## Universidad Tecnológica Nacional Facultad Regional Avellaneda



				3,4	UINFra
Técnico Sup	erior en Pro	gramación -	Técnico Supe	erior en Sisten	nas Informáticos
Materia: La	boratorio	de Progra	amación II		
Apellido:			Fecha:		
Nombre:			Docente <sup>(2)</sup> :		
División:			Nota(2):	Nota <sup>(2)</sup> :	
Legajo:			Firma <sup>(2)</sup> :		
Instancia <sup>(1)</sup> :	PP	RPP	SP	RSP	FIN

(1) Las instancias validas son: 1et Parcial (PP), Recuperatorio 1et Parcial (RPP), 2do Parcial (SP), Recuperatorio 2do 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 <u>estático</u> de la clase Paciente.
- 2. (1pt)Constructores: respetar el diagrama para la creación de los constructores
  - a. El constructor **Paciente** (string, string, int) asignará los valores a cada atributo, modificando también ultimoTurnoDado por el valor recibido.
  - b. El constructor **Paciente** (string, string) incrementará el valor de ultimoTurnoDado en 1 y se lo asignará al turno.
  - c. El constructor de clase en **Medico** instanciará a tiempoAleatorio. El atributo tendrá visibilidad de protegido.
- 3. (1 punto)En paciente: ToString() retornará los datos del paciente con el siguiente formato "Turno Nº{0}: {2}, {1}", siendo los valores número de turno, apellido y nombre respectivamente.
- 4. (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".
- 6. (1 punto)En Medico: "Atender" será protegido y abstracto.
- 7. (1 punto)En Medico: el método "FinalizarAtencion" lanzará el evento "AtencionFinalizada" y luego asignará null al paciente actual.
- 8. (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.
- 10. (1punto) 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. Medico Persona Interface Abstract Class - Methods - Fields apellido string
a, nombre string () Iniciar Atencian(Paciente p) : void Paciente Methods Class → Persona Persona(string numbre, string apellido) - Fields tume int 🐾 ultimoTumoDado : int Medica - Properties Abstract Class + Persona ➢ Turno ( get; ): int - Methods 📭 Paciente() 
 ✓ parienteActual Paciente

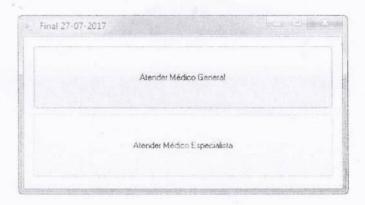
 ✓ tiempoAleatorio Random (a) Paciente(string nombre, string apellido) Paciente(string nombre, string apelligo, int turno) □ ToString() : string Æ AtenderA ( set; ) Paciente F EstaAtendiendoA [ get; ]: string IMedico - Methods MEspecialista , Atenderg : void FinalizarAtencion(): void

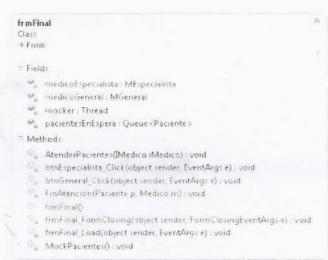
Medico()

Medico(string nombre; string apellido) + Medico especialidad : Especialidad = Events - Methods AtencionFinalizada : FinAtencionPaciente 4. Atender(): void \* Nested Types IniciarAtencion(Paciente p) : void MEspecialista(string nombre, string apellido, Especialidad e) - Nested Types Medico Especialidad MGeneral Class → Medico Traumatologo Odontologo Methods Atender() word ImmarAtencion(Paciente p) - void

6 MGeneral(string nombre, string apellido)

Diseñar el siguiente formulario:





- 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 mocker.
- C. frmFinal\_FormClosing() el evento de cierre del formulario, dónde, si el hilo mocker aun está activo, se abortará.
- D. MockPacientes() dónde se agreguen pacientes a la cola pacientesEnEspera, haciendo un Sleep de 5000 (Obread.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.