

**Universidad Tecnológica Nacional
Facultad Regional Avellaneda**



Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos

Materia: Laboratorio de Programación II

Apellido:		Fecha:	
Nombre:		Docente ⁽²⁾ :	
División:		Nota ⁽²⁾ :	
Legajo:		Firma ⁽²⁾ :	
Instancia ⁽¹⁾ :	PP	RPP	SP
			RSP
			FIN

(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.

Colocar sus datos personales en el nombre del proyecto principal, colocando: Apellido.Nombre.AñoCursada.
Ej: Pérez.Juan.2016. **No sé corregirán proyectos que sea identificable su autor.**

TODAS las clases e interfaces deberán ir en una Biblioteca de Clases llamada *Entidades*.
No se corregirán exámenes que no compilen.

- (1pt) En la clase paciente se debe tener un atributo STATIC llamado "ultimoTurnoDado", que sea inicializado en cero (0) en el constructor estático de la clase Paciente.
- (1pt) Constructores: respetar el diagrama para la creación de los constructores
 - El constructor **Paciente** (string, string, int) asignará los valores a cada atributo, modificando también ultimoTurnoDado por el valor recibido.
 - El constructor **Paciente** (string, string) incrementará el valor de ultimoTurnoDado en 1 y se lo asignará al turno.
 - El constructor de clase en **Medico** instanciará a tiempoAleatorio. El atributo tendrá visibilidad de protegido.
- (1 punto) En paciente: ToString() retornará los datos del paciente con el siguiente formato "Turno N°{0}: {2}, {1}", siendo los valores número de turno, apellido y nombre respectivamente.
- (1 punto) En Medico: La propiedad "EstaAtendiendoA" será de sólo lectura y virtual, retornando los datos del "pacienteActual".
- (1 punto) En Medico: La propiedad "AtenderA" será de sólo escritura, asignando el valor al atributo "pacienteActual".
- (1 punto) En Medico: "Atender" será protegido y abstracto.
- (1 punto) En Medico: el método "FinalizarAtencion" lanzará el evento "AtencionFinalizada" y luego asignará null al paciente actual.
- (1 punto) **MGeneral** y **MEspecialista**: El método "IniciarAtencion" será el encargado de crear y lanzar un hilo dónde se ejecutará el método "Atender".
- (1 punto) **MGeneral** y **MEspecialista**: El método Atender hará un Sleep de un tiempo aleatorio (de entre 5000 y 10000 para MEspecialista y de entre 1500 y 2200 para MGeneral). Luego avisará que finalizó la atención.
- (1 punto) **Test Unitario**:
Realizar un Test Unitario de nombre ConstructoresPaciente que pruebe los constructores de paciente. Al ejecutar los siguientes constructores en el orden propuesto, deberán cumplirse los siguientes requerimientos:
Si hacemos new Paciente("Nombre", "Apellido"); el valor del Turno deberá ser 1.
Si hacemos new Paciente("Nombre", "Apellido", 5); el valor del Turno deberá ser 5.

Si hacemos new Paciente("Nombre", "Apellido"); el valor del Turno deberá ser 6.

IMedico
Interface

Methods

● IniciarAtencion(Paciente p): void

Persona

Abstract Class

Fields

● apellido: string

● nombre: string

Methods

● Persona(string nombre, string apellido)

Paciente

Class

→ Persona

Fields

● turno: int

● ultimoTurnoDado: int

Properties

● Turno { get; } : int

Methods

● Paciente()

● Paciente(string nombre, string apellido)

● Paciente(string nombre, string apellido, int turno)

● ToString(): string

Medico

Abstract Class

→ Persona

Fields

● pacienteActual: Paciente

● tiempoAleatorio: Random

Properties

● AtenderA { set; } : Paciente

● EstaAtendiendoA { get; } : string

Methods

● Atender(): void

● FinalizarAtencion(): void

● Medico()

● Medico(string nombre, string apellido)

Events

● AtencionFinalizada: FinAtencionPaciente

Nested Types

IMedico

MEspecialista

Class

→ Medico

Fields

● especialidad: Especialidad

Methods

● Atender(): void

● IniciarAtencion(Paciente p): void

● MEspecialista(string nombre, string apellido, Especialidad e)

Nested Types

Especialidad

Enum

Traumatologo

Odontologo

IMedico

MGeneral

Class

→ Medico

Methods

● Atender(): void

● IniciarAtencion(Paciente p): void

● MGeneral(string nombre, string apellido)

Diseñar el siguiente formulario:

The screenshot shows a Windows application window with the title bar 'Final 27-07-2017'. Inside the window, there are two buttons stacked vertically. The top button is labeled 'Atender Médico General' and the bottom button is labeled 'Atender Médico Especialista'.

```

frmFinal
Class
+ Form

Fields
medicoEspecialista : MEspecialista
medicoGeneral : MGeneral
mock : Thread
pacientesEnEspera : Queue<Paciente>

Methods
AtenderPacientes(IMedico iMedico) : void
btnEspecialista_Click(object sender, EventArgs e) : void
btnGeneral_Click(object sender, EventArgs e) : void
FinAtencion(Paciente p, Medico m) : void
frmFinal()
frmFinal_FormClosing(object sender, FormClosingEventArgs e) : void
frmFinal_Load(object sender, EventArgs e) : void
MockPacientes() : void
  
```

11) (1 punto) Crear los botones y las modificaciones necesarias para poder serializar y desSerializar una lista de pacientes auxiliar, tomando los datos de pacientesEnEspera

12) (3 puntos) Modelar un sistema de atención para un Sanatorio.

Siendo:

- A. frmFinal() el constructor del formulario, dónde se instanciarán los atributos del formulario, siendo medicoEspecialista y medicoGeneral:
`this.medicoGeneral = new MGeneral("Luis", "Salinas");`
`this.medicoEspecialista = new MEspecialista("Jorge", "Iglesias", MEspecialista.Especialidad.Traumatologo);`
- B. frmFinal_Load() el evento de carga del formulario, dónde inicializaremos el hilo mock.
- C. frmFinal_FormClosing() el evento de cierre del formulario, dónde, si el hilo mock aun está activo, se abortará.
- D. MockPacientes() dónde se agreguen pacientes a la cola pacientesEnEspera, haciendo un Sleep de 5000 (`Thread.Sleep(5000)`).
- E. AtenderPacientes(IMedico) será invocado por los eventos click de los botones (btnEspecialista_Click y btnGeneral_Click) pasandole el médico que corresponda (medicoEspecialista o medicoGeneral, respectivamente). En el caso de haber pacientes en espera, se deberá iniciar la atención del primer elemento de la cola.

- F. FinAtencion(Paciente, Medico) mostrará por medio de un MessageBox un mensaje con el formato "Finalizó la atención de {0} por {1}!", dónde se indicará el nombre del paciente y el del médico que lo atendió, respectivamente.

