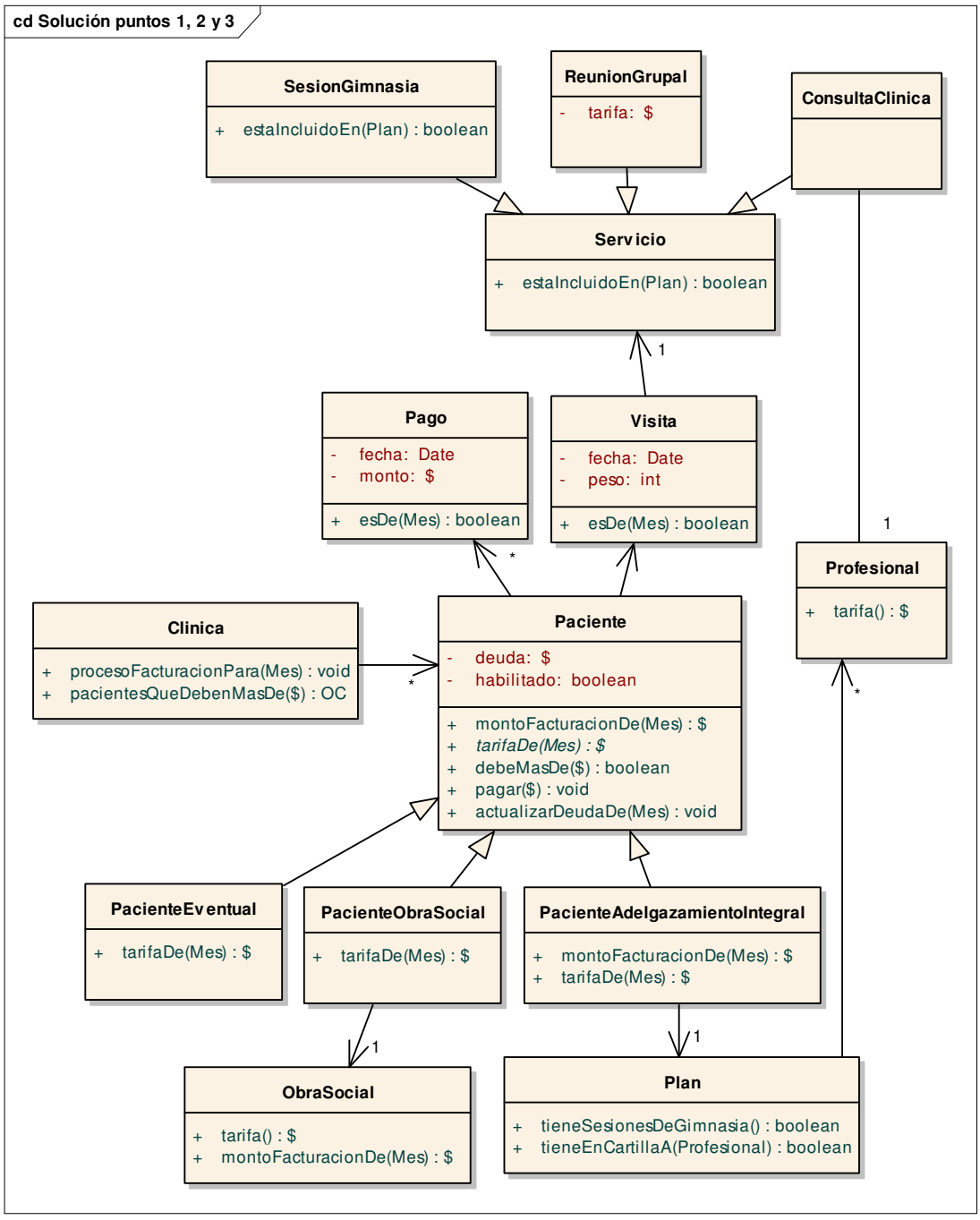
Una solución posible

Diagrama de clases



No están todos los métodos (no es necesario que estén todos, claro, sólo los más representativos).

Código

Punto 1)

montoFacturacionDe: unMes (#Paciente)
^(self serviciosDe: unMes) inject: 0 into: [:acum :servicio | acum + (self tarifaDe: servicio)]

serviciosDe: unMes (#Paciente)
^(self visitasDe: unMes) collect: [:visita | visita servicio]

visitasDe: unMes (#Paciente)
^visitas select: [:visita | visita esDe: unMes]

esDe: unMes (#Visita)
^fecha month = unMes (y listo…)

podríamos crear un objeto que represente un Periodo (mes/año) y delegarle la respuesta:
esDe: unMes (#Visita)
^unMes estaDentroDe: fecha
Pero no es obligatorio…

tarifaDe: unServicio (#PacienteEventual)
^unServicio tarifa

tarifaDe: unServicio (#PacienteObraSocial)
^unServicio tarifa * obraSocial porcentajeAsociado

porcentajeAsociado (#ObraSocial)
^100 - porcentajeCobertura

tarifaDe: unServicio (#PacienteAdelgazamientoIntegral)
^(unServicio estaIncluidoEn: plan)
ifTrue: [0]
ifFalse: [unServicio tarifa]

montoFacturacionDe: unMes (#PacienteAdelgazamientoIntegral)
^(super montoFacturacionDe: unMes) + plan tarifa

estaIncluidoEn: unPlan (#Servicio)
^unPlan tieneEnCartillaA: self profesional

tieneEnCartillaA: unProfesional (#Plan)
^profesionales includes: unProfesional

estaIncluidoEn: unPlan (#SesionGimnasia)
^unPlan tieneSesionesDeGimnasia (es un getter)

tarifa (#ConsultaClinica)
^profesional tarifa

tarifa (#ReunionGrupal)
^ tarifa

tarifa (#SesionGimnasia)
^10

Punto 2)

montoFacturacionDe: unMes (#ObraSocial)
^self pacientes
inject: 0
into: [:acum :paciente | acum + (paciente montoFacturacionObraSocialDe: unMes)]

montoFacturacionObraSocialDe: unMes (#PacienteObraSocial)
^(self serviciosDe: unMes) inject: 0 into: [:acum :servicio | acum + (self tarifaDe: servicio)]

tarifaDe: unServicio (#PacienteObraSocial)
^unServicio tarifa * obraSocial porcentajeCobertura (getter)

Punto 3)

3a)
procesoMensualDeFacturacionPara: unMes (#Clinica)
pacientes do: [:paciente | paciente actualizarDeudaDe: unMes]

actualizarDeudaDe: esteMes (#Paciente)
(self hayQueInhabilitar: esteMes)
ifTrue: [self inhabilitar]
deuda := deuda + (self montoFacturacionDe: Mes).

hayQueInhabilitar: esteMes
^(self tienePagosEn: esteMes) not & self esMoroso

esMoroso (#Paciente)
^deuda > 0

tienePagosEn: unMes (#Paciente)
^pagos anySatisfy: [:pago | pago esDe: unMes]

esDe: unMes (#Pago)
^fecha month = unMes (lo mismo que en el punto 1)

inhabilitar (#Paciente)
habilitado := false.

3b)
pagar: unMonto (#Paciente)
deuda := deuda – unMonto.
pagos add: (Pago new fecha: Date today; monto: unMonto)

3c)
pacientesQueDebenMasDe: unMonto (#Clinica)
^pacientes select: [:paciente | paciente debeMasDe: unMonto]

debeMasDe: unMonto (#Paciente)
^deuda > unMonto

3d)

```
inhabilitarPacientesQueDebenMasDe: unMonto (#Clinica)
    (self pacientesQueDebenMasDe: unMonto) do: [ :paciente | paciente inhabilitar ]
```

Punto 4) BONUS

Este punto es bastante intrincado, requiere subclasificar Profesional en Interno y Externo (hay otras opciones). En el caso del profesional interno, simplemente devolvemos el sueldo fijo que tiene asignado.

```
cuantoCobra: esteMes (#ProfesionalInterno)
   ^sueldo
```

El profesional externo conoce el conjunto de visitas en las que participó.

```
cuantoCobra: esteMes (#ProfesionalExterno)
   ^(self serviciosDe: esteMes) inject: 0
                                    into: [:acum :servicio | acum + (servicio tarifaProfesionalDe: esteMes) ]

serviciosDe: esteMes (#ProfesionalExterno)
    ^(self visitasDe: esteMes) collect: [ :visita | visita servicio ]

visitasDe: esteMes (#ProfesionalExterno)
   ^visitas select: [ :visita | visita esDe: esteMes ]
```

Entonces buscamos cuál es la tarifa del profesional de cada servicio en un mes determinado. En la Consulta es fácil: 60% de la tarifa del profesional (por cada consulta → cada visita representa una consulta en un momento diferente del mes)

```
tarifaProfesionalDe: esteMes (#Consulta)
    ^profesional tarifa * 0.6
```

En el caso de la reunión grupal, nosotros sabemos las visitas que tuvo: no necesitamos saber la cantidad de asistentes, porque cada visita a una reunión grupal de ese mes es un objeto visita de la colección que se obtiene en el método visitasDe: esteMes. Entonces simplemente devolvemos el 5% de la tarifa por cada asistente a una reunión grupal:

```
tarifaProfesionalDe: esteMes (#ReunionGrupal)
    ^profesional tarifa * 0.05
```

Por último un profesional no debería tener sesiones de gimnasia en su colección de visitas:

```
tarifaProfesional (#SesionGimnasia)
    ^0
```

Punto 5)

5a)

```
pacientesQueMasAdelgazaron: esteMes (#Clinica)
    ^(pacientes asSortedCollection:
        [ :a :b | (a cuantoAdelgazo: esteMes) > (b cuantoAdelgazo: esteMes) ])
        copyFrom: 1 to: 10
```

cuantoAdelgazo: esteMes (#Paciente)
```
| visitas |
visitas := self visitasDe: esteMes. ← resuelta en punto 1
^visitas last peso – visitas first peso
```

5b)

promedioDePesoDescendidoDe: unMes (#Clinica)
```
| pacientesRegulares |
pacientesRegulares := self pacientesRegularesDe: unMes.
^ (pacientesRegulares inject: 0 into: [ :acum :paciente | acum + (paciente cuantoAdelgazo: unMes) ])
  / pacientesRegulares size
```

pacientesRegularesDe: unMes (#Clinica)
```
^pacientes select: [ :paciente | paciente esRegularEn: unMes ]
```

esRegularEn: unMes (#Paciente)
```
^(self visitasDe: unMes) size >= 4
```

5c)

La Clínica necesita ahora conocer a las reuniones grupales. La Reunión Grupal concoce a sus pacientes…

grupoDeMayorDescensoDe: unMes (#Clinica)
```
^(grupos asSortedCollection: [ :a :b | (a cuantoDescendio: unMes) > (b cuantoDescendio: unMes) ])
        first
```

cuantoDescendio: unMes (#ReunionGrupal)
```
^pacientes inject: 0 into: [ :acum :paciente | acum + (paciente cuantoAdelgazo: unMes) ]
```