día de extendido el plazo y se actualizará la fecha original de vencimiento a la nueva fecha
 - ii. Para los préstamos en dólares se incrementará el monto original en 2.5 dólares por cada día de extendido el plazo y se actualizará la fecha original de vencimiento a la nueva fecha.
 - c. **Mostrar**: Retornará una cadena con los atributos de la clase base, los propios de la clase y la propiedad **Interes**. Utilizar StringBuilder para concatenar los String a mostrar (no utilizar el operador +).

Financiera:

9. Crear la clase dentro de un Namespace llamado **EntidadFinanciera**.
10. Atributos privados con sus correspondientes Propiedades de sólo lectura.
11. Métodos:
 - a. Constructor: Recibirá la razón social y luego inicializará la lista genérica en el constructor por defecto el cual deberá ser private.
 - b. **Mostrar**: Método estático que devolverá una cadena. Deberá reutilizar la conversión explícita.
 - c. Conversión **explícita** a string retornará una cadena que contendrá la razón social, los intereses totales ganados por préstamos otorgados, los intereses por préstamos en pesos y por préstamos en dólares y el detalle de cada préstamo ordenados por fecha (se deberá utilizar StringBuilder para armar la cadena a devolver).
 - d. **OrdenarPrestamos**: Deberá ordenar por fecha de vencimiento la lista de préstamos.
 - e. **CalcularInteresGanado**: Método privado que recibe un Enumerado de tipo *TipoDePrestamo* y retornará el valor equivalente a la suma de intereses entre todos los préstamos (invocar a la propiedad **Interes** de la clase PrestamoPesos o PrestamoDolar según el tipo de préstamo evaluado).
12. Sobrecarga de operadores:
 - a. Se deberán sobrecargar el operador "+" para que permitirá cargar un préstamo a la financiera
 - b. El operador "==" que será utilizado para validar que el que un mismo préstamo no sea cargado más de una vez en la financiera.

Nota: Las propiedades InteresesTotales, InteresesEnPesos y InteresesEnDolares retornarán el interés obtenido según el criterio. Se calculará en el método CalcularInteresGanado().

Main():

```
static void Main(string[] args)
{
    Financiera financiera = new Financiera("Mi Financiera");
    PrestamoDolar pd1 = new PrestamoDolar(1500, new DateTime(2017, 11, 01),
        PeriodicidadDePagos.Mensual);
    PrestamoDolar pd2 = new PrestamoDolar(2000, new DateTime(2017, 12, 05),
        PeriodicidadDePagos.Bimestral);
    PrestamoDolar pd3 = new PrestamoDolar(2500, new DateTime(2018, 01, 01),
```

```

PeriodicidadDePagos.Trimestral);

PrestamoPesos pp1 = new PrestamoPesos(8000, new DateTime(2018, 01, 01), 20);
PrestamoPesos pp2 = new PrestamoPesos(7000, new DateTime(2001, 10, 01), 25);
PrestamoPesos pp3 = new PrestamoPesos(5000, new DateTime(2017, 11, 20), 20);

financiera = financiera + pd1;
financiera = financiera + pd2;
financiera = financiera + pd3;
financiera = financiera + pd3; //Préstamo repetido

financiera = financiera + pp1;
financiera = financiera + pp2;
financiera = financiera + pp3;
financiera = financiera + pp3; //Préstamo repetido

Console.WriteLine((String)financiera);

pd1.ExtenderPlazo(new DateTime(2017,12,01));
pp1.ExtenderPlazo(new DateTime(2018,02,01));

financiera.OrdenarPrestamos();

Console.WriteLine("\n *****ORDENADOS POR FECHA*****");
Console.WriteLine(Financiera.Mostrar(financiera));
Console.ReadKey();
}

```

Verificar que los intereses obtenidos coincidan con lo esperado.