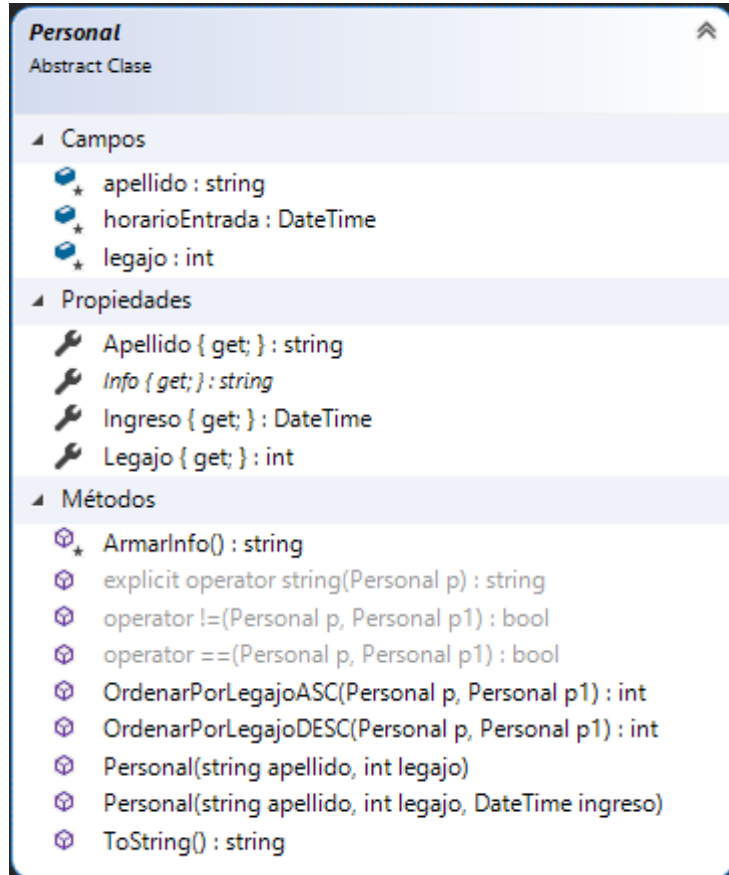


Recuperatorio Laboratorio II 2021 (Primer Parcial)

Administración de Médicos

Esta aplicación nos permitirá controlar el ingreso y egreso de los médicos que trabajan en un hospital. Además se podrá ordenar el listado. Para lograr esto, se pide realizar en un proyecto de tipo 'Class Library' (EntidadesRPP), las clases **Personal**, **Medico**, **MedicoEgresado** y **Hospital** y el enumerado **Especialidad**, respetando los diagramas que se muestran a continuación:

Personal:



Personal
Abstract Class

- Campos
 - apellido : string
 - horarioEntrada : DateTime
 - legajo : int
- Propiedades
 - Apellido { get; } : string
 - Info { get; } : string
 - Ingreso { get; } : DateTime
 - Legajo { get; } : int
- Métodos
 - ArmarInfo() : string
 - explicit operator string(Personal p) : string
 - operator !=(Personal p, Personal p1) : bool
 - operator ==(Personal p, Personal p1) : bool
 - OrdenarPorLegajoASC(Personal p, Personal p1) : int
 - OrdenarPorLegajoDESC(Personal p, Personal p1) : int
 - Personal(string apellido, int legajo)
 - Personal(string apellido, int legajo, DateTime ingreso)
 - ToString() : string

Todos sus atributos son protegidos.

Posee un constructor de instancia sobrecargado (reutilizar código). El valor predeterminado del atributo **horarioEntrada** será el valor de DateTime.Now.

Todas las propiedades son públicas y de sólo lectura y se vincularán con sus respectivos atributos, a excepción de la propiedad abstracta **Info**.

Método de instancia (protegido y virtual) **ArmarInfo**: retornará el apellido y el legajo del personal con el siguiente formato: **Pérez, legajo: 24**, siendo Pérez el apellido del personal y 24 su número de legajo. Se relacionará con la propiedad abstracta.

Polimorfismo en **ToString**, retornará una cadena conteniendo el apellido y el legajo del personal (reutilizar código) y su horario de ingreso, respetando el siguiente formato: **Pérez, legajo: 24 - ingreso: 09:12:18 hs.**, siendo Pérez el apellido del personal, 24 su número de legajo y 09:12:18 la hora, minutos y segundos de su horario de ingreso (utilizar método **ToString**, del atributo de tipo DateTime).

Sobrecarga de operadores:

Explícito. Retornará el legajo (en formato de cadena de texto) del personal que recibe como parámetro.

Igualdad (==) (Personal, Personal). Retornará true, si los apellidos y los legajos son iguales, false, caso contrario.

Reutilizar código.

Métodos de clase:

OrdenarPorLegajoASC y **OrdenarPorLegajoDESC**: Establecerán el criterio de comparación entre los objetos de tipo Personal que reciben como parámetros. Reutilizar código.

Estos métodos se utilizarán para ordenar listas genéricas (método Sort).

Medico
Clase
Personal

Campos

especialidad : Especialidad

Propiedades

Especialidad { get; } : Especialidad

Info { get; } : string

Métodos

ArmarInfo() : string

Medico(string apellido, int legajo, DateTime ingreso, Especialidad especialidad)

ToString() : string

Medico (hereda de Personal):

Posee un único atributo protegido propio, que será inicializado por su único constructor y una propiedad pública y de sólo lectura asociada a dicho atributo.

Sobrescribir el método *ArmarInfo* para que retorne la siguiente cadena de texto:

DOCTOR – Pérez, legajo: 24, especialidad: Pediatra, siendo Pérez el apellido, 24 su número de legajo y Pediatra la especialidad del médico.

Polimorfismo en *ToString*, retornará una cadena, respetando el siguiente formato:

DOCTOR – Pérez, legajo: 24, especialidad: Pediatra - ingreso: 09:12:18 hs., siendo Pérez el apellido del personal, 24 su número de legajo, Pediatra la especialidad y 09:12:18 la hora, minutos y segundos del horario de ingreso del médico (utilizar método *ToLongTimeString*, del atributo de tipo *DateTime*).

Especialidad

Enum

Cirujano

Pediatra

Clínico

MedicoEgresado (hereda de Médico):

Posee un único atributo protegido propio, que será inicializado por su único constructor, con el valor de la propiedad **Now**, de la clase *DateTime*.

El método privado *CalcularJornal*, retornará el valor del jornal del médico egresado, para ello se deberá obtener la diferencia (horario de egreso menos horario de ingreso) y multiplicarlo según la especialidad, de acuerdo a la siguiente tabla de valores:

Valor por segundo	Especialidad
\$90,00	Cirujano
\$70,00	Clínico
\$40,00	Pediatra

Nota: para obtener la diferencia entre dos objetos de tipo *DateTime*, se debe de invocar a la sobrecarga del operador de sustracción (-).
Ejemplo: *TimeSpan dif = fecha1 – fecha2*, siendo *fecha1* y *fecha2* dos objetos de tipo *DateTime*. Utilizar la propiedad *TotalSeconds* (*TimeSpan*) para obtener dicha diferencia expresada en segundos.

MedicoEgresado

Clase

Medico

Campos

horarioSalida : DateTime

Propiedades

Egreso { get; } : DateTime

Info { get; } : string

Jornal { get; } : double

Métodos

ArmarInfo() : string

CalcularJornal() : double

MedicoEgresado(Medico medico)

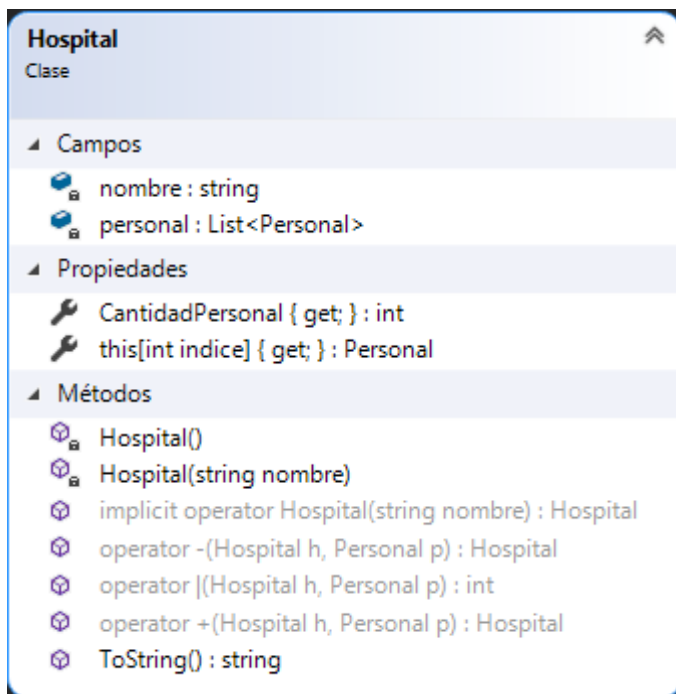
ToString() : string

La propiedad de sólo lectura **Jornal**, retornará el valor del método **CalcularJornal**.

Sobrescribir el método **ArmarInfo** para que retorne la siguiente cadena de texto: **DOCTOR – Pérez, legajo: 24, especialidad: Pediatra – JORNAL: \$900,00** , siendo Pérez el apellido, 24 su número de legajo y Pediatra la especialidad del médico. Reutilizar código.
Se deben mostrar sólo dos decimales en el valor del jornal.

Polimorfismo en **ToString**, retornará una cadena, respetando el siguiente formato:

DOCTOR – Pérez, legajo: 24, especialidad: Pediatra – JORNAL: \$900,00 - ingreso: 09:12:18 hs. - egreso: 09:12:28 hs, siendo Pérez el apellido del personal, 24 su número de legajo, Pediatra la especialidad, 09:12:18 la hora, minutos y segundos del horario de ingreso del médico y 09:12:28 la hora, minutos y segundos del horario de egreso del médico (utilizar método **ToLongTimeString**, del atributo de tipo **DateTime**). Reutilizar código.



La última clase que tendrá el proyecto será **Hospital**:

Dicha clase posee dos atributos, ambos privados. Uno indicará el nombre del hospital, el otro es una colección genérica de tipo **Personal**.

El constructor y su sobrecarga son privados. El constructor por defecto será el único que inicializará la lista genérica. La sobrecarga, inicializará el nombre del hospital. Reutilizar código.

La propiedad de sólo lectura, **CantidadPersonal**, retornará la cantidad de personal que hay en el hospital.

Sobrecarga de operadores:

Implícito, retornará una instancia de **Hospital** cuyo nombre coincida con el parámetro recibido. Reutilizar código.

Pipe (**|**), retornará el valor del índice del objeto de tipo **Personal**, si es que el personal se encuentra en el hospital, -1, caso contrario. Reutilizar código.

Adición (**+**), si el personal no se encuentra en el hospital, lo agregará a la colección. Reutilizar código.

Sustracción (**-**), si el personal se encuentra en el hospital, lo removerá según su índice. Reutilizar código.

Indexador:

Será de sólo lectura y retornará al objeto de tipo **Personal** que se ubique en el índice recibido. Retornar null si el índice es negativo o mayor o igual a la cantidad de elementos actual. Reutilizar código.

Polimorfismo en **ToString**, retornará una cadena con toda la información del hospital, incluyendo el detalle (completo) de cada personal. Reutilizar código.

Añadir a la misma solución, el proyecto de tipo ‘**Windows Forms**’, dónde se tendrán **cuatro (4)** formularios.
(**FrmPrincipal**, **FrmPersonal**, **FrmMedico** y **FrmMedicoEgresado**)

FrmPrincipal:

Este formulario tendrá un **Hospital** y una lista genérica de tipo **Personal** como atributos **protegidos**.

FrmPersonal:

Formulario base que contiene una propiedad de sólo lectura **PersonalDelForm** (abstracta) y dos métodos: **btnCancelar_Click** y **btnAceptar_Click** (virtual).

FrmMedico y FrmMedicoEgresado

Deriban de *FrmPersonal*, cada uno tendrán un atributo privado de tipo **Medico** y **MedicoEgresado** respectivamente. **FrmMedicoEgresado** deberá tener un constructor sobrecargado.

FrmPrincipal:

Apellido y nombre del alumno

Personal

Ingresado

Egresado

Ingresar

Egresar

Ordenar por:

Al pulsar el botón **Ingresar**, se mostrará una instancia de **FrmMedico** (modal), se completarán: Apellido, legajo y fecha. Al pulsar el botón Aceptar, se creará una instancia de tipo Medico, que se expondrá en la propiedad pública para ser agregado en el atributo de tipo Hospital del formulario principal.

Al ser agregados, se mostrarán en el ListBox correspondiente.

Personal

Ingresado

DOCTOR - Pratto, legajo: 68, Cirujano - ingreso: 16:31:40 hs.
DOCTOR - Quintero, legajo: 109, Clínico - ingreso: 16:33:42 hs.
DOCTOR - Martínez, legajo: 122, Pediatra - ingreso: 16:34:18 hs.

FrmMedico

Apellido:

Legajo:

Fecha de ingreso:

Especialidad:

Aceptar Cancelar

Al seleccionar un personal médico, se podrá egresarlo (si se pulsa el botón **Egresar**).

Personal

Ingresado

DOCTOR - Pratto, legajo: 68, Cirujano - ingreso: 16:31:40 hs.
DOCTOR - Quintero, legajo: 109, Clínico - ingreso: 16:33:42 hs.
DOCTOR - Martínez, legajo: 122, Pediatra - ingreso: 16:34:18 hs.

Se mostrará una instancia de **FrmMedicoEgresado** (modal), pasándole al constructor, el objeto de tipo Medico correspondiente al índice de la selección. Se mostrarán los datos del médico a ser egresado (incluyendo el cálculo del jornal). Si se acepta el egreso, se quitará del hospital y se agregará en la lista genérica de tipo Personal de **FrmPrincipal**.

Egresado

DOCTOR - Quintero, legajo: 109, Clínico - JORNAL: \$18140,92 - ingreso: 16:33:42 hs. - egreso: 16:38:01 hs.
DOCTOR - Pratto, legajo: 68, Cirujano - JORNAL: \$43785,67 - ingreso: 16:31:40 hs. - egreso: 16:39:46 hs.
DOCTOR - Martínez, legajo: 122, Pediatra - JORNAL: \$13255,81 - ingreso: 16:34:18 hs. - egreso: 16:39:50 hs.

Por último, se podrá ordenar el listado de egresados según la selección del ComboBox.

Ordenar por:

Legajo ascendente
Legajo ascendente
Legajo descendente

Ordenar por:

Legajo descendente
Legajo ascendente
Legajo descendente

FrmMedicoE...

Apellido:

Legajo:

Fecha de ingreso:

Especialidad:

Jomal:

Aceptar Cancelar