Icon

Description automatically generated with medium confidence**EXAMEN FINAL DE PARADIGMAS DE PROGRAMACION – MAYO 2004**

**Trabajo Práctico**

**Paradigmas de la Programación**

**Grupo 09**

Bur Stettler, Damián Alejando - 48882

Variego, Manuel - 50925

Marchesa, Joel Andrés – 49495

**Ingeniería en Sistemas de Información**

Comisión 203

Año 2023

**Facultad Regional Rosario**

La administración de un sanatorio decide ampliar su sistema informático, implementando un módulo para gestionar la programación de las intervenciones quirúrgicas a pacientes.

El sanatorio guarda información sobre sus pacientes, sus médicos y sobre las intervenciones quirúrgicas habilitadas por nomenclador (estos datos ya están registrados en el sistema).

De cada médico registra: nombre y apellido, matrícula profesional, especialidad y condición de disponible o no para intervenir.

De cada paciente registra: documento de identidad, nombre y apellido, nombre de la obra social a la que pertenece (si no tiene obra social, guarda el valor “No tiene”), porcentaje de cobertura sobre arancel (que es igual a cero si no tiene obra social) y conjunto de las intervenciones quirúrgicas realizadas.

Las intervenciones quirúrgicas habilitadas por nomenclador pueden ser **comunes** o de **alta complejidad.** De cada intervención se registra: código, descripción, especialidad a la que pertenece y arancel (precio) según nomenclador.

Si la intervención es de alta complejidad, se registra además el porcentaje adicional que se cobra por uso de equipo especial y que es el mismo para todas las intervenciones de alta complejidad.

Cuando se solicita una intervención para un paciente, debe registrarse la siguiente información: fecha de la intervención, intervención a efectuar, médico que interviene y condición de pagado o pendiente de pago.

Se solicita:

1. utilizando el paradigma de objetos, desarrollar el diagrama de clases completo indicando: clases, subclases, variables de instancia, variables de clase, métodos de instancia, métodos de clase, relaciones de ensamble y de herencia que considere necesarios para resolver el problema.
2. codificar en lenguaje Smalltalk el método **menú** y **todos los métodos necesarios** para:

* **registrar una intervención para un paciente** de acuerdo a las siguientes pautas:

1. el código de intervención a realizar se ingresa por teclado y debe verificarse que la intervención esté habilitada por nomenclador
2. el médico que interviene debe corresponder a la especialidad de la intervención y estar en condición de disponible para intervenir
3. el sanatorio asegura que siempre se encontrará al menos un médico para cada especialidad en condición de disponible para intervenir
4. el sanatorio solo interviene a pacientes registrados. El documento de identidad del paciente a intervenir se ingresa por teclado.

* **obtener la liquidación de las intervenciones pendientes de pago de un paciente**

El listado de liquidación deberá tener el siguiente formato:

Liquidación del paciente: Juan Pérez. Obra social: “MED”

Fecha Descripción Médico Mat. Importe

5/12/03 Cataratas Dra. Susana Greco 2314 $1000

10/12/03 Corazón2 Dr. Jorge Almada 14.234 $5000

Total $6000

Cobertura Obra Social $600

Neto a pagar $5400

El importe de cobertura de la obra social surge de aplicar el porcentaje de cobertura del paciente al importe total de las intervenciones adeudadas.

* **Agregado: Realizar consultas por DNI, matrícula y código de especialidad.**
* **Menú administrador:** menú al que se accede con /admin cuando se ingresa la primera opción del menú. Al entrar al mismo, se dispone de otro menú que permite realizar el alta de pacientes, médicos e intervenciones. La lógica detrás de esto es aplicar el principio del mínimo privilegio, y que solo los encargados/responsables del área alta puedan agregar datos a las colecciones. Los recepcionistas solo podrán registrar Intervenciones de pacientes, obtener liquidaciones y consultar datos.
* **Además, se agregan múltiples validaciones y gestión de errores.** Al registrar una Intervención para el paciente se le mostrará una **lista detallada de los médicos e intervenciones** de acuerdo con la especialidad requerida, para beneficio del cliente.

Se toman las medidas necesarias para **evitar tener que recurrir al Transcript**, enviándole el contenido a MessageBox para beneficio del usuario.

1. Considerar el siguiente programa principal:

**│w│**

**w := Sanatorio new.**

**w inicializar; menú**

El método **inicializar** prepara los datos de pacientes, médicos e intervenciones habilitadas para ser utilizados por los restantes métodos y **no** debe codificarse.

**NOTA**: Se evaluará la óptima utilización de los recursos del paradigma y del lenguaje.

A close-up of a diagram

Description automatically generated

|  |
| --- |
| **#Sanatorio** |

**mi: buscarEnColeccion: unValor y: unaColeccion**

"Busca en la colección seleccionada el objeto cuyo id coincida para posteriormente poder hacer referencia al mismo en la liquidación"

|temp|

(unaColeccion = intervencion) ifTrue:[

temp:= unaColeccion detect:[:i | i codigo = unValor]

].

(unaColeccion = medico) ifTrue: [

temp:= unaColeccion detect:[:i | i matricula = unValor]

].

(unaColeccion = paciente) ifTrue: [

temp:= unaColeccion detect:[:i | i dni = unValor]

].

^temp

**mi: calcDescuento: unTotal y: unPorcentaje**

"Calcula el descuento usado para la liquidación teniendo en cuenta los parámetros utilzados"

^(unTotal\*unPorcentaje)/100

**mi: consulta**

"Menú con las opciones de consulta."

|op|

op:='5'.

[op='0'] whileFalse: [(MessageBox notify: '1 - Buscar pacientes

2 - Buscar médicos

3. Buscar Intervenciones

0 - Volver al menú' caption: 'Menú > Consultas').

op:=(Prompter prompt: 'Ingrese una opción').

(op='1') ifTrue: [self consultaPaciente].

(op='2') ifTrue: [self consultaMedico].

(op='3') ifTrue: [self consultaIntervencion ].

((((op='1' or: [op='2']) or: [op='0']) or:[op='0']) or: [op='3']) ifFalse: [op:=(Prompter prompt: 'Opcion invalida. Ingrese otra.')].

]

**mi: consultaIntervencion**

"Se ingresa el código y se busca en la colección intervención, si exise, muestra sus datos."

|cod t|

[t isNil] whileTrue:[

cod:=Prompter prompt: 'Ingrese el código de la intervención' caption:'Consulta > Intervención'.

t:= intervencion detect:[:i | i codigo = cod]

ifNone:[ MessageBox notify: 'Incorrecto. Vuelva a ingresar el código o escriba SALIR para regresar al menú.'. t:= nil. ((cod='SALIR') ifTrue: [t:='3']) ]].

(t isNil and: [cod='SALIR']) ifFalse: [t muestra].

**mi:** **consultaMedico**

"Se ingresa una matrícula y se busca si existe un objeto con esa matrícula en la colección medico. Si existe, se muestran sus datos."

|mat m|

[m isNil] whileTrue:[

mat:= Prompter prompt: 'Ingrese la matrícula del profesional' caption:'Consulta > Médico'.

m:= medico detect:[:i | i matricula=mat]

ifNone:[ MessageBox notify: 'Incorrecto. Vuelva a ingresar legajo o escriba SALIR para regresar al menú.'. m:= nil. ((mat='SALIR') ifTrue: [m:='3'])]].

(m isNil and: [mat='SALIR'] ) ifFalse: [m muestra].

**mi: consultaPaciente**

"Se ingresa un DNI y se lo busca en la colección paciente. Si existe, se muestran sus datos."

|pac p|

[p isNil] whileTrue:[

pac:=Prompter prompt: 'Ingrese el DNI del paciente' caption:'Consulta > Paciente'.

p:= paciente detect:[:i | i dni=pac ]

ifNone:[ MessageBox notify: 'Incorrecto. Vuelva a ingresar el DNI o escriba SALIR para regresar al menú.'. p:= nil. ((pac='SALIR') ifTrue: [p:='3'])]].

(p isNil and: [pac='SALIR'] ) ifFalse: [p muestra].

**mi: esFechaValida: unaFecha**

"Valida que el argumento unaFecha, que es un String, pueda ser convertido a una fecha correctamente. Valida además que la fecha no sea del pasado."

| fechaHoy fecha temp |

fecha := [Date fromString: unaFecha format: 'MM/DD/yyyy'] on: Error do: [:each | temp:= false].

(temp=false) ifTrue: [temp:=false] ifFalse:[temp:=true].

fechaHoy := Date today.

^ (temp=true and: [fecha >= fechaHoy]).

**mi: estadoliquidacion: unDNI**

"Recibe como parámetro unDNI y primero valida que exista el paciente. Si existe busca en la colección IntervenciónPaciente los datos pertenecientes a ese paciente con sus intervenciones que no estén pagadas. Si el paciente no registra deudas, se emite el mensaje correspondiente."

|coleccionPaciente coleccion2 tempInt tempMed acumAdic total|

total:= 0.

acumAdic:=0.

coleccionPaciente:= (self buscarEnColeccion: unDNI y: paciente).

coleccion2 := intervencionPaciente select:[:each | each paciente = unDNI and:[each condicionPago = false]].

Transcript clear.

Transcript show: 'Paciente: ';show: coleccionPaciente nombre;show: ' ';show: coleccionPaciente apellido;show: ' Obra social: '; show: coleccionPaciente obraSocial; cr.

Transcript show:'';show: 'Fecha';tab; show: ' Descripcion ';show: ' Medico ';tab; show: ' Mat. '; show: 'Importe'; cr.

coleccion2 do: [:i |

tempInt:= (self buscarEnColeccion: (i intervencion) y: intervencion).

tempMed:= (self buscarEnColeccion: (i medico) y: medico).

Transcript show: i fecha;show: ' '; show: tempInt descripcion;show: ' ' ;show: tempMed nombre;show:' ';show: tempMed apellido;show: ' '; show: tempMed matricula;show: ' ';show:' ';show: '$'; print: tempInt arancel; tab; tab;

cr.

(tempInt isKindOf: AltaComplejidad) ifTrue: [total:= total + ((tempInt arancel) \* (1+ (AltaComplejidad adicional / 100))). acumAdic:= (((tempInt arancel) \* (1+ (AltaComplejidad adicional / 100)))-tempInt arancel) + acumAdic ] ifFalse: [total:= total + (tempInt arancel)]

].

Transcript cr; show: 'Carga por Adicional';tab;tab;tab;show:'$';print: acumAdic; cr.

Transcript show: 'Total';tab;tab;tab;tab;show:'$';print: total; cr.

Transcript show: 'Cobertura Obra social';tab;tab;show: '$';print: (self calcDescuento: total y: coleccionPaciente porcCobertura) ;cr.

Transcript show: 'Neto a pagar ';tab;tab;tab;show:'$';print: (self netoaPagar: total y: coleccionPaciente porcCobertura) ;cr.

(total = 0) ifTrue: [^'Este paciente no registra deudas'] ifFalse:[^(Transcript contents) asString].

**mi: existeCOD: unCOD**

"Valida que exista el código de intervención"

|i int|

int:= unCOD.

i:= intervencion detect:[:each | each codigo=int] ifNone:[i:= 'no'.].

(i='no') ifTrue: [^false] ifFalse: [^true ]

**mi: existeDNI: unDNI**

"Valida que exista el código de DNI"

| p pac|

pac:= unDNI.

p:= paciente detect:[:i | i dni=pac] ifNone:[p:= 'no'.].

(p='no') ifTrue: [^false] ifFalse: [^true ]

**mi: existeEspecialidad: unaEspecialidad**

"Valida que exista una especialidad en la colección intervención. Muy útil para validaciones posteriores."

|e|

e:= intervencion detect:[:i | i especialidad=unaEspecialidad] ifNone:[e:= 'no'.].

(e='no') ifTrue: [^false] ifFalse: [^true ]

**mi: existeMatricula: unaMatricula**

"Valida que exista la matrícula enviada en el argumento en la colección medico"

| m med|

med:= unaMatricula.

m:= medico detect:[:i | i matricula=med] ifNone:[m:= 'no'.].

(m='no') ifTrue: [^false] ifFalse: [^true ]

Mi: inicio

"Se inician las colecciones y se ejecuta el menú"

paciente := OrderedCollection new.

medico := OrderedCollection new.

intervencion := OrderedCollection new.

intervencionPaciente:= OrderedCollection new.

self menu.

**mi: intervencionesDisponibles: unaEspecialidad**

"Lista de intervenciones disponibles teniendo en cuenta la especialidad seleccionada."

|coleccion temp|

coleccion := intervencion select: [:each | each especialidad=unaEspecialidad].

(coleccion isEmpty)

ifTrue:[MessageBox notify: 'No hay intervenciones disponibles.' ]

ifFalse: [

Transcript clear.

Transcript show: 'ESPECIALIDAD - '; show: unaEspecialidad asUppercase; cr.

Transcript show: 'CÓDIGO'; tab;tab; show:'DESCRIPCIÓN';tab;tab;show:'ARANCEL'; cr.

coleccion do: [:each |

Transcript

show: each codigo; tab;tab;

show: each descripcion; tab;tab;

show: each arancel printString; tab;tab;

cr.

].

temp:= (Transcript contents) asString.

^temp

]

**mi: liquidacion**

"Permite que se ingrese el DNI del usuario del que se quiera obtener la liquidación, valida que exista y muestra la liquidación invocando al método estadoliquidacion"

|coleccion2 rta dni|

rta:=1.

dni:=(Prompter prompt: 'Ingrese el DNI del paciente con intervenciones registradas').

[rta = 1] whileTrue:[

coleccion2 := intervencionPaciente select: [:each | each paciente = dni].

(coleccion2 isEmpty) ifTrue: [

(MessageBox warning: 'No existe una intervencion pendiente de pago registrado con ese DNI. Intente nuevamente').

] ifFalse: [

MessageBox notify: (self estadoliquidacion: dni) caption: 'LIQUIDACIÓN'.

].

dni:=(Prompter prompt: 'Ingrese otro DNI o 0 para salir').

(dni = '0') ifTrue:[ rta:=0].

].

**mi: medicosDisponibles: unaEspecialidad y: unaOpcion**

"Muestra una lista de médicos disponibles teniendo en cuenta la especialidad. Muy importante para validaciones y para el registro de intervenciones de pacientes. De acuerdo a la opción, mostrará una lista o un booleano indicando si existen médicos para esa especialidad o no."

|coleccion1 coleccion2 temp temp2|

coleccion1 := medico select: [:each | each condicion].

coleccion2:= coleccion1 select: [:each | each especialidad=unaEspecialidad].

(coleccion2 isEmpty)

ifTrue:[(unaOpcion =1) ifTrue: [MessageBox notify: 'No hay médicos disponibles.'.]. temp2:=false.]

ifFalse: [

Transcript clear.

Transcript show: 'ESPECIALIDAD - '; show: unaEspecialidad asUppercase; cr.

Transcript show: 'MATRÍCULA'; tab; show:'PROFESIONAL'; cr.

coleccion2 do: [:each |

Transcript

show: each matricula; tab; tab;

show: each nombre; show: ' '; show: each apellido; tab; tab;

cr.

].

temp2=true.

temp:= (Transcript contents) asString.

].

(unaOpcion=1)ifTrue:[^temp] ifFalse:[^temp2]

**mi: menu**

"Menú desde el cual se pueden registrar intervenciones para pacientes y acceder a la liquidación. Para realizar altas se deberá ingresar al menú admin, introduciendo /admin como opción"

|op|

op:='5'.

[op='0'] whileFalse: [MessageBox notify: '1 - Registrar Intervencion

2 - Mostrar liquidacion

3. Consultas

0 - Salir' caption: 'MENU DE OPCIONES'.

op:=(Prompter prompt: 'Ingrese una opcion').

(op='1') ifTrue: [self registrarIntervencionPaciente].

(op='2') ifTrue: [self liquidacion].

(op='3') ifTrue: [self consulta].

(op= '/admin') ifTrue: [self menuAdmin].

((((op='1' or: [op='2']) or: [op='/admin']) or:[op='3']) or: [op='0']) ifFalse: [op:=(Prompter prompt: 'Opcion invalida. Ingrese otra.')]

]

**mi: menuAdmin**

"Menú que permite el alta de todas las colecciones críticas"

|op|

op:='4'.

[op='0'] whileFalse: [MessageBox notify: '1 - Registrar paciente

2 - Registrar médico

3 - Registrar intervención

0 - Volver al menú' caption: 'PANEL DE ADMINISTRADOR'.

op:=(Prompter prompt: 'Ingrese una opción:').

(op='1') ifTrue: [self registrarPaciente].

(op= '2') ifTrue: [self registrarMedico].

(op='3') ifTrue: [self registrarIntervencion].

(((op='1' or: [op='2']) or: [op='0']) or: [op='3']) ifFalse: [op:=(Prompter prompt: 'Opcion invalida. Ingrese otra.')]

]

**mi: netoaPagar: unTotal y: unPorcentaje**

"Método que calcula el monto neto a pagar. Se utiliza en estadoLiquidacion:"

^(unTotal - (self calcDescuento: unTotal y: unPorcentaje))

**mi: registrarIntervencion**

"Se registra una intervención validando además que haya médicos disponibles para la especialidad correspondiente."

|rta rta2 t cod especialidad|

rta:= true.

(intervencion isEmpty) ifTrue: [AltaComplejidad cargaAdicional. MessageBox notify: 'Adicional agregado con éxito' .].

[rta] whileTrue: [

cod := (Prompter prompt: 'Ingrese el codigo' caption:'Menú administrador > Registro > Intervención').

(self existeCOD: cod) ifTrue: [

MessageBox warning: 'El codigo ya existe. Por favor, ingrese otro.' caption:'Menú administrador > Registro > Intervención'.

] ifFalse: [

especialidad:=(Prompter prompt: 'Ingrese la especialidad').

((self medicosDisponibles: especialidad y: 2)=false) ifTrue: [MessageBox warning: 'No hay médicos disponibles para esa especialidad.' caption:'Menú administrador > Registro > Intervención'.]

ifFalse:[

rta2 := MessageBox confirm: '¿Es una intervencion de alta complejidad?' caption:'Menú administrador > Registro > Intervención'.

t := rta2

ifTrue: [AltaComplejidad new]

ifFalse: [Intervencion new].

t cargaDatos: cod y: especialidad.

intervencion add: t.

].

rta:= MessageBox confirm: '¿Desea ingresar otra intervencion?' caption:'Menú administrador > Registro > Intervención'

]].

**mi: registrarIntervencionPaciente**

"Se registra una intervención de paciente, validando todos los datos correspondientes."

|rta p fecha inter matricula pac espe|

rta:= true.

(paciente isEmpty or: [intervencion isEmpty or: [medico isEmpty]]) ifTrue: [MessageBox errorMsg: 'BASE DE DATOS VACIA.' caption: 'Error del sistema'] ifFalse: [

[rta] whileTrue: [

fecha := (Prompter prompt: 'Ingrese una fecha. (MM/DD/YYYY)' caption:'Menú administrador > Registro > Intervención de paciente').

(self esFechaValida: fecha) ifFalse: [

MessageBox warning: 'Fecha inválida. Vuelva a intentarlo.' caption:'Menú administrador > Registro > Intervención de paciente'.

] ifTrue: [

pac := (Prompter prompt: 'Ingrese el DNI del paciente' caption:'Menú administrador > Registro > Intervención de paciente').

(self existeDNI: pac) ifFalse: [

pac:= MessageBox errorMsg: 'El documento ingresado no coincide con nuestros registros. Vuelva a intentarlo.' caption:'Menú administrador > Registro > Intervención de paciente'.

].

espe:= Prompter prompt: 'Ingrese la especialidad.' caption:'Menú administrador > Registro > Intervención de paciente'.

[self existeEspecialidad: espe ] whileFalse: [

espe:= Prompter prompt: 'Los datos ingresados no coinciden con nuestros registros. Vuelva a intentarlo.' caption:'Menú administrador > Registro > Intervención de paciente'.

].

MessageBox notify:(self medicosDisponibles: espe y: 1).

matricula:= Prompter prompt: 'Ingrese la matrícula del profesional' caption:'Menú administrador > Registro > Intervención de paciente'.

[self validarMedico: matricula y: espe] whileFalse: [

MessageBox warning:('La matrícula ingresada no coincide con nuestros registros. Vuelva a intentarlo.') caption:'Menú administrador > Registro > Intervención de paciente'.

MessageBox notify:(self medicosDisponibles: espe y: 1).

matricula:= Prompter prompt: 'Ingrese la matrícula del profesional.' caption:'Menú administrador > Registro > Intervención de paciente'.

].

MessageBox notify:(self intervencionesDisponibles: espe).

inter:= Prompter prompt: 'Ingrese el código de intervención' caption:'Menú administrador > Registro > Intervención de paciente'.

[self validarIntervencion: inter y: espe] whileFalse: [

MessageBox warning:('El código de intervención ingresado no coincide con nuestros registros. Vuelva a intentarlo.') caption:'Menú administrador > Registro > Intervención de paciente'.

MessageBox notify:(self intervencionesDisponibles: espe).

espe:= Prompter prompt: 'Ingrese el código de intervención' caption:'Menú administrador > Registro > Intervención de paciente'.

].

p:= IntervencionRegistrada new.

p cargaDatos: fecha y: pac y:matricula y: inter y: (MessageBox confirm: '¿Está pagada?' ).

intervencionPaciente add: p.

rta:= MessageBox confirm: '¿Desea registrar otra intervención?' caption:'Menú administrador > Registro > Intervención de paciente'

]]].

**mi: registrarMedico**

"Se lleva a cabo el registro de médicos, validando además que siempre haya un médico disponible por cada especialidad"

|rta m matricula especialidad disponibilidad|

rta:= true.

[rta] whileTrue: [

matricula := (Prompter prompt: 'Ingrese la matrícula' caption:'Menú administrador > Registro > Médico').

(self existeMatricula: matricula) ifTrue: [

MessageBox warning: 'La matrícula ya existe. Por favor, ingrese otra.' caption:'Menú administrador > Registro > Médico'.

] ifFalse: [

especialidad := (Prompter prompt: 'Ingrese la especialidad' caption:'Menú administrador > Registro > Médico').

disponibilidad := (MessageBox confirm: '¿Está disponible?' caption:'Menú administrador > Registro > Médico').

(((self medicosDisponibles: especialidad y: 2)=false) and:[disponibilidad=false]) ifTrue: [

MessageBox warning: 'Debe haber un médico disponible por cada especialidad' caption:'Menú administrador > Registro > Médico'.

]

ifFalse:[

m:= Medico new.

m cargaDatos: matricula y: especialidad y: disponibilidad.

medico add: m.

].

rta:= MessageBox confirm: '¿Desea ingresar otro médico?' caption:'Menú administrador > Registro > Médico'.

].

].

**mi: registrarPaciente**

"Se registra un paciente validando todos los datos correspondientes"

|rta p dni|

rta:= true.

[rta] whileTrue: [

dni := (Prompter prompt: 'Ingrese DNI' caption:'Menú administrador > Registro > Paciente').

(self existeDNI: dni) ifTrue: [

MessageBox warning: 'El DNI ya existe. Por favor, ingrese otro.' caption:'Menú administrador > Registro > Paciente'.

] ifFalse: [

p:= Paciente new.

p cargaDatos: dni .

paciente add: p.

rta:= MessageBox confirm: 'Desea ingresar otro paciente?' caption:'Menú administrador > Registro > Paciente'

]].

**mi: validarIntervencion: unCodigo y: unaEspec**

"Valida que exista una intervención con el código y especialidad mandados como parámetros"

| m cod inter|

cod:= unCodigo.

m:= intervencion detect:[:i | i codigo=unCodigo and: [i especialidad=unaEspec] ] ifNone:[m:= 'no'.].

(m='no') ifTrue: [^false] ifFalse: [^true ]

**mi: validarMedico: unaMatricula y: unaEspecialidad**

"Valida que exista un médico en la colección teniendo en cuenta su matrícula, especialidad y condición"

| m med int|

med:= unaMatricula.

int:=unaEspecialidad.

m:= medico detect:[:i | i matricula=med and: [i especialidad=int and: [i condicion=true] ] ] ifNone:[m:= 'no'.].

(m='no') ifTrue: [^false] ifFalse: [^true ]

|  |
| --- |
| **#Paciente** |

**mi: cargaDatos: unDni**

"Desde aquí se cargan los datos del paciente. unDNI se pasa como argumento porque viene con datos validados"

|ob temp|

dni:=unDni.

nombre:=(Prompter prompt: 'Ingrese el nombre' caption:'Registro > Paciente').

apellido:=(Prompter prompt: 'Ingrese el apellido' caption: 'Registro > Paciente').

ob:=(MessageBox confirm:'¿Usted posee obra social?' caption: 'Registro > Paciente').

ob ifTrue: [obraSocial:=(Prompter prompt: 'Ingrese el nombre de su obra social' caption: 'Registro > Paciente').

(temp:=(Prompter prompt: 'Ingrese el porcentaje de cobertura' caption: 'Registro > Paciente')).

[((self esFlotante: temp)=false)] whileTrue: [

MessageBox errorMsg: 'Debe ingresar un número' caption:'Registro > Paciente'.

temp:=(Prompter prompt: 'Ingrese el porcentaje de cobertura' caption: 'Registro > Paciente').

((self esFlotante: temp))

].

porcCobertura:=temp asNumber asFloat.

].

ob ifFalse: [obraSocial:='No posee Obra Social'. porcCobertura:=0].

**mi: esFlotante: unNumero**

"Valida que el numero ingresado sea un entero"

|temp |

temp := true.

[(unNumero asNumber asFloat)] on: Error do: [:each | temp:= false].

^ temp

**mi: muestra**

"Muestra los datos relevantes de un determinado paciente. Utilizado en el menú consultas"

Transcript cr; show: nombre; tab; tab; show:apellido; tab;tab;show:dni printString.

(MessageBox notify: 'PACIENTE ', nombre, ' ', apellido, '

',

'DNI ', dni, '

',

'COBERTURA ', porcCobertura printString , '%' ,'

','

',

((obraSocial='No posee Obra Social') ifFalse: [' ✔️ ', obraSocial, '' ] ifTrue: [' ❌ No tiene Obra Social ']) caption:'Resultado encontrado'

)

**mi: nombre**

"getter"

^nombre

**mi: dni**

"getter"

^dni

**mi: obraSocial**

"getter"

^obraSocial

**mi: porcCobertura**

"getter"

^porcCobertura

|  |
| --- |
| **#Medico** |

**mi: cargaDatos: unaMatricula y: unaEspecialidad y: unaDisponibilidad**

"Se cargan los datos de un médico, unaMatricula, unaEspecialidad y unaDisponibilidad vienen validadas"

matricula:=unaMatricula.

nombre:=(Prompter prompt: 'Ingrese el nombre' caption:'Registro > Médico').

apellido:=(Prompter prompt: 'Ingrese el apellido' caption: 'Registro > Médico').

especialidad:=unaEspecialidad.

condicion:=unaDisponibilidad.

**mi: muestra**

"Muestra los datos de un médico. Utilizado en el menú consultas."

Transcript cr; show: nombre; tab; tab; show:apellido; tab;tab;show:matricula printString.

(MessageBox notify: 'PROFESIONAL ', nombre, ' ', apellido, '

',

'MATRÍCULA ', matricula, '

',

'ESPECIALIDAD ', especialidad ,'

','

',

(condicion ifTrue: [' ✔️ Disponible '] ifFalse: [' ❌No disponible ']) caption:'Resultado encontrado'

)

**mi: nombre**

"getter"

^nombre

**mi: apellido**

"getter"

^apellido

**mi: condicion**

"getter"

^condición

**mi: especialidad**

"getter"

^ especialidad

**mi: matricula**

"getter"

^matricula

|  |
| --- |
| **#IntervencionRegistrada** |

**mi: cargaDatos: unaFecha y: unPaciente y: unMedico y:unaIntervencion** **y:unaCondicion**

"Se permite la carga de datos para la intervención del paciente, todos los datos son validados"

fecha:=unaFecha.

paciente:=unPaciente.

medico:=unMedico.

intervencion:=unaIntervencion.

condicionPago:= unaCondicion.

**mi: condicionPago**

"getter"

^condicionPago

**mi: fecha**

"getter"

^fecha

**mi: intervencion**

"getter"

^intervención

**mi: medico**

"getter"

^medico

**mi: paciente**

"getter"

^paciente

|  |
| --- |
| **#Intervención** |

**mi: cargaDatos: unCod y: unaEspecialidad**

"Permite la carga de datos de una Intervención"

|temp|

codigo:=unCod.

descripcion:=(Prompter prompt: 'Ingrese la descripcion' caption:'Registro > Intervención').

especialidad:=unaEspecialidad.

(temp:=(Prompter prompt: 'Ingrese el arancel' caption: ' Registro > Intervención')).

[((self esFlotante: temp)=false)] whileTrue: [

MessageBox errorMsg: 'Debe ingresar un número' caption: ' Registro > Intervención'.

temp:=(Prompter prompt: 'Ingrese el arancel' caption:' Registro > Intervención').

((self esFlotante: temp))

].

arancel:=temp asNumber asFloat.

**mi: esFlotante: unNumero**

"Permite validar si un número es flotante"

|temp |

temp := true.

[(unNumero asNumber asFloat)] on: Error do: [:each | temp:= false].

^ temp

**mi: muestra**

"Muestra los datos de una intervención en particular. Utilizado en el menú consultas"

Transcript cr; show: codigo ; tab; tab; show:descripcion ; tab;tab;show:especialidad;tab;tab;show: arancel printString.

(MessageBox notify: 'DESCRIPCIÓN ', descripcion , '

',

'ESPECIALIDAD ', especialidad ,'

','

','ARANCEL ', arancel printString caption: 'Búsqueda de intervenciones > Intervención ',codigo).

**mi: codigo**

"getter"

^código

**mi: descripcion**

"getter"

^descripción

**mi: especialidad**

"getter"

^especialidad

**mi: arancel**

"getter"

^arancel

**mc: esFlotante: unNumero**

"Valida si un número es flotante. Este método lo va a usar la clase AltaComplejidad"

|temp |

temp := true.

[(unNumero asNumber asFloat)] on: Error do: [:each | temp:= false].

^ temp

|  |
| --- |
| **#AltaComplejidad** |

**mi: muestra**

Transcript cr; show: codigo ; tab; tab; show:descripcion ; tab;tab;show:especialidad;tab;tab;show: arancel printString.

(MessageBox notify: 'DESCRIPCIÓN ', descripcion , '

',

'ESPECIALIDAD ', especialidad ,'

','

','ARANCEL ', arancel printString , '

', 'ADICIONAL ', Adicional printString,'%' caption: 'Búsqueda de intervenciones > Intervención ',codigo).

**mc: cargaAdicional**

"Permite ingresar el adicional de las Intervenciones de alta complejidad"

|temp|

(temp:=(Prompter prompt: 'Ingrese el adicional' caption:'Intervenciones > Alta complejidad')).

[((super esFlotante: temp)=false)] whileTrue: [

MessageBox errorMsg: 'Debe ingresar un número' caption:'Intervenciones > Alta complejidad'.

temp:=(Prompter prompt: 'Ingrese el adicional' caption:'Intervenciones > Alta complejidad').

((super esFlotante: temp)).

].

Adicional:=temp asNumber asFloat.

**mc: adicional**

"getter"

^Adicional