Trabajo Práctico de Paradigmas de Programación

# La clínica de adelgazamiento

Cohelo Paulo 45891

Grancelli Eliseo 46935

Paratore Eneas 47018

Rodriguez Ignacio 45767

Nro. de comisión: 204

Año: 2020

## |Enunciado de la situación problemática

La clínica de adelgazamiento "CormiShot" desea implementar un sistema para gestionar la información clínica de sus pacientes y la información de los pagos y cobros a realizar.

Esta clínica presta tres tipos de servicios:

- **Consultas Clínicas:** Que se hacen con un profesional, que puede ser médico clínico, nutricionista o psicólogo. Las consultas clínicas tienen un costo que depende de la tarifa de cada profesional.
- **Reuniones grupales:** en las que los pacientes pueden compartir sus experiencias. Estas reuniones tienen un profesional que las coordina. Cada reunión tiene su costo al que se le suma la tarifa del profesional que coordina.
- **Sesiones de gimnasia:** son reuniones al aire libre donde se realizan diferentes tipos de ejercicios especializados. Todas las sesiones tienen el mismo costo y se cobran por cada vez que el paciente asiste.

Cada uno de estos servicios tiene una descripción.

Además, en esta clínica se atienden dos tipos de pacientes:

- **Pacientes particulares,** a los que se cobra por cada servicio prestado.
- **Pacientes de obra social,** a los que se les cobra un porcentaje de cada servicio, que es distinto para cada obra social.

De cada paciente se registra, además: tipo y nro. de documento, apellido y nombres, dirección, teléfono y obra social si corresponde.

De los profesionales que asisten a las consultas clínicas y a las reuniones grupales, se registra: tipo y nro. de documento, apellido y nombres, dirección, teléfono, matrícula, tarifa que cobra por consulta y tarifa que cobra por coordinar reuniones de grupo.

De las obras sociales se registra: código, nombre, porcentaje de cobertura al paciente

Cada vez que el paciente ingresa a la clínica para realizar alguna actividad, se confecciona una minuta de actividad con los siguientes datos: fecha, paciente que asiste, el peso actual del paciente y el servicio que va a recibir. Si el servicio requiere de un profesional asistente, se lo asignará en el momento.

Se pide:

1- Utilizando el paradigma orientado a 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- Utilizando el lenguaje Smalltalk, codificar todos los métodos necesarios para:

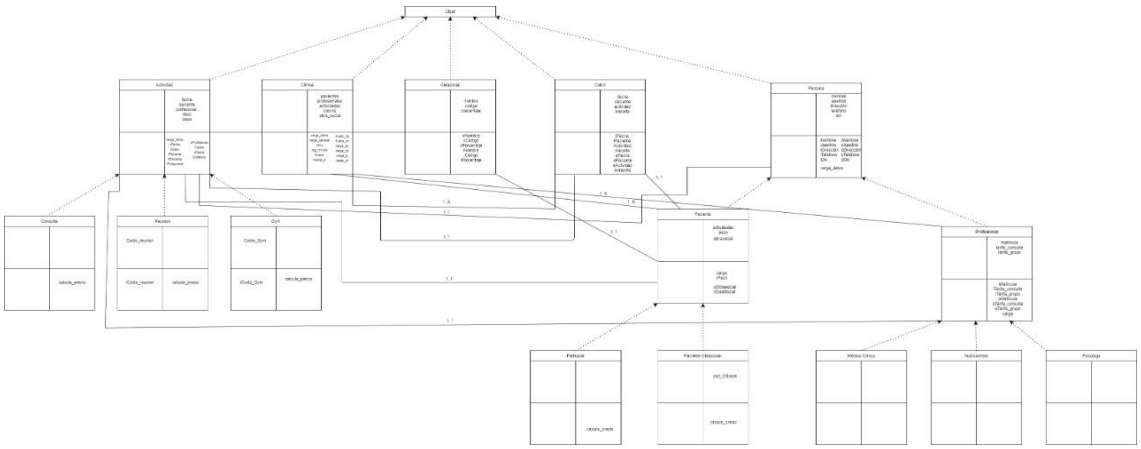
- Registrar las minutas de actividad de los pacientes que asisten a la clínica
- Obtener un listado de facturación para una obra social en un período determinado, con el detalle de los servicios prestados a los pacientes que tengan esa obra social.
  Las fechas de inicio y fin del listado, así como el código de la obra social se ingresarán por teclado. El listado deberá contener: nombre y porcentaje de cobertura de la obra social y por cada actividad facturada: fecha, apellido y nombre del paciente, descripción de la actividad realizada, costo de la actividad e importe a pagar por la obra social. Además, al final del listado deberá mostrar el importe total que debe abonar la obra social.

- Un **menú** que permita acceder a las acciones indicadas en los ítems anteriores y el correspondiente **programa principal.**

**Nota:** Considerar que los datos de los pacientes, las obras sociales y los profesionales ya están registrados en el sistema. **<Omitido>**

Se evaluará la óptima utilización de los recursos del paradigma y del lenguaje

## Diagrama de Clases



## Programa principal utilizado desde el Workspace para ejecutar el sistema

```
|clin|
clin := Clinica new.
clin menu.
```

## Listado completo de la programación de cada una de las clases

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


package classNames
        add: #Actividad;
        add: #Clinica;
        add: #Cobro;
        add: #Consulta;
        add: #Gym;
        add: #MedicoClinico;
        add: #Nutricionista;
        add: #Obra_Social;
        add: #Paciente;
```

```
        add: #Particular;
        add: #Pcte_ObraSocial;
        add: #Persona;
        add: #Profesional;
        add: #Psicologo;
        add: #Reunion;
        yourself.

package binaryGlobalNames: (Set new
        yourself).

package globalAliases: (Set new
        yourself).

package setPrerequisites: #(
        '..\..\..\Documents\Dolphin Smalltalk 7\Core\Object
Arts\Dolphin\Base\Dolphin'
        '..\..\..\Documents\Dolphin Smalltalk 7\Core\Object
Arts\Dolphin\Base\Dolphin Legacy Date & Time'
        '..\..\..\Documents\Dolphin Smalltalk 7\Core\Object
Arts\Dolphin\Base\Dolphin Message Box'
        '..\..\..\Documents\Dolphin Smalltalk 7\Core\Object
Arts\Dolphin\MVP\Presenters\Prompters\Dolphin Prompter').

package!
```

"Class Definitions"!

Object subclass: #Actividad

instanceVariableNames: 'fecha paciente profesional descripcion peso band'

classVariableNames: ''

poolDictionaries: ''

classInstanceVariableNames: ''!

Object subclass: #Clinica

instanceVariableNames: 'profesionales pacientes actividades cobros obra_social'

classVariableNames: ''

poolDictionaries: ''

classInstanceVariableNames: ''!

Object subclass: #Cobro

instanceVariableNames: 'fecha paciente actividades importe'

classVariableNames: ''

poolDictionaries: ''

classInstanceVariableNames: ''!

Object subclass: #Obra_Social

instanceVariableNames: 'nombre codigo porcentaje_cobertura'

classVariableNames: ''

poolDictionaries: ''

classInstanceVariableNames: ''!

Object subclass: #Persona

instanceVariableNames: 'nombre apellido direccion telefono dni'

classVariableNames: ''

poolDictionaries: ''

classInstanceVariableNames: ''!

Actividad subclass: #Consulta

instanceVariableNames: ''

classVariableNames: ''

poolDictionaries: ''

classInstanceVariableNames: ''!

Actividad subclass: #Gym

instanceVariableNames: ''

classVariableNames: 'Costo_sesion'

poolDictionaries: ''

classInstanceVariableNames: ''!

Actividad subclass: #Reunion

instanceVariableNames: ''

classVariableNames: 'Costo_reunion'

poolDictionaries: ''

classInstanceVariableNames: ''!

Persona subclass: #Paciente

instanceVariableNames: 'actividades peso obraSocial cod'

classVariableNames: 'Prox_Num'

poolDictionaries: ''

classInstanceVariableNames: ''!

Persona subclass: #Profesional

instanceVariableNames: 'matricula tarifa_consulta tarifa_grupo'

classVariableNames: ''

poolDictionaries: ''

classInstanceVariableNames: ''!

Paciente subclass: #Particular

instanceVariableNames: ''

classVariableNames: ''

poolDictionaries: ''

classInstanceVariableNames: ''!

Paciente subclass: #Pcte_ObraSocial

instanceVariableNames: 'cod_OSocial'

classVariableNames: ''

poolDictionaries: ''

classInstanceVariableNames: ''!

Profesional subclass: #MedicoClinico

instanceVariableNames: ''

classVariableNames: ''

poolDictionaries: ''

classInstanceVariableNames: ''!

Profesional subclass: #Nutricionista

instanceVariableNames: ''

classVariableNames: ''

poolDictionaries: ''

classInstanceVariableNames: ''!

Profesional subclass: #Psicologo

instanceVariableNames: "

classVariableNames: "

poolDictionaries: "

classInstanceVariableNames: "!

"Global Aliases"!

"Loose Methods"!

"End of package definition"!

"Source Globals"!

"Classes"!

Actividad guid: (GUID fromString: '{34c075a2-4e4e-4e20-9c8b-c1b5bed1907b}')!

Actividad comment: "!

!Actividad categoriesForClass!Kernel-Objects! !

!Actividad methodsFor!

iBand

""

band:=1.!

```
iDesc: a

descripcion:=a.!

iFecha
"Ingreso fecha act"

fecha:= Date today.!

iOldfecha
fecha:= Date fromString: (Prompter prompt: 'ingrese fecha de actividad') .!

iPaciente: a
paciente := a.!

iPeso: a
""

peso:= a.!

iProfesional: a
""

profesional:= a.
```

!

oBand

""

^band!

oDesc

^descripcion.!

oFecha

"Devuelve fecha"

^fecha.!

oPaciente

"devuelve el paciente de la actividad"

^paciente.!

oPeso

""

^peso.!

oProfesional

"Devuelve el profesional a cargo de la actividad"

^profesional.! !

!Actividad categoriesFor: #iBand!public! !

!Actividad categoriesFor: #iDesc:!public! !

!Actividad categoriesFor: #iFecha!public! !

!Actividad categoriesFor: #iOldfecha!public! !

!Actividad categoriesFor: #iPaciente:!public! !

!Actividad categoriesFor: #iPeso:!public! !

!Actividad categoriesFor: #iProfesional:!public! !

!Actividad categoriesFor: #oBand!public! !

!Actividad categoriesFor: #oDesc!public! !

!Actividad categoriesFor: #oFecha!public! !

!Actividad categoriesFor: #oPaciente!public! !

!Actividad categoriesFor: #oPeso!public! !

!Actividad categoriesFor: #oProfesional!public! !

Clinica guid: (GUID fromString: '{77c9d781-77ee-4d15-817e-640028dbcef2}')!

Clinica comment: ''!

!Clinica categoriesForClass!Kernel-Objects! !

!Clinica methodsFor!

busca_oSoc: c

""

|ban|

```
ban:= obra_social detect: [:x | x oCod = c ] ifNone: [ban:=nil].
^ban.!

busca_paciente: dni

|ban|
ban:= pacientes detect: [:x | x oDni = dni] ifNone: [ban:=0].
^ban.
!

busca_pr: dni
""

|ban|
ban:= profesionales detect: [:x | x oDni = dni ] ifNone: [ban:=0].
^ban.!

busqueda_paciente
""

| p |
p:= self busca_paciente: (Prompter prompt: 'Ingrese dni del paciente a buscar').
[p=0] whileTrue: [p:= self busca_paciente: (Prompter prompt: '(DNI no
encontrado) Ingrese nuevamente dni del paciente a buscar')].
```

(p isMemberOf: Pcte_ObraSocial) ifTrue: [Transcript clear; show: 'Nombre'; tab; show: 'Apellido'; tab; show: 'Dirección'; tab; show: 'DNI'; tab; show: 'Teléfono'; tab; show: 'Peso'; tab; show: 'Obra social'; cr;

show: p oNombre,' ', p oApellido ,' ', p oDireccion, ' ', p oDni, ' ', p oTelefono, ' ', p oPeso, ' ' , p oObraSocial oNombre ; cr]

ifFalse: [

Transcript clear; show: 'Nombre'; tab; show: 'Apellido'; tab; show: 'Dirección'; tab; show: 'DNI'; tab; show: 'Teléfono'; tab; show: 'Peso'; cr

show: p oNombre,' ', p oApellido ,' ', p oDireccion, ' ', p oDni, ' ', p oTelefono, ' ', p oPeso; cr]!

carga_ac

|op act pcte tipo d pr res|

res:=MessageBox confirm: 'Desea ingresar una actividad?'.

[res] whileTrue: [

act:= nil.

pcte:= nil.

pr:= nil.

op:= (Prompter prompt: '1- Consultas clinicas | 2- Reuniones grupales | 3- Sesion de gimnasio' caption: 'Ingrese actividad') asNumber.

op=1 ifTrue: [act:= Consulta new.].

op=2 ifTrue: [act:=Reunion new.].

op=3 ifTrue: [act:= Gym new.].

act iOldfecha.

d := Prompter prompt: 'Ingrese DNI del paciente'.

[(self busca_paciente: d) =0] whileTrue: [d := Prompter prompt: '(Paciente no encontrado) Ingrese DNI nuevamente'].

```
pcte := self busca_paciente: d.
act iPeso: (Prompter prompt: 'Ingrese el peso del paciente') asNumber asFloat.
act iPaciente: pcte.

tipo:= (Prompter prompt: '1- Medico Clinico. 2- Nutricionista. 3- Psicologo'
caption: 'Ingrese el tipo de profesional') asNumber.
(tipo=1) ifTrue: [pr:= MedicoClinico].
(tipo=2) ifTrue: [pr:= Nutricionista].
(tipo=3) ifTrue: [pr:= Psicologo].
self lista_pr: pr.
pr:= self busca_pr: (Prompter prompt: 'Ingrese el dni del profesional') .
[pr =0] whileTrue: [pr:= self busca_pr: (Prompter prompt: '(DNI no encontrado)
Ingrese el dni del profesional nuevamente').].
act iProfesional: pr.
act iDesc: (Prompter prompt: 'Ingrese descripcion de la actividad').
actividades add: act.
res := (MessageBox confirm: 'Desea ingresar una nueva actividad?')].!

carga_co
        ""

        | co res p act f e |
        act := OrderedCollection new.
        res := MessageBox confirm: 'Desea ingresar un cobro?'.
        [res] whileTrue:
```

```
[p := Paciente new.
e := true.
e := MessageBox confirm: 'Es un cobro con fecha de hoy?'.
e = true
        ifTrue: [f := Date today]
        ifFalse: [f := Date fromString: (Prompter prompt:
'Ingrese fecha')].
p := self busca_paciente: (Prompter prompt: 'Ingrese el dni
del paciente').
[p = 0] whileTrue:
                [p := self
                                busca_paciente:
(Prompter prompt: '(No se encontro paciente) Ingrese nuevamente el dni del
paciente:')].
act := actividades select: [:x | x oPaciente = p & (x oBand =
nil)].
co := nil.
co := Cobro new.
co iImporte: 0.
co ini.
co iFecha: f.
co iPaciente: p.
act do:
                [:i |
                i iBand.
                co iImporte: co oImporte + i calcula_precio.
                co iActividades: i].
```

```
                    cobros add: co.
                    res := MessageBox confirm: 'Desea ingresar un nuevo
cobro?']!

carga_os
|resp os|
resp:=MessageBox confirm: 'Desea ingresar una obra social?'.
[resp] whileTrue:[
os := nil.
os:= Obra_Social new.
os cargaos.
obra_social add: os.
resp := MessageBox confirm: 'Desea ingresar una nueva obra social?'
]!

carga_p
"Carga pacientes"
|resp p op|
resp := MessageBox confirm: 'Desea ingresar un paciente?'.
[resp] whileTrue: [
p:= nil.
op := (Prompter prompt: 'Ingrese tipo de paciente 1- Particular 2- Paciente con
Obra Social') asNumber .
op = 1 ifTrue: [p := Particular new].
op = 2 ifTrue: [p := Pcte_ObraSocial new.
```

```
                    p iObraSocial: (self busca_oSoc: (Prompter prompt:
'ingrese codigo de obra social')).].

(op=1) | (op=2) ifFalse: [MessageBox notify: 'No ha ingresado tipo correcto de
paciente (intente nuevamente)' ]

ifTrue: [ p carga_datos .

pacientes add: p.

resp := MessageBox confirm: 'Desea ingresar otro paciente?'

]].

!


carga_pr

".."

|resp pr t|

resp := MessageBox confirm: 'Desea ingresar un profesional?'.

[resp] whileTrue: [


pr := nil.

t :=(Prompter prompt: 'Ingrese tipo de profesional 1-Medico Clinico 2-
Nutricionista 3-Psicologo') asNumber.

t=1 ifTrue: [pr :=MedicoClinico new].

t=2 ifTrue: [pr :=Nutricionista new].

t=3 ifTrue: [pr :=Psicologo new].

(t=1) | (t=2) | (t=3) ifFalse: [MessageBox notify: 'Tipo de profesional incorrecto' ]

ifTrue:[ pr carga_datos .

profesionales add: pr.

resp:=MessageBox confirm: 'Desea ingresar un nuevo profesional?'.
```

].

]!

inicializa

"Crea colecciones"

actividades:= OrderedCollection new.

pacientes:= OrderedCollection new.

profesionales:= OrderedCollection new.

cobros:= OrderedCollection new.

obra_social:= OrderedCollection new.

Gym iCosto_sesion: (Prompter prompt: 'Ingrese el costo de una sesion de gym' caption: 'Por favor') asNumber.

Reunion iCostoReunion: (Prompter prompt: 'Ingrese el costo de una reunion' caption: 'Por favor') asNumber.

Paciente iProxNum: 1.

self carga_os.

self carga_p.

self carga_pr.

self carga_ac.

self carga_co.!

```
lista_pr: a
""
|m|
m := OrderedCollection new.
m:= profesionales select: [:i |  i class = a].


Transcript clear; show:'Listado'; cr.
Transcript show:'Nombre','  ','Apellido','  ','Tarifa consulta','  ','Tarifa grupo'; cr.
m do:[ :i|  Transcript show: i oNombre,'  ',i oApellido,'  ',  i otarifac printString ,'  ',
i otarifag printString ; cr. ].
!

listado
        ""

        | f_inicio f_fin cod cobro_obraSoc obra imp_total resp |
        imp_total := 0.
        cobro_obraSoc := OrderedCollection new.
        resp := false.
        resp
                ifFalse:
                        [cod := Prompter prompt: 'Ingrese el codigo de la obra
social'.
```

```
                resp := obra_social detect: [:x | x oCod = cod]].
        f_inicio := Date fromString: (Prompter prompt: 'Ingrese fecha de inicio de
cobros').
        f_fin := Date fromString: (Prompter prompt: 'Ingrese fecha de fin de
cobros').
        cobro_obraSoc := cobros select:
                                [:a |
                                a oPaciente oObraSocial oCod = cod & ((a
oFecha subtractDate: f_inicio) >= 0)
                                        & ((a oFecha subtractDate: f_fin) <= 0)].
        cobro_obraSoc := cobro_obraSoc asSortedCollection: [:a :b | (a oFecha
subtractDate: b oFecha) <= 0].
        obra := self busca_oSoc: cod.
        Transcript
                clear;
                show: 'Nombre';
                tab;
                show: 'Porcentaje de cobertura';
                cr.
        Transcript
                show: obra oNombre;
                tab;
                show: obra oPorcentaje printString , '%';
                cr;
                cr.
        Transcript
```

```
            show: 'Cobros';
            cr;
            cr;
            show: 'Fecha';
            tab; tab;
            show: 'Nombre';
            tab; tab; tab;
            show: 'Apellido';
            tab; tab; tab;
            show: 'Actividad';
            tab; tab; tab;
            show: 'Importe';
            cr.
        cobro_obraSoc isEmpty
            ifTrue: [MessageBox notify: 'Listado de cobros vacio']
            ifFalse:
                [cobro_obraSoc do:
                    [:co | (co oActividades asSortedCollection: [:a
:b | (a oFecha subtractDate: b oFecha) <= 0])  do: [:act |
                    Transcript
                        show: act oFecha shortString;
tab;show: act oPaciente oNombre; tab; show: act oPaciente oApellido; tab;
show: act oDesc; tab; show: (act calcula_precio * obra oPorcentaje / 100)
printString; cr.
                    imp_total := (act calcula_precio * obra
oPorcentaje / 100) + imp_total]].
                Transcript cr; show: 'Importe total : ' , imp_total printString].!
```

```
menu
	"Displaya menu y direcciona"

	| direc |
	direc := 5.
	self inicializa.
	[direc ~= 0] whileTrue:
			[direc := (Prompter prompt: '1- Registro minuta | 2- Listado cobros | 3- Cobrar | 4- Busca paciente | 0- Salir' caption: 'Menu') asNumber.
			direc = 1 ifTrue: [self registro_minuta].
			direc = 2 ifTrue: [self listado].
			direc = 3 ifTrue: [self carga_co].
			direc = 4 ifTrue: [self busqueda_paciente]]!

registro_minuta
	| op act pcte tipo d pr res |
	res := true.
	[res] whileTrue:
			[act := nil.
			pcte := nil.
			pr := nil.
			op := (Prompter prompt: '1- Consultas clinicas | 2- Reuniones grupales | 3- Sesion de gimnasio'
					caption: 'Ingrese una opcion') asNumber.
```

```
op = 1 ifTrue: [act := Consulta new].
op = 2 ifTrue: [act := Reunion new].
op = 3 ifTrue: [act := Gym new].
act iFecha.
d := Prompter prompt: 'Ingrese DNI del paciente'.
[(self busca_paciente: d) = 0]
    whileTrue: [d := Prompter prompt: '(Paciente no encontrado) Ingrese DNI nuevamente'].
pcte := self busca_paciente: d.
act iPeso: (Prompter prompt: 'Ingrese el peso del pacinte') asNumber asFloat.
act iPaciente: pcte.
op = 3
    ifTrue: [pr := nil]
    ifFalse:
        [tipo := (Prompter prompt: '1- Medico Clinico. 2- Nutricionista. 3- Psicologo'
                caption: 'Ingrese el tipo de profesional') asNumber.
        tipo = 1 ifTrue: [pr := MedicoClinico].
        tipo = 2 ifTrue: [pr := Nutricionista].
        tipo = 3 ifTrue: [pr := Psicologo].
        self lista_pr: pr.
        pr := self busca_pr: (Prompter prompt: 'Ingrese el dni del profesional').
        [pr = 0] whileTrue: [pr := self busca_pr: (Prompter prompt: 'Ingrese el dni del profesional')]].
```

act iProfesional: pr.

act iDesc: (Prompter prompt: 'Ingrese descripcion de la actividad').

actividades add: act.

res := MessageBox confirm: 'Desea ingresar una nueva minuta?']! !

!Clinica categoriesFor: #busca_oSoc:!public! !

!Clinica categoriesFor: #busca_paciente:!public! !

!Clinica categoriesFor: #busca_pr:!public! !

!Clinica categoriesFor: #busqueda_paciente!public! !

!Clinica categoriesFor: #carga_ac!public! !

!Clinica categoriesFor: #carga_co!public! !

!Clinica categoriesFor: #carga_os!public! !

!Clinica categoriesFor: #carga_p!public! !

!Clinica categoriesFor: #carga_pr!public! !

!Clinica categoriesFor: #inicializa!public! !

!Clinica categoriesFor: #lista_pr:!public! !

!Clinica categoriesFor: #listado!public! !

!Clinica categoriesFor: #menu!public! !

!Clinica categoriesFor: #registro_minuta!public! !

Cobro guid: (GUID fromString: '{de70ca32-5374-408f-b419-943bc4bff403}')!

Cobro comment: ''!

!Cobro categoriesForClass!Kernel-Objects! !

!Cobro methodsFor!

```
iActividades: a
""

actividades add: a.
!

iFecha: a
""
fecha:= a.!

iImporte: a
""
importe:= a .!

ini
""

actividades:= OrderedCollection new.!

iPaciente: a
""
paciente:= a.!

oActividades
```

""

^actividades!

oFecha
""
^fecha.!

oImporte
""

^importe.!

oPaciente
""
^paciente.!

oTipo
""

^(paciente isMemberOf: Pcte_ObraSocial).! !
!Cobro categoriesFor: #iActividades:!public! !
!Cobro categoriesFor: #iFecha:!public! !
!Cobro categoriesFor: #iImporte:!public! !
!Cobro categoriesFor: #ini!public! !

!Cobro categoriesFor: #iPaciente:!public! !

!Cobro categoriesFor: #oActividades!public! !

!Cobro categoriesFor: #oFecha!public! !

!Cobro categoriesFor: #oImporte!public! !

!Cobro categoriesFor: #oPaciente!public! !

!Cobro categoriesFor: #oTipo!public! !

Obra_Social guid: (GUID fromString: '{10e5da0c-2524-464e-a5d1-9f83d40919f6}')!

Obra_Social comment: ''!

!Obra_Social categoriesForClass!Kernel-Objects! !

!Obra_Social methodsFor!

cargaos

nombre:= Prompter prompt: 'Ingrese nombre de la obra social'.

codigo:= (Prompter prompt: 'Ingrese el codigo') .

porcentaje_cobertura:= (Prompter prompt: 'Ingrese % de cobertura') asNumber.!

oCod

""

^codigo.!

oNombre

""

^nombre.!

oPorcentaje
""
^porcentaje_cobertura.! !
!Obra_Social categoriesFor: #cargaos!public! !
!Obra_Social categoriesFor: #oCod!public! !
!Obra_Social categoriesFor: #oNombre!public! !
!Obra_Social categoriesFor: #oPorcentaje!public! !

Persona guid: (GUID fromString: '{d09852f7-37ac-420b-b675-3a950ccd4d22}')!
Persona comment: ''!
!Persona categoriesForClass!Unclassified! !
!Persona methodsFor!

carga_datos
	".."

	self iNombre: (Prompter prompt: 'Ingrese nombre').
	self iApellido: (Prompter prompt: 'Ingrese apellido').
	self iTelefono: (Prompter prompt: 'Ingrese telefono').
	self iDireccion: (Prompter prompt: 'Ingrese direccion').
	self iDni: (Prompter prompt: 'Ingrese dni')!

iApellido: a

""

apellido:= a.!

iDireccion: a

""

direccion:= a.!

iDni: a

""

dni:= a.!

iNombre: a

""

nombre:= a.!

iTelefono: a

""

telefono:= a.!

oApellido

""

^apellido.!

oDireccion

""

^direccion.!

oDni

""

^dni.!

oNombre

""

^nombre.!

oTelefono

""

^telefono.! !

!Persona categoriesFor: #carga_datos!public! !

!Persona categoriesFor: #iApellido:!public! !

!Persona categoriesFor: #iDireccion:!public! !

!Persona categoriesFor: #iDni:!public! !

!Persona categoriesFor: #iNombre:!public! !

!Persona categoriesFor: #iTelefono:!public! !

!Persona categoriesFor: #oApellido!public! !

!Persona categoriesFor: #oDireccion!public! !

!Persona categoriesFor: #oDni!public! !

!Persona categoriesFor: #oNombre!public! !

!Persona categoriesFor: #oTelefono!public! !

Consulta guid: (GUID fromString: '{29508332-c79d-4d90-95a5-49ee45ec95be}')!

Consulta comment: "!

!Consulta categoriesForClass!Kernel-Objects! !

!Consulta methodsFor!

calcula_precio

"Calcula precio"

|tarifa|

tarifa:= super oProfesional otarifac .

^tarifa! !

!Consulta categoriesFor: #calcula_precio!public! !

Gym guid: (GUID fromString: '{ace7f242-098e-4fa6-8c65-c47d8322a02f}')!

Gym comment: "!

!Gym categoriesForClass!Kernel-Objects! !

!Gym methodsFor!

calcula_precio

"Calcula precio"

^Costo_sesion.! !

```
!Gym categoriesFor: #calcula_precio!public! !

!Gym class methodsFor!

iCosto_sesion: a

Costo_sesion:= a.! !
!Gym class categoriesFor: #iCosto_sesion:!public! !

Reunion guid: (GUID fromString: '{052fc3b2-8a05-47c6-b8f5-a89c09c0c1c1}')!
Reunion comment: ''!
!Reunion categoriesForClass!Kernel-Objects! !
!Reunion methodsFor!

calcula_precio
"Calcula precio"

|tarifa|
tarifa:= super oProfesional otarifag .
^Costo_reunion + tarifa.! !
!Reunion categoriesFor: #calcula_precio!public! !

!Reunion class methodsFor!

iCostoReunion: a
```

"Ingresa el costo de una reunion"

Costo_reunion:= a.! !

!Reunion class categoriesFor: #iCostoReunion:!public! !

Paciente guid: (GUID fromString: '{e5436e27-ec5d-4db2-b4fc-027db2586bb3}')!

Paciente comment: ''!

!Paciente categoriesForClass!Unclassified! !

!Paciente methodsFor!

carga_datos

	".."

	self iNombre: (Prompter prompt: 'Ingrese nombre').

	self iApellido: (Prompter prompt: 'Ingrese apellido').

	self iTelefono: (Prompter prompt: 'Ingrese telefono').

	self iDireccion: (Prompter prompt: 'Ingrese direccion').

	self iDni: (Prompter prompt: 'Ingrese dni').

	cod:=Prox_Num.

	Paciente iProxNum: Prox_Num + 1.

	peso := Prompter prompt: 'Ingrese peso'.

	!

iObraSocial: a

""

```
obraSocial:= a.!

iPeso: a
peso :=a.!

oObraSocial
""

^obraSocial.!

oPeso

^peso! !
!Paciente categoriesFor: #carga_datos!public! !
!Paciente categoriesFor: #iObraSocial:!public! !
!Paciente categoriesFor: #iPeso:!public! !
!Paciente categoriesFor: #oObraSocial!public! !
!Paciente categoriesFor: #oPeso!public! !

!Paciente class methodsFor!

iProxNum: a
"Inicializa numero de codigo"
Prox_Num:=a.! !
```

```
!Paciente class categoriesFor: #iProxNum:!public! !

Profesional guid: (GUID fromString: '{ded84e79-1e5b-4c40-9dd1-
3b6c6c135e86}')!

Profesional comment: ''!

!Profesional categoriesForClass!Unclassified! !

!Profesional methodsFor!

carga_datos
	"."

	self iNombre: (Prompter prompt: 'Ingrese nombre').

	self iApellido: (Prompter prompt: 'Ingrese apellido').

	self iTelefono: (Prompter prompt: 'Ingrese telefono').

	self iDireccion: (Prompter prompt: 'Ingrese direccion').

	self iDni: (Prompter prompt: 'Ingrese dni').



	matricula := Prompter prompt: 'Ingrese matricula'.

	tarifa_consulta:= (Prompter prompt: 'Ingrese tarifa de consulta')
asNumber .

	tarifa_grupo := (Prompter prompt: 'Ingrese tarifa de grupo') asNumber .!

omatri

^(matricula)!
```

otarifac

^(tarifa_consulta )!

otarifag

^(tarifa_grupo)! !

!Profesional categoriesFor: #carga_datos!public! !

!Profesional categoriesFor: #omatri!public! !

!Profesional categoriesFor: #otarifac!public! !

!Profesional categoriesFor: #otarifag!public! !

Particular guid: (GUID fromString: '{28cfc633-bda9-458b-b069-ff6f99023d3f}')!

Particular comment: ''!

!Particular categoriesForClass!Unclassified! !

Pcte_ObraSocial guid: (GUID fromString: '{8ac98c16-129c-4740-8fe5-5827af32a29f}')!

Pcte_ObraSocial comment: ''!

!Pcte_ObraSocial categoriesForClass!Unclassified! !

MedicoClinico guid: (GUID fromString: '{fce960c0-aef3-4b18-8c02-6c60624a9e2e}')!

MedicoClinico comment: ''!

!MedicoClinico categoriesForClass!Unclassified! !

Nutricionista guid: (GUID fromString: '{8114c3f8-6bb0-41ec-9665-8ad53221af20}')!

Nutricionista comment: ''!

!Nutricionista categoriesForClass!Unclassified! !

Psicologo guid: (GUID fromString: '{3a889b63-a6eb-4ef9-bb09-28888b786f5e}')!

Psicologo comment: ''!

!Psicologo categoriesForClass!Unclassified! !

"Binary Globals"!

## Respuestas al Cuestionario

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

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

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

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

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

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

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.

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

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

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

Desarrollo

a) Si, usamos clases abstractas para cumplir el objetivo del programa. Son abstractas porque no producen instancias, pero poseen variables y métodos que pueden utilizar todas las instancias que se crean en las subclases de dicha clase.
Ejemplo: utilizamos como clase abstracta la clase Actividad, la cual tiene como variables fecha, paciente, profesional, desc y peso, también tiene distintos métodos para realizar distintas operaciones en ella, pero esta clase no tiene instancias propias de la clase, tiene las instancias de las subclases derivadas de esta (Consulta, Reunion y Gym), quienes utilizan sus métodos y variables.

b) Si, existen clases concretas. Son concretas porque tienen métodos y variables que se utilizan para definir y manipular las instancias de las mismas.

   Ejemplo: la clase concreta Cobro tiene variables y métodos para cargar y mostrar fecha, actividad, paciente e importe al realizar un cobro, creando por cada cobro una nueva instancia de la misma con los datos mencionados anteriormente.

   La clase concreta Obrasocial tiene variables y métodos para cargar y mostrar datos de una obra social, al cargar datos de una nueva obra social se está creando una nueva instancia de la clase.

c) Las relaciones de herencia que se pueden reconocer en el programa son relaciones de herencia múltiple.

   Ejemplo: la clase paciente tiene dos subclases: Particular y Paciente_Obrasocial las cuales heredan las variables y métodos de la clase padre

   La clase persona tiene dos subclases que heredan de ella las variables y los métodos, estas son: Paciente y Profesional.

   Las relaciones de herencia son subclases de una clase que heredan las variables y atributos de esta, esto no quita que las subclases puedan tener aparte sus propias variables y métodos.

d) Las relaciones de ensamble que se pueden reconocer son: Actividad-Profesional, Actividad-Paciente, Actividad-Cobro, Actividad-Clinica, Actividad-Persona, Clinica-

Obrasocial, Clinica-Cobro, Clinica-Paciente, Clinica-Profesional, Obrasocial-Paciente, Cobro-Paciente.

Ejemplo: la clase Actividad tiene una relación de ensamble con la clase Clinica, ya que una actividad es parte de la clinica.

La clase clinica tiene una relación de ensamble con Obrasocial, ya que la clinica puede tener varias instancias de Obrasocial, una Obrasocial es parte de la clinica.

e) Si, existen. Al método calcula_precio se define con el mismo nombre, pero depende la clase que lo utiliza (Gym, Consulta o Reunión) es el procedimiento que realiza.

El polimorfismo es que más de un objeto responden a un determinado mensaje, llamado de la misma manera, pero cada uno con su forma, la cual no necesariamente es la misma.

f) Si, existen.
Ejemplo: el método carga_datos de Persona está redefinido en Paciente y Profesional. Redefinición es cuando un método cambia de acuerdo a la clase del objeto receptor del mensaje. En el caso de carga_datos, dependiendo si la clase es una u otra va a cargar datos distintos.

g) Clase Actividad:

Subclase Gym: VC: Costo_sesion

Subclase Reunion: VC: Costo_Reunion

Ejemplos: Si tenemos la variable de clase Precio_Dolar cargada con el precio actual del dólar, en el programa de un negocio, y a esta se la utiliza para calcular el precio final de un producto a través de alguna fórmula (ej: precio del producto * Precio_Dolar) podremos tener todos los productos en función del dólar y el precio del dólar será el mismo para todos.

Las variables de clase son variables que son propias de una clase y permite que cualquier instancia de esta pueda usar su valor (siendo el mismo para cualquier instancia). En el presente trabajo las tuvimos que utilizar para cargar los costos generales de las sesiones de gimnasio y de las reuniones, ya que estas tenían un costo inicial y a ese costo se le sumaba lo que cobrara el profesional que las llevaba a cabo, si lo había.

h) Todas las variables de instancia utilizadas se detallan en el diagrama (en el segundo bloque de cada clase), algunas de ellas son: de la clase Obrasocial: nombre, código, porcentaje.

   Las variables de instancia pueden ser accedidas por las instancias de una clase, y cada una de ellas puede tener su propio valor en dicha variable a diferencia de las variables de clase que todas las instancias comparten el mismo valor de la variable.

i) A los métodos de clase los podemos encontrar dentro del diagrama, en el tercer casillero de cada clase. Algunos de los que utilizamos son: en la subclase Gym de la clase Actividad: iCosto_Gym; en la subclase Reunion de la clase Actividad: iCosto_reunion.

   Estos métodos son necesarios para el ingreso de los valores a las variables de clase.

j) Ejemplos de mensajes unarios:
   - Clase Clinica, método listado: cobro_obraSoc := OrderedCollection **new**.
   - Clase Clinica, método menú: direc := (Prompter prompt: '1- Registro minuta | 2- Listado | 3- Cobrar | 0- Salir' caption: 'Menu') **asNumber**.

Ejemplos de mensajes binarios:
   - Clase Clinica, método menú: [direc ~= 0]
   - Clase Clinica, método carga: (op = 1) | (op = 2) ifFalse: [MessageBox notify: 'No ha ingresado tipo correcto de paciente (intente nuevamente)' ]

Ejemplos de mensajes de palabra clave:
   - Clase Clinica, método listado: f_inicio := Date **fromString:** (Prompter prompt: 'Ingrese fecha de inicio de cobros').
   - Clase Clinica, método listado: resp := obra_social **detect:** [:x | x oCod = cod]].

.