

**Universidad Tecnológica Nacional  
Facultad Regional Avellaneda**



Técnico Superior en Programación - Técnico Superior en Sistemas Informáticos

**Materia: Laboratorio de Programación II**

Apellido:		Fecha:	
Nombre:		Docente <sup>(2)</sup> :	
División:		Nota <sup>(2)</sup> :	
Legajo:		Firma <sup>(2)</sup> :	
Instancia <sup>(1)</sup> :	<div style="display: flex; justify-content: space-around;"> <span>PP</span> <span></span> <span>RPP</span> <span></span> <span>SP</span> <span></span> <span>RSP</span> <span></span> <span>FIN</span> <span></span> </div>		

(1) Las instancias validas son: 1<sup>er</sup> Parcial (**PP**), Recuperatorio 1<sup>er</sup> Parcial (**RPP**), 2<sup>do</sup> Parcial (**SP**), Recuperatorio 2<sup>do</sup> Parcial (**RSP**), Final (**FIN**). Marque con una cruz.

(2) Campos a ser completados por el docente.

## RECUPERATORIOS

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.**

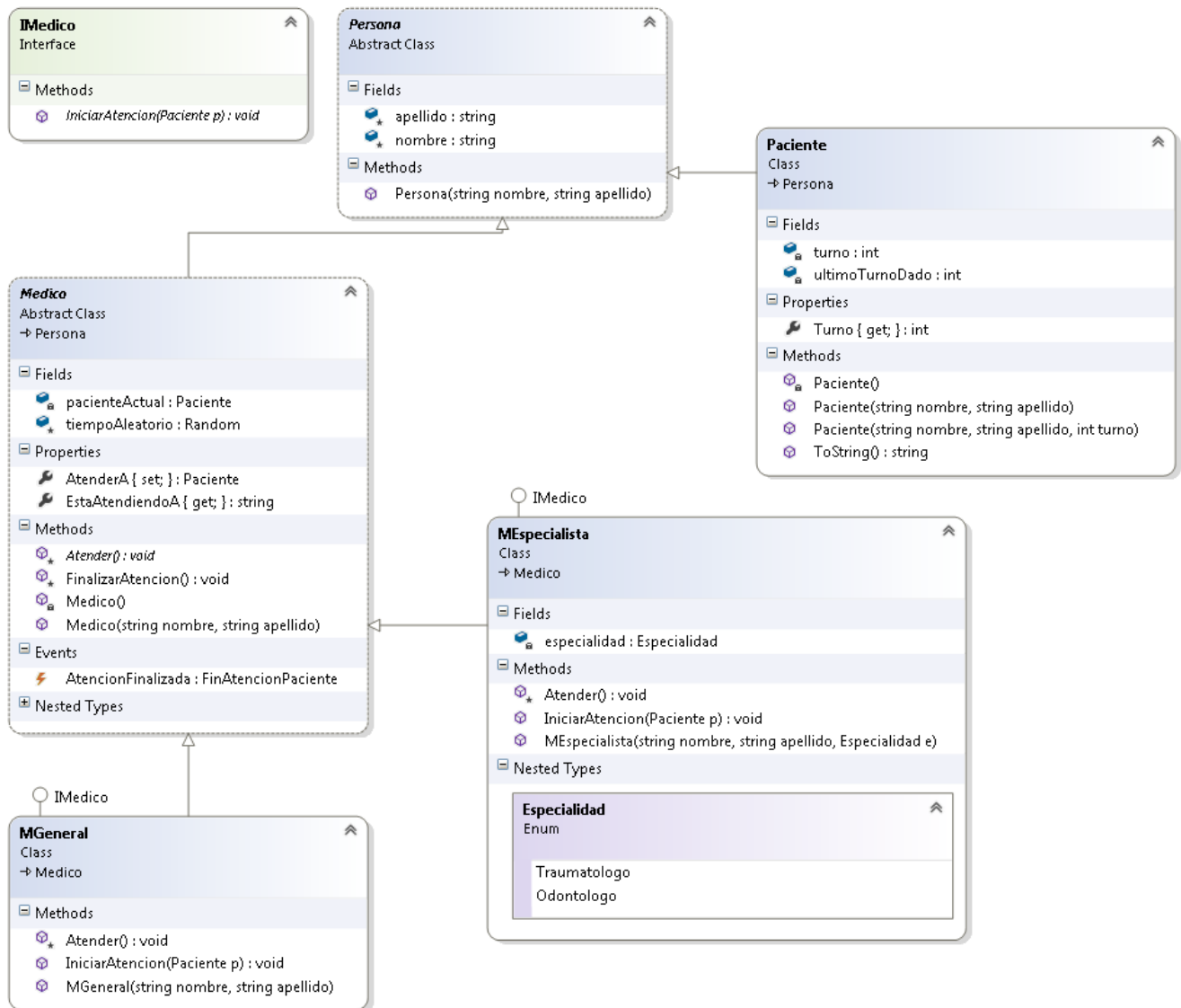
TODAS las clases e interfaces deberán ir en una Biblioteca de Clases llamada *Entidades*.

**No se corregirán exámenes que no compilen.**

1. (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.
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".
5. (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":
  - a. 1er parcial: retornará el paciente actual y asignará al atributo pacienteActual el valor null.
  - b. 2do parcial: lanzará el evento "AtencionFinalizada" y luego asignará null al paciente actual.
  - c. Final 2015: ídem primer parcial.
8. (1 punto) **MGeneral y MEspecialista**: El método "IniciarAtencion" será el encargado de:
  - a. 1er parcial: ejecutará el método Atender. Modificar IniciarAtencion para que retorne un Paciente.
  - b. 2do parcial: crear y lanzar un hilo dónde se ejecutará el método "Atender".
  - c. Final 2015: ídem primer parcial.
9. (1 punto) **MGeneral y MEspecialista**: El método Atender:
  - a. 1er parcial: avisará que finalizó la atención.

- b. 2do parcial: 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.
  - c. Final 2015: ídem primer parcial.
10. (1punto) Al finalizar cada atención, se agregará al archivo pacientes\_atendidos.txt los datos de la atención.

Diagrama ilustrativo:



---

*Primer Parcial – Final Alumnos 2015:*

---

11. (2 puntos) Agregar la clase Sanatorio:

a. Atributos privados.

i. medicoEspecialista : MedicoEspecialista

ii. medicoGeneral : MedicoGeneral

iii. pacientesEnEspera: Queue<Paciente>

iv. turnosTotales: int

b. Un constructor privado dónde se instanciarán los atributos, siendo medicoEspecialista y medicoGeneral:

`this.medicoGeneral = new MGeneral("Luis", "Salinas");`

`this.medicoEspecialista = new MEspecialista("Jorge", "Iglesias",  
MEspecialista.Especialidad.Traumatologo);`

c. Un constructor público que recibirá un número entero.

d. Crear un método TomarDatos(Persona), que sea privado y estático. En el se serializará como XML la persona recibida como parámetro. El nombre del archivos será "[nombre].[apellido].xml", utilizando nombre y apellido del Paciente.

e. Sobrecargar el operador + para que agregue un nuevo paciente a la cola, mientras turnosTotales sea mayor a 0 (cero). De no contar con más turnos, se invocará a TomarDatos.

12. (1 punto) Generar un proyecto Test del tipo Consola.

a. Generar un sanatorio con 3 turnos totales.

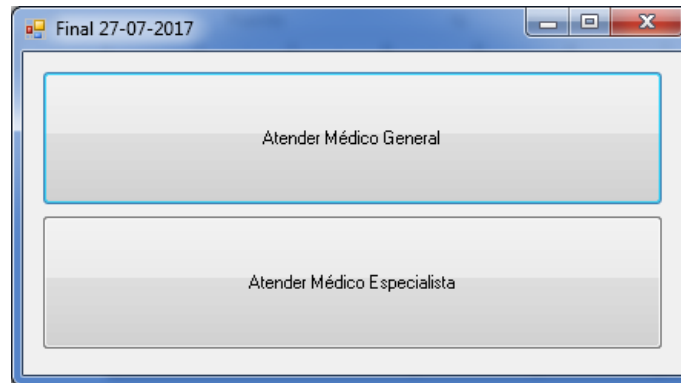
b. Agregar 4 pacientes al sanatorio, avisando por pantalla si no pudo agregarlo.

---

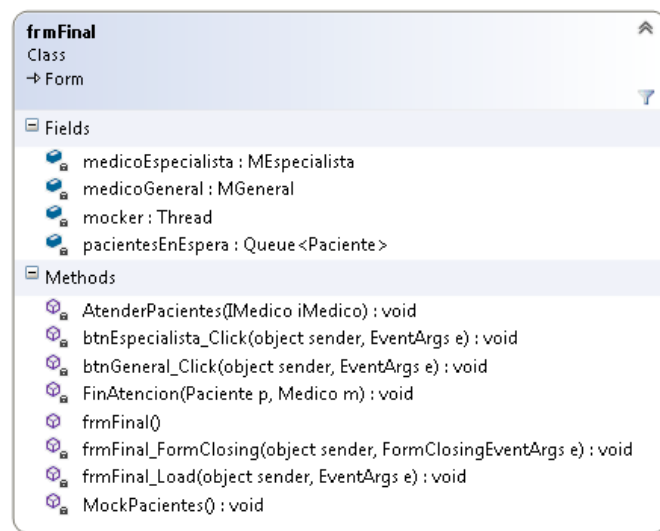
## Segundo Parcial:

---

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 rectangular buttons stacked vertically. The top button is labeled 'Atender Médico General' and the bottom button is labeled 'Atender Médico Especialista'.



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 (`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.