Trabajo Práctico de Paradigmas de Programación

Conti Stéfano 52850

Dana Agustín 52935

Frías Ignacio 52093

Vitali Bruno 53137

Universidad Tecnológica Nacional

Com2K02

Cursada 2024

Trabajo Práctico Integrador

Administración de un Sanatorio

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.

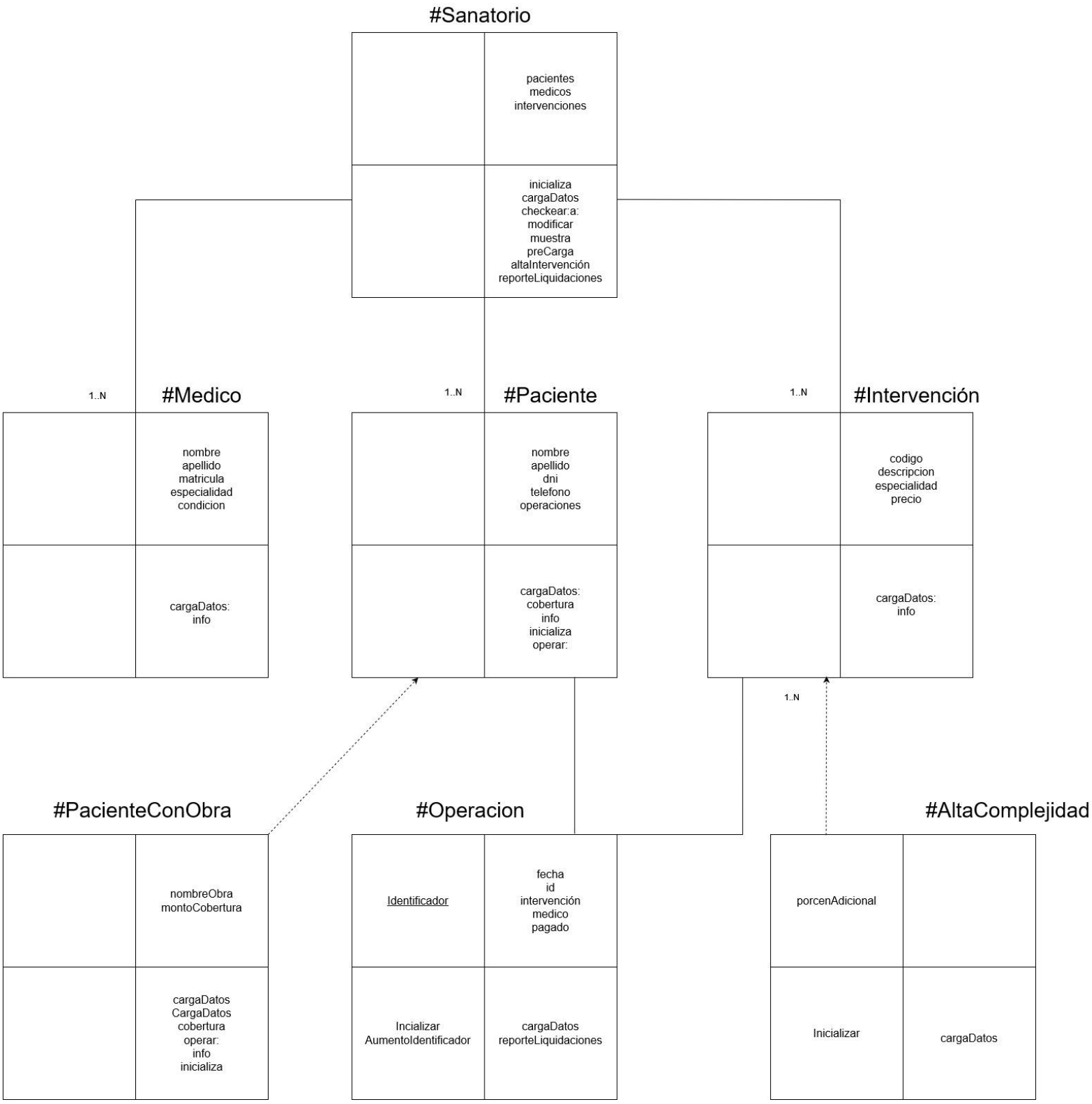
De cada paciente registra: documento de identidad, nombre y apellido, teléfono y el conjunto de las intervenciones quirúrgicas realizadas. Si el paciente cuenta con obra social además se registra el nombre de la misma y el monto de cobertura sobre el arancel.

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: identificador (valor numérico asignado automáticamente por el sistema), fecha de la intervención, intervención a efectuar, médico que interviene y condición de pagado o pendiente de pago. Se debe verificar que el médico interviniente tenga la misma especialidad que la intervención a realizar. Si el paciente no se encuentra registrado en el sistema al momento de realizar la intervención, debe ser dado de alta en ese momento.

Por último el sanatorio desea obtener un reporte de las liquidaciones pendientes de pago de los pacientes. El mismo debe contener la siguiente información: identificador, fecha, descripción, nombre y apellido del paciente, nombre y matrícula del médico interviniente, obra social del paciente (poner un guión medio si no tuviese) y el importe total a pagar.


#Sanatorio
pacientes
medicos
intervenciones
inicializa
cargaDatos
checkear:a:
modificar
muestra
preCarga
altaIntervención
reporteLiquidaciones
1..N
#Medico
nombre
apellido
matricula
especialidad
condicion
cargaDatos:
info
1..N
#Paciente
nombre
apellido
dni
telefono
operaciones
cargaDatos:
cobertura
info
inicializa
operar:
1..N
#Intervención
codigo
descripcion
especialidad
precio
cargaDatos:
info
1..N
#PacienteConObra
nombreObra
montoCobertura
cargaDatos
CargaDatos
cobertura
operar:
info
inicializa
#Operacion
Identificador
fecha
id
intervención
medico
pagado
Incializar
AumentoIdentificador
cargaDatos
reporteLiquidaciones
#AltaComplejidad
porcenAdicional
Inicializar
cargaDatos

Link del Diagrama de clases:
https://app.diagrams.net/#G1tpCrS16lguuPUvUDNMptlJ7DIiGHZg7t#%7B%22pageId%22%3A%22yNiqoTpbhQDUVH8JpZfU%22%7D

Nota aclaratoria: Con el propósito de que la visualización del diagrama de clases sea más prolija, obviamos describir en los métodos de instancia a los respectivos "setter's" y "getter's" de las variables de instancia, pero están presentes en el .pac.

```
| sanatorio op |
sanatorio:= Sanatorio new.
sanatorio inicializa.
sanatorio preCarga. "hace pre-carga con objetos"
[op ~= 0] whileTrue: [
op := (Prompter prompt: 'MENU/1. Solicitar intervención/2. Reporte de liquidaciones/3.
Carga/4. Modificar /0. Salir') asNumber.
(op = 1) ifTrue: [sanatorio altaIntervencion].
(op = 2) ifTrue: [sanatorio reporteLiquidaciones].
(op = 3) ifTrue:[sanatorio cargaDatos].
(op=4) ifTrue: [sanatorio modificar]. ].
```

Link del repositorio de Git con el Workspace y Paquete de Smalltalk con las clases y sus métodos: https://github.com/pepicont/ProyectoSmalltalk

Cuestionario:

a) Indique si encuentra clases abstractas. ¿Por qué son abstractas? Definir y dar ejemplos.

En nuestra codificación no utilizamos clases abstractas. Nosotros tomamos por clase abstracta a aquella clase que se define junto a sus atributos pero que nunca se instancia un objeto de esa clase mayor, sino de una de sus subclases.

Tomemos como ejemplo el final llamado FERRETERÍA en el cual tenemos una clase abstracta llamada Herramienta la cual tiene dos subclases: HerramientaLiviana y HerramientaPesada. En este caso nunca creamos una instancia de la clase Herramienta, sino que trabajamos siempre con instancias de sus subclases y nos ayudamos de la clase abstracta para definir métodos y así utilizarlos en sus subclases ayudándonos de la herencia.

b) Indique si encuentra clases concretas. ¿Por qué son concretas? Definir y dar ejemplos.

Si, nosotros utilizamos una amplia variedad de clases concretas, ya que de cada clase instanciamos al menos un objeto. En nuestro caso, por ejemplo, tenemos una clase concreta Paciente del cual instanciamos un objeto con atributos como por ejemplo nombre de la persona y apellido de la persona, o la clase Intervención, donde creamos un objeto con atributos como el nombre y código de la misma.

c) ¿Qué relaciones de herencia reconoce? Dar ejemplos. Explicar el concepto de herencia.

Es la relación entre las clases en las cuales una clase comparte su estructura y comportamiento de otra. Una subclase hereda todo de una superclase y sufre una modificación sobre su estructura y/o métodos (en el caso de los métodos, se denomina polimorfismo).

Por ejemplo, en nuestra clase AltaComplejidad (clase hija de Intervención) se heredan las variables de instancia como nombre y descripción y sus respectivos métodos “getter’s” y “setter’s”

d) ¿Qué relaciones de ensamble reconoce? Dar ejemplos. Explicar el concepto de ensamble.

Es la relación en la cual los objetos parte se relacionan con un objeto que es el todo. Esto es distinto de herencia, ya que en el ensamble hablamos de dos objetos distintos y la herencia relaciona clases. En nuestro caso, un ensamble es claramente visible en una instancia de la clase Operación, que como uno de sus atributos posee un objeto de la clase Intervención.

e) ¿Existen métodos polimórficos? Dar ejemplos. Explicar el concepto de polimorfismo.

Si, existen métodos polimórficos en nuestro código. Es la habilidad de dos o más objetos de poder responder a un mensaje con el mismo nombre, cada uno según su propia manera. Un ejemplo de métodos polimórficos en nuestro código se da con el llamado “info” el cual devuelve una cadena con la información de la clase. Lo utilizamos por ejemplo con la clase Paciente y Médico.

Paciente

info

    |header|

    header:= 'Nombre: ', nombre, ' ', apellido, ' DNI: ', dni, ' Obra Social: ---'.

^header

Medico

info

    |header disp|

    (condicion='1') ifTrue:[disp:='si'] ifFalse:[disp:='no'].

header:= 'Médico: ', nombre, ' ', apellido,String tab,' Matricula: ', matricula,String tab,'Disponible: ',disp.

^header

f) ¿Existen métodos con redefinición polimórfica? Dar ejemplos. Explicar el concepto de redefinición

Tiene que ver con el concepto de que toda clase puede redefinir un método heredado de la superclase. Un ejemplo de una redefinición polimórfica en nuestro código se da en la clase AltaComplejidad, la cual es una subclase de Intervención.

En esta oportunidad, al tener la subclase una variable de clase que es utilizada para modificar el precio de la intervención (PorcenAdicional), utilizamos el cargaDatos de las intervenciones comunes que ya teníamos creado y lo modificamos para utilizar la variable de clase.

cargaDatos: aCode

|recargo|

super cargaDatos: aCode. "Utiliza la herencia de Intervención"

recargo:=precio asNumber * PorcenAdicional.

precio:=(precio asNumber + recargo) asInteger . "porque precio está ingresado como string"

precio:=precio printString.

g) Identifique las variables de Clase utilizadas. Dar ejemplos. Explique qué son las variables de Clases y porque/cómo se utilizan en el presente trabajo.

Son variables accesibles por todas las instancias de la clase. En nuestro caso una variable de clase fue Identificador que era de la clase Intervención. Las usamos porque el Identificador tenía que ser único e incremental para cada instancia de clase y para eso, usamos la variable de clase, así tenía un valor único para todos los objetos de la clase.

La otra variable de clase que utilizamos fue PorcenAdicional (de la clase AltaComplejidad) que la utilizamos porque el porcentaje adicional de recargo tenía que ser el mismo para todas las instancias de esa clase.

h) Identifique las variables de Instancia utilizadas. Dar ejemplos. Explique qué son las variables de Instancia y sus diferencias con las variables de Clase.

Es una variable declarada dentro de una clase y se asocia a cada objeto que se crea a partir de esa clase. Tomando como ejemplo la clase Paciente, las variables de instancia de esta son:

nombre, apellido, telefono ,dni y operaciones.

La diferencia entre estas y las variables de clase es que para acceder a una variable de instancia necesitamos, como su nombre lo dice, una instancia de esta clase que tenga valores cargados en estas variables. Mientras que para acceder a las variables de clase solo necesitamos la clase, y no una instancia de la misma.

i) Identifique los métodos de Clase. Dar ejemplos. ¿Por qué son necesarios los métodos de clase?

Los métodos de clase son necesarios para definir comportamientos o responsabilidades que pertenecen a la clase en sí misma y no a sus instancias.

Los métodos de clase que utilizamos fueron “AumentoIdentificador” e “Inicializar” en la clase Operación y en la clase AltaComplejidad “Inicializar”

j) Identifique mensajes unarios, binarios y de palabra clave. De ejemplos de cada uno de ellos.

Los mensajes unarios son aquellos que no tienen argumentos, solo interviene el objeto receptor, los binarios son un tipo de mensaje especial que solo toma un argumento, mientras que los mensajes de palabra clave son mensajes en lo que se puede enviar uno o más argumentos, el selector está compuesto por una o más palabras clave y cada palabra clave precedida de un argumento.

Ejemplo de mensaje unario ya utilizado anteriormente:

"med info". Dónde info es:

```
*Medico*

info

    |header disp|

    (condicion='1') ifTrue:[disp:='si'] ifFalse:[disp:='no'].

    header:= 'Médico: ', nombre, ' ', apellido,String tab,' Matricula: ', matricula,String tab,'Disponible: ',disp.

^header
```

Ejemplo de mensaje binario ya utilizado anteriormente:

"*Identificador + 1*"

Ejemplo de mensaje de palabra clave utilizado:

“self checkear: inter a: med”. Donde

Sanatorio

checkear: anIntervention a: aDoctor "Checkea que coincida la especialidad del médico con la de la intervención"

|especIntervencion especMedico rta|

especIntervencion:= anIntervention especialidad.

especMedico:=aDoctor especialidad.

(especIntervencion=especMedico) ifTrue: [rta:=1] ifFalse:[rta:=nil].

^rta

Paquete:

```
| package |
package := Package name: 'tp'.
package paxVersion: 1;
    basicComment: ''.

package classNames
    add: #AltaComplejidad;
    add: #Intervencion;
    add: #Medico;
    add: #Operacion;
    add: #Paciente;
    add: #PacienteConObra;
    add: #Sanatorio;
    yourself.

package binaryGlobalNames: (Set new
    yourself).

package globalAliases: (Set new
```

yourself).

package setPrerequisites: #(
'..\Core\Object Arts\Dolphin\Base\Dolphin'
'..\Core\Object Arts\Dolphin\Base\Dolphin Legacy Date & Time'
'..\Core\Object Arts\Dolphin\Base\Dolphin Message Box'
'..\Core\Object Arts\Dolphin\MVP\Presenters\Prompters\Dolphin Prompter').

package!

"Class Definitions"!

Object subclass: #Intervencion
instanceVariableNames: 'codigo descripcion especialidad precio'
classVariableNames: "
poolDictionaries: "
classInstanceVariableNames: "!
Object subclass: #Medico
instanceVariableNames: 'nombre apellido matricula especialidad condicion'
classVariableNames: "
poolDictionaries: "
classInstanceVariableNames: "!

```
Object subclass: #Operacion
    instanceVariableNames: 'fecha id intervencion medico pagado montoAPagar'
    classVariableNames: 'Identificador'
    poolDictionaries: ''
    classInstanceVariableNames: ''!
Object subclass: #Paciente
    instanceVariableNames: 'nombre apellido dni telefono operaciones'
    classVariableNames: ''
    poolDictionaries: ''
    classInstanceVariableNames: ''!
Object subclass: #Sanatorio
    instanceVariableNames: 'medicos pacientes intervenciones'
    classVariableNames: ''
    poolDictionaries: ''
    classInstanceVariableNames: ''!
Intervencion subclass: #AltaComplejidad
    instanceVariableNames: ''
    classVariableNames: 'PorcenAdicional'
    poolDictionaries: ''
    classInstanceVariableNames: ''!
Paciente subclass: #PacienteConObra
    instanceVariableNames: 'nombreObra montoCobertura'
```

classVariableNames: "

poolDictionaries: "

classInstanceVariableNames: "!

"Global Aliases"!

"Loose Methods"!

"End of package definition"!

"Source Globals"!

"Classes"!

Intervencion guid: (GUID fromString: '{31a9d5b1-46d7-46e6-87c5-7f6ba8d4b47a}')!

Intervencion comment: "!

!Intervencion categoriesForClass!Kernel-Objects! !

!Intervencion methodsFor!

cargaDatos: aCode

```
codigo := aCode .
descripcion := Prompter prompt: 'Ingrese la descripcion de la intervención: '.
especialidad := Prompter prompt: 'Ingrese la especialidad de la intervención: '.
precio := Prompter prompt: 'Ingrese el precio de la intervención: '.




!

codigo
    ^codigo!

codigo: anObject
    codigo := anObject!

descripcion
    ^descripcion!

descripcion: anObject
    descripcion := anObject!
```

especialidad

^especialidad!

especialidad: anObject

especialidad := anObject!

info

|informacion|

informacion:='Codigo: ',codigo, String tab,'Descripción: ',descripcion,String tab,'Especialidad: ',especialidad,String tab,'Precio: ',precio.

^informacion

"Arma un string con toda la información de la intervención para no ensuciar el Transcript"!

precio

^precio!

precio: anObject

precio := anObject! !

!Intervencion categoriesForMethods!

cargaDatos:!public! !

codigo!accessing!private! !

codigo:!accessing!private! !

descripcion!accessing!private! !

descripcion:!accessing!private! !

especialidad!accessing!private! !

especialidad:!accessing!private! !

info!public! !

precio!accessing!private! !

precio:!accessing!private! !

!

Medico guid: (GUID fromString: '{36eb3dec-f645-4ffa-97d3-a2a049e71ded}')!

Medico comment: "!

!Medico categoriesForClass!Kernel-Objects! !

!Medico methodsFor!

apellido

    ^apellido!

apellido: anObject

    apellido := anObject!

cargaDatos: aMatricula

    nombre := Prompter prompt: 'Ingrese el nombre del médico a ingresar'.

```
    apellido := Prompter prompt: 'Ingrese el apellido del médico a ingresar'.

    matricula:= aMatricula.

    especialidad := Prompter prompt: 'Ingrese la especialidad del médico a
ingresar'.

    condicion:=Prompter prompt: 'Ingrese disponibilidad. 1-disponible/2-No
disponible'.

!



condicion

    ^condicion!



condicion: anObject

    condicion := anObject!



especialidad

    ^especialidad!



especialidad: anObject

    especialidad := anObject!



info

    |header disp|

    (condicion='1') ifTrue:[disp:='si'] ifFalse:[disp:='no'].
```

```
    header:= 'Médico: ', nombre, ' ', apellido,String tab,' Matricula: ',
matricula,String tab,'Disponible: ',disp.

^header!


matricula

    ^matricula!


matricula: anObject

    matricula := anObject!


nombre

    ^nombre!


nombre: anObject

    nombre := anObject! !

!Medico categoriesForMethods!

apellido!accessing!private! !

apellido:!accessing!private! !

cargaDatos:!public! !

condicion!accessing!private! !

condicion:!accessing!private! !

especialidad!accessing!private! !
```

especialidad:!accessing!private! !

info!public! !

matricula!accessing!private! !

matricula:!accessing!private! !

nombre!accessing!private! !

nombre:!accessing!private! !

!

Operacion guid: (GUID fromString: '{4b2822fc-12b4-4b5d-bc51-22f4dd7cf7c4}')!

Operacion comment: "!

!Operacion categoriesForClass!Kernel-Objects! !

!Operacion methodsFor!

cargaDatos:anObject a: anotherObject con:unaCobertura

|cober|

cober:= unaCobertura asNumber .

fecha:= Date fromString: (Prompter prompt: 'Ingrese la fecha de la operación dd/mm/aaaa').

[fecha>Date today] whileFalse: ["Valida la fecha ingresada"

MessageBox notify: 'La fecha ingresada debe ser mayor a la de hoy'.

fecha:= Date fromString: (Prompter prompt: 'Ingrese la fecha de la operación dd/mm/aaaa').].

Operacion AumentoIdentificador. "incrementa el id interno de las operaciones"

id:= Identificador.

intervencion:= anObject.

montoAPagar:= ((intervencion precio) asNumber) - cober. "calcula el monto final dependiendo de la cobertura"

medico:=anotherObject.

pagado:= Prompter prompt: 'Ingrese 1-Pagado/2-Pendiente de pago'.

[pagado~='1' and:[pagado ~='2']] whileTrue: ["Valida que el valor ingresado sea correcto"

MessageBox notify: 'Por favor ingrese un valor correcto'.

pagado:= Prompter prompt: 'Ingrese 1-Pagado/2-Pendiente de pago'.]

!

fecha

    ^fecha!

fecha: anObject

    fecha := anObject!

```
id

    ^id!


id: anObject

    id := anObject!


info

    |header|

    header:= 'Identificador: ', id displayString, ' ', 'Fecha: ',fecha displayString, String cr,String tab, String tab,' Descripción: ', intervencion descripcion, String tab,'Monto: ', montoAPagar displayString.

^header!


intervencion

    ^intervencion!


intervencion: anObject

    intervencion := anObject!


medico

    ^medico!
```

```
medico: anObject
    medico := anObject!

pagado
    ^pagado!

pagado: anObject
    pagado := anObject! !
!Operacion categoriesForMethods!
cargaDatos:a:con:!public! !
fecha!accessing!private! !
fecha:!accessing!private! !
id!accessing!private! !
id:!accessing!private! !
info!public! !
intervencion!accessing!private! !
intervencion:!accessing!private! !
medico!accessing!private! !
medico:!accessing!private! !
pagado!accessing!private! !
pagado:!accessing!private! !
!
```

```
!Operacion class methodsFor!

AumentoIdentificador
Identificador:=Identificador+1.
!

Inicializar
    Identificador:=0.! !
!Operacion class categoriesForMethods!
AumentoIdentificador!public! !
Inicializar!public! !
!

Paciente guid: (GUID fromString: '{217f8510-1557-4bcb-b9a6-bda460760c44}')!
Paciente comment: "!
!Paciente categoriesForClass!Kernel-Objects! !
!Paciente methodsFor!

apellido
    ^apellido!
```

```
apellido: anObject
    apellido := anObject!

cargaDatos:anObject
dni:=anObject.
nombre :=Prompter prompt: 'ingrese nombre del paciente '.
apellido :=Prompter prompt: 'ingrese apellido del paciente '.
telefono :=Prompter prompt: 'ingrese telefono del paciente '.!

cobertura
^'0'!

dni
    ^dni!

dni: anObject
    dni := anObject!

info
    |header|
    header:= 'Nombre: ', nombre, ' ', apellido, ' DNI: ', dni, ' Obra Social: ---'.
^header!
```

```
inicializa
operaciones := OrderedCollection new. !

nombre
    ^nombre!

nombre: anObject
    nombre := anObject!

operaciones
    ^operaciones!

operaciones: anObject
    operaciones := anObject!

operar: anOperacion
self operaciones add: anOperacion.

!

telefono
```

```
    ^telefono!


telefono: anObject
    telefono := anObject! !
!Paciente categoriesForMethods!
apellido!accessing!private! !
apellido:!accessing!private! !
cargaDatos:!public! !
cobertura!public! !
dni!accessing!private! !
dni:!accessing!private! !
info!public! !
inicializa!public! !
nombre!accessing!private! !
nombre:!accessing!private! !
operaciones!accessing!private! !
operaciones:!accessing!private! !
operar:!public! !
telefono!accessing!private! !
telefono:!accessing!private! !
!
```

Sanatorio guid: (GUID fromString: '{14193734-aa95-4daa-be4c-41a45d316d8e}')!

Sanatorio comment: "!

!Sanatorio categoriesForClass!Kernel-Objects! !

!Sanatorio methodsFor!

altaIntervencion

| operacion paciente dni ob flag1 flag2 inter inttemp med medint flag3 flag4 cobertura|

dni:= Prompter prompt:'Ingrese dni del paciente'.

paciente:= pacientes detect:[:pas| pas dni = dni] "Busca el paciente del dni ingresado en la colección de pacientes"

ifNone: [ob:=Prompter prompt: '¿Tiene obra social? 1-Si/2-No'. "Pregunta si el ingresado tiene o no obra social"

(ob='1') ifTrue: [paciente:= PacienteConObra new]

ifFalse:[

paciente:= Paciente new].

paciente inicializa. "inicializa la ordered collection de operaciones del paciente"

paciente cargaDatos:dni."Carga los datos del paciente"

cobertura:= paciente cobertura.

pacientes add:paciente]. "Agrega el paciente registrado a la colección de pacientes del sanatorio"

```
    cobertura:= paciente cobertura. "metodo polimórfico entre pacientes con y sin
obra para traer la cobertura del paciente"

    MessageBox notify: 'La información de los médicos y las intervenciones está
disponible en el Transcript'.

    self muestra. "Muestra la informaciónd de todos los médicos y las
intervenciones cargados"

    operacion:= Operacion new.

    inttemp:= Prompter prompt: 'Ingrese el código de la intervención a realizar/
0.salir'. "NUEVA LINEA"

    inter:= intervenciones detect:[:in | (in codigo)=inttemp ] ifNone: [nil].  "busca
la intervencion en la colección de intervenciones"

    (inter isNil and: [inttemp ~= '0'] ) ifTrue: [MessageBox notify: 'La intervención
ingresada no se encuentra en la colección' ]. "notifica al usuario"

    "si la intervención se encontro y el código ingresado es distinto de 0 ,ingresa"

    [inter isNil or: [inttemp='0'] ] whileFalse: [

        medint:= Prompter prompt:'Ingrese matricula del médico/ 0.salir'.

        med:= medicos detect: [:med1 | (med1 matricula) = medint] ifNone:[nil].
"Busca el médico en la colección de medicos"

        (med isNil and: [medint~= '0'] ) ifTrue: [MessageBox notify: 'El médico
ingresado no se encuentra en la colección de médicos' ]"si no se enceuntra el
medico, notifica al usuario""Si el codigo es 0 ,no se notifica nada".

        (med isNil and: [medint ='0']) ifTrue:[inttemp :='0'].  "Si ingreso 0.Salir
actualiza inttemp para que siga el while "

        (med isNil) ifFalse: [ "Entra si encontró el médico ingresado"

                ((self checkear: inter a: med) isNil ) ifTrue:
[MessageBox notify: 'El médico ingresado no coincide con la especialidad de la
intervención'] "Checkeamos si coincide la especialidad del médico con la de la
intervención"
```

ifFalse: [(med condicion ='1')"Checkea que el médico esté disponible" ifTrue: [operacion cargaDatos: inter a: med con:cobertura"Carga los datos del objeto operación con el médico, la intervención y calcula el monto final dependiendo de si el paciente tiene o no cobertura de la obra social". paciente operar:operacion. MessageBox notify: 'Operación cargada con éxito' "Añade la operación a la OC de operaciones del paciente"] ifFalse:[MessageBox notify: 'Lamentamos informarle que el medico no esta disponible'.]]. "Carga el id, la fecha, la intervención y el médico"

inttemp:='0'. "Sale del bucle"

]

].

!

cargaDatos

|opc paciente intervencion matricula dni cod var medico complejo op|

opc:=(Prompter prompt: '1-Ingresar Pacientes/2-Ingresar Medico/ 3-Ingresar Intervenciones') asNumber .

(opc=1) ifTrue: [

op:= Prompter prompt: 'El paciente tiene obra social? 1-Si/2-No'.

(op='1') ifTrue:[paciente:= PacienteConObra new] ifFalse:[paciente:= Paciente new].

paciente inicializa.

dni:=Prompter prompt:'Ingrese dni del paciente a cargar'.

var:=(pacientes detect:[:ele | ele dni= dni] ifNone:[paciente cargaDatos:dni .pacientes add:paciente. MessageBox notify: 'Paciente cargado con éxito'. ^nil]).

(var isNil) ifFalse: [MessageBox notify: 'paciente ya se encuentra cargado'].

].

(opc=2) ifTrue: [

medico:=Medico new.

matricula:=Prompter prompt:'Ingrese la matricula del médico a cargar'.

var:=(medicos detect:[:ele | ele matricula= matricula] ifNone:[medico cargaDatos:matricula. medicos add:medico. MessageBox notify: 'Médico cargado con éxito'. ^nil]).

(var isNil) ifFalse:[MessageBox notify:'El médico ya se encuentra cargado'].

].

(opc=3) ifTrue: [

complejo:=Prompter prompt: '¿La intervención es de alta complejidad? 1-Si/2-No'.

(complejo='1') ifTrue: [intervencion := AltaComplejidad new] ifFalse:[intervencion :=Intervencion new].

cod:= Prompter prompt: 'Ingrese el codigo de la intervención a cargar'.

var:=(intervenciones detect:[:elem | elem codigo = cod] ifNone:[intervencion cargaDatos: cod. intervenciones add:intervencion. MessageBox notify:'Intervención cargada con éxito'. ^nil]).

(var isNil) ifFalse:[MessageBox notify:'La intervención ya se encuentra cargada'].

].

```
!

checkear: anIntervention a: aDoctor "Checkea que coincida la especialidad del médico con la de la intervención"

|especIntervencion especMedico rta|

especIntervencion:= anIntervention especialidad.

especMedico:=aDoctor especialidad.

(especIntervencion=especMedico) ifTrue: [rta:=1] ifFalse:[rta:=nil].

^rta!


inicializa

    pacientes:= OrderedCollection new.

    medicos:= OrderedCollection new.

    intervenciones:= OrderedCollection new.

    AltaComplejidad Inicializar. "Inicializa el porcentaje de recargo que tienen las operaciones de este tipo"

    Operacion Inicializar. "Inicializamos el contador para los id en 0"!


intervenciones

    ^intervenciones!


intervenciones: anObject
```

```
    intervenciones := anObject!


medicos

    ^medicos!


medicos: anObject

    medicos := anObject!


modificar

|opcion matricula med pac dni cod oper|

opcion:=(Prompter prompt: '1-Modificar estado Medico/2-Pagar operación/0-
Salir') asNumber asInteger.

(opcion=1) ifTrue: [ MessageBox notify: 'Se han mostrado en el transcript los
datos de los médicos'.

                    self muestra. "muestra datos médicos"

                    matricula := Prompter prompt: 'Ingrese la matrícula del
médico a modificar'. "ingresa la matricula del médico"

                    med:= medicos detect: [:medico | medico
matricula=matricula ] ifNone:[matricula:=0 ]. "busca al médico en la colección
medico"

                    (matricula ~= 0) ifTrue: [med condicion:(Prompter prompt:
'Ingrese disponibilidad. 1-disponible/2-No disponible').

                    MessageBox notify: 'La disponibilidad fue cambiada con
éxito'.

 ] ifFalse:[MessageBox notify: 'La matricula ingresada no corresponde a un
medico cargado']. ]. "Cambia el estado de un médico"
```

(opcion=2) ifTrue: [self reporteLiquidaciones. "muestra las operaciones pendientes"

MessageBox notify: 'Las operaciones en deuda fueron impresas en el Transcript'.

dni:=Prompter prompt: 'Ingrese el dni del paciente en deuda'.

pac:= pacientes detect: [:paciente | paciente dni=dni] ifNone:[dni:=0]. "busca el paciente en la colección pacientes"

(dni=0) ifTrue: [MessageBox notify: 'El dni ingresado no corresponde a un paciente cargado'] ifFalse:[

cod:=(Prompter prompt: 'ingrese el código de la operación a pagar') asNumber.

oper:= (pac operaciones) detect: [:operacion | operacion id = cod] ifNone:[oper := 0].

(oper=0) ifTrue: [MessageBox notify: 'El codigo de operación ingresado no corresponde a una operación'] ifFalse:[

oper pagado:'1'. "Registramos la operación como pagada"

MessageBox notify: 'La operación ha sigo pagada con éxito'.

]]].

!

muestra

"Muestra medicos e intervenciones cargadas"

Transcript clear.

Transcript tab;tab;tab;show:'MEDICOS:';cr.

```
Transcript show:'-------------------------------------------------------------------------------
-----------------------------------------';cr.


medicos do:[:med| Transcript show: (med info) printString
;tab;show:'Especialidad: ';show: (med especialidad) printString;tab;cr ].

Transcript cr.

Transcript tab;tab;tab;show:'INTERVENCIONES';cr.

Transcript show:'-------------------------------------------------------------------------------
-----------------------------------------';cr.

intervenciones do:[:int|  Transcript show: (int info) printString;cr ].




!


pacientes

    ^pacientes!


pacientes: anObject

    pacientes := anObject!


preCarga

|int med pac|

int:=Intervencion new.
```

int codigo: '1'.

int descripcion: 'Apertura de rodilla'.

int especialidad: 'traumatologia'.

int precio: '1000'.

intervenciones add: int. "carga una intervención"

int:=Intervencion new.

int codigo: '2'.

int descripcion: 'Operación de corazón'.

int especialidad: 'cardiologia'.

int precio: '10000'.

intervenciones add: int. "Hasta acá cargamos dos intervenciones por defecto para no andar cargando a mano cada vez que"

"queremos probar el sistema"

med:=Medico new.

med nombre: 'Stefano'.

med apellido: 'Conti'.

med matricula:'52850'.

med especialidad: 'traumatologia'.

med condicion: '2'. "condición=2 significa médico no disponible"

medicos add: med. "Cargamos 1 médico"

```
med:=Medico new.
med nombre: 'Agustin'.
med apellido: 'Dana'.
med matricula:'52935'.
med especialidad: 'cardiologia'.
med condicion: '1'. "condición=1 significa médico  disponible"
medicos add: med. "Cargamos otro médico"

pac:=PacienteConObra new.
pac nombre:'Agustin'.
pac apellido:'Dana'.
pac inicializa.
pac nombreObra:'osde'.
pac montoCobertura:'500'.
pac dni:'45949176'.
pac telefono:'3413946996'.
pacientes add:pac.  "Cargamos un paciente"

pac:=Paciente new.
pac nombre:'Stefano'.
pac apellido:'Conti'.
pac inicializa.
```

pac dni:'46132662'.

pac telefono:'3411233443'.

pacientes add:pac. "Cargamos otro paciente"

!

reporteLiquidaciones

Transcript clear.

pacientes do:[:paciente | (paciente operaciones) do:[:operacion | ((operacion pagado='2')) ifTrue:[Transcript show: paciente info printString;cr;tab;show: operacion info printString;cr;tab;tab;tab; show:operacion medico info printString;cr;cr]]].

"Recorre la colección pacientes, luego dentro de pacientes recorre sus operaciones y si la operación no está paga, armamos un header con la info del paciente, medico y la operacion por separado y printeamos todo con TS."

"Operacion medico accede al objeto medico dentro de operación y info hace el header con la info del med"

MessageBox notify: 'El reporte fue cargado en el transcript'.! !

!Sanatorio categoriesForMethods!

altaIntervencion!public! !

cargaDatos!public! !

checkear:a:!public! !

inicializa!public! !

intervenciones!accessing!private! !

intervenciones:!accessing!private! !

medicos!accessing!private! !

medicos:!accessing!private! !

modificar!public! !

muestra!public! !

pacientes!accessing!private! !

pacientes:!accessing!private! !

preCarga!public! !

reporteLiquidaciones!public! !

!

AltaComplejidad guid: (GUID fromString: '{01f3471a-934b-45bb-969d-5bd476675db6}')!

AltaComplejidad comment: "!

!AltaComplejidad categoriesForClass!Kernel-Objects! !

!AltaComplejidad methodsFor!

cargaDatos: aCode

|recargo|

```
super cargaDatos: aCode. "Utiliza la herencia de Intrevención"

recargo:=precio asNumber * PorcenAdicional.

precio:=(precio asNumber + recargo) asInteger . "porque precio está ingresado como string"

precio:=precio printString.! !

!AltaComplejidad categoriesForMethods!

cargaDatos:!public! !

!


!AltaComplejidad class methodsFor!


Inicializar

 PorcenAdicional:= 0.2 .!


porcenAdicional

    ^PorcenAdicional!


porcenAdicional: anObject

    PorcenAdicional := anObject! !

!AltaComplejidad class categoriesForMethods!

Inicializar!public! !

porcenAdicional!accessing!private! !
```

porcenAdicional:!accessing!private! !

!

PacienteConObra guid: (GUID fromString: '{8d99fb2f-b3a7-4406-8ac6-4db319d7375a}')!

PacienteConObra comment: "!

!PacienteConObra categoriesForClass!Kernel-Objects! !

!PacienteConObra methodsFor!

cargaDatos:anDni

"dni:=Prompter prompt: 'ingrese dni del paciente '."

dni:=anDni.

nombre :=Prompter prompt: 'ingrese nombre del paciente '.

apellido :=Prompter prompt: 'ingrese apellido del paciente '.

telefono :=Prompter prompt: 'ingrese telefono del paciente '.

nombreObra :=Prompter prompt: 'ingrese obra social del paciente'.

montoCobertura :=Prompter prompt: 'ingrese monto de cobertura de la obra del paciente'.!

cobertura

^montoCobertura!

incializa

```
operaciones := OrderedCollection new.

!


info

    |header|

    header:= 'Nombre: ', nombre, ' ', apellido, ' DNI: ', dni, ' Obra Social: ', nombreObra.

^header!


montoCobertura

    ^montoCobertura!


montoCobertura: anObject

    montoCobertura := anObject!


nombreObra

    ^nombreObra!


nombreObra: anObject

    nombreObra := anObject!


operar: anOperacion
```

super operar:anOperacion.! !

!PacienteConObra categoriesForMethods!

cargaDatos:!public! !

cobertura!public! !

incializa!public! !

info!public! !

montoCobertura!accessing!private! !

montoCobertura:!accessing!private! !

nombreObra!accessing!private! !

nombreObra:!accessing!private! !

operar:!public! !

!

"Binary Globals"!