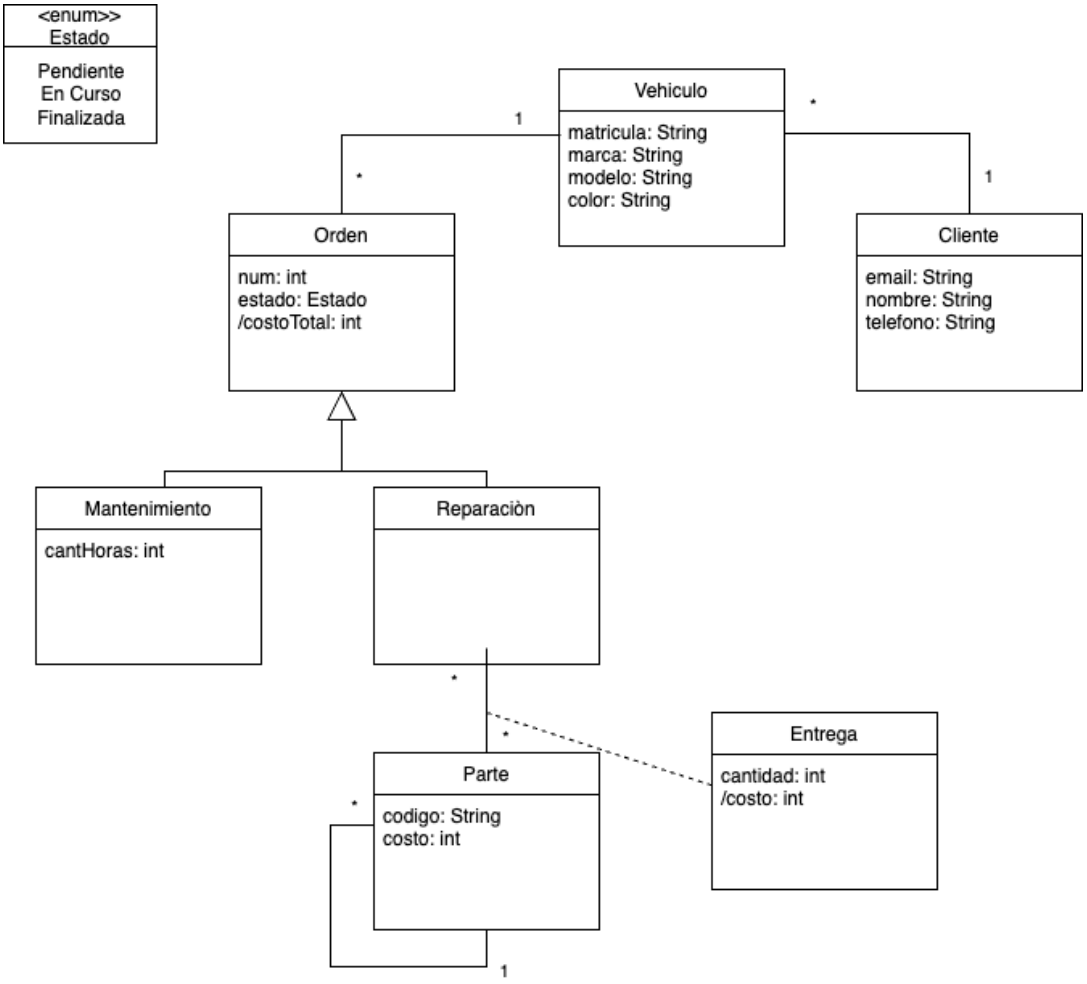
Programación 4

SOLUCIÓN EXAMEN FEBRERO 2024

Problema 1 (30 puntos)

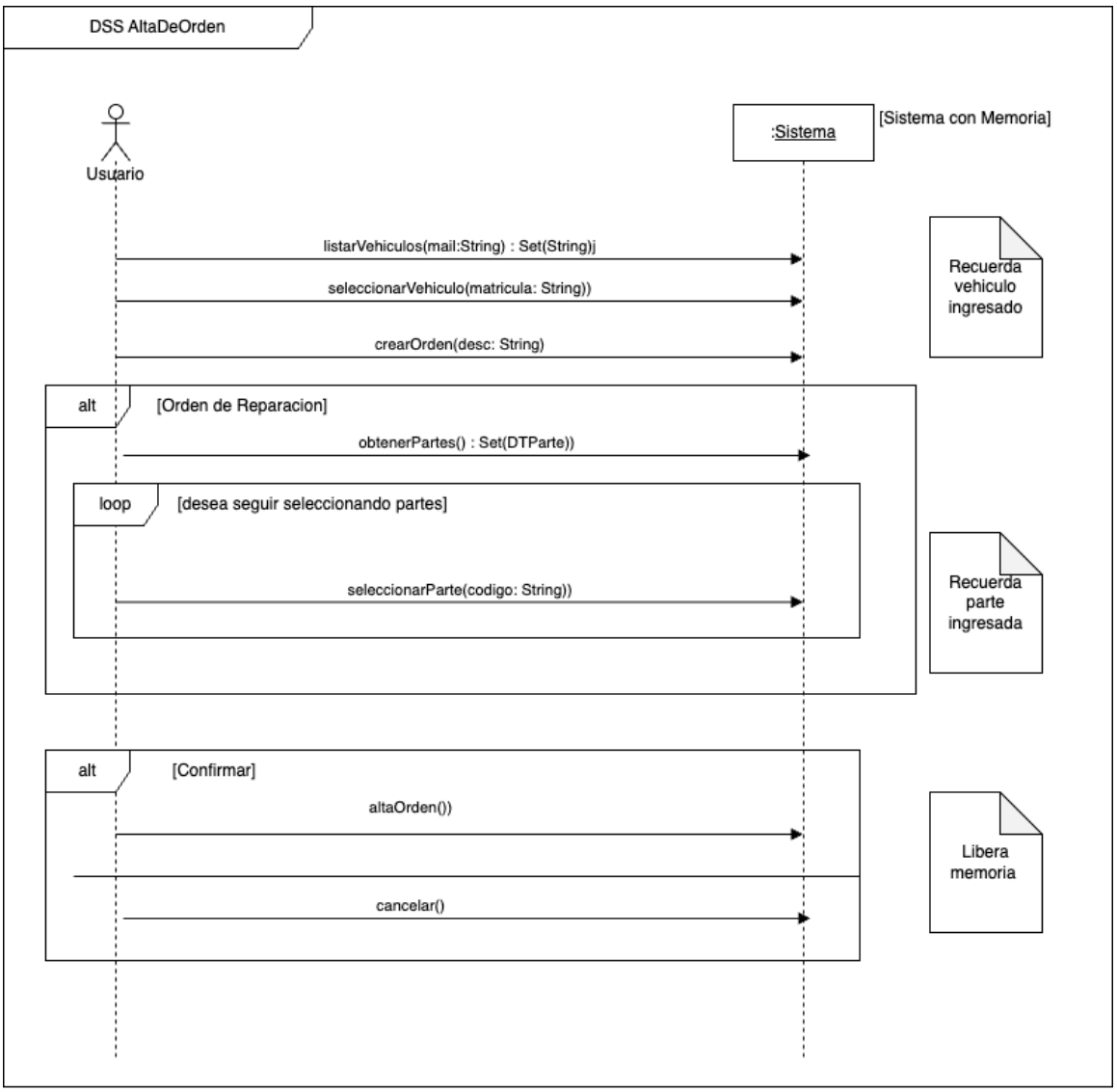
a)

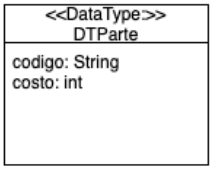


Restricciones:

- Cliente con email único.
- Vehículo con matrícula única.
- Orden con num único.
- Parte con código único.
- Una parte no puede tenerse a sí misma como componente.
- El costo de la orden se calcula en base a su tipo.

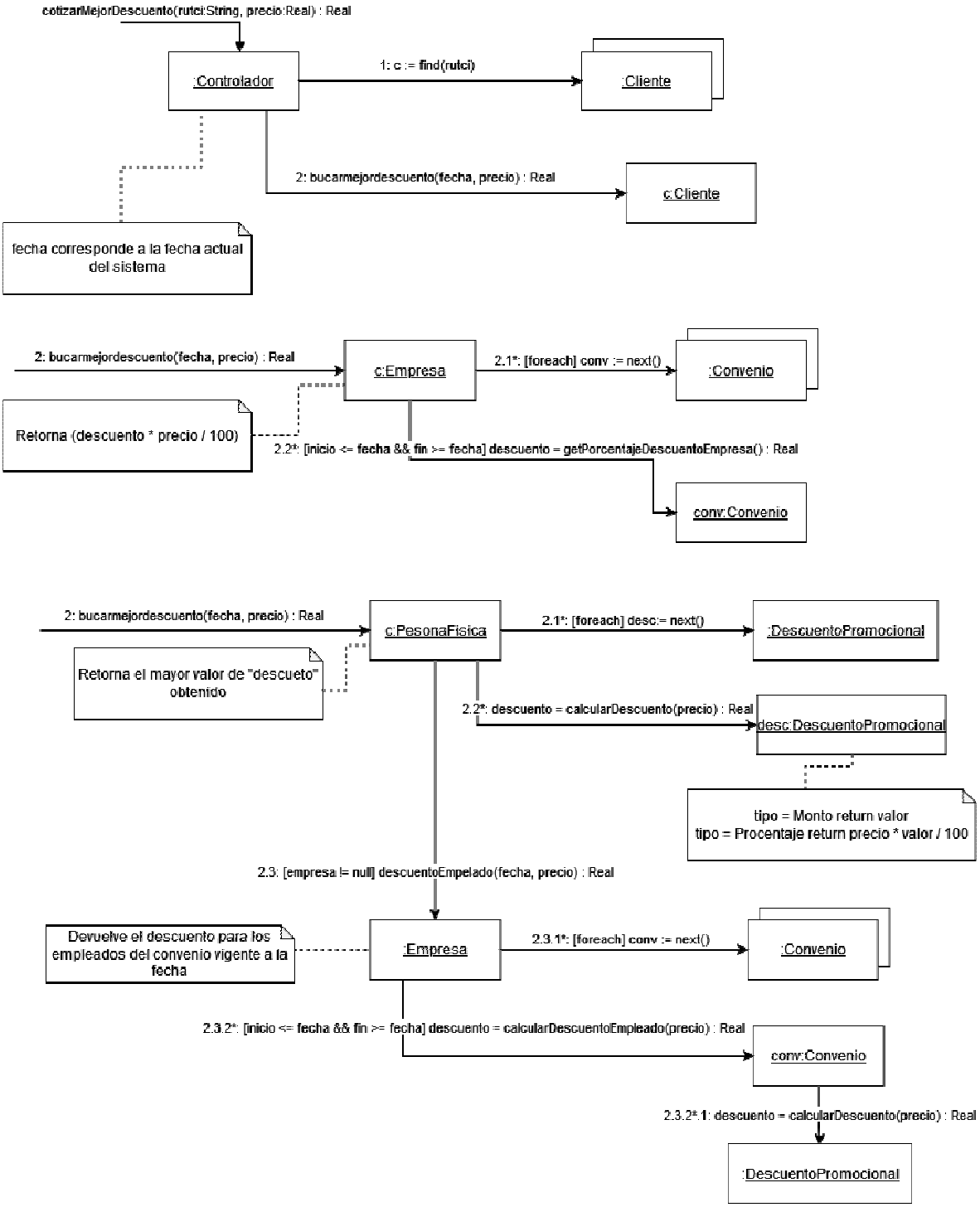
b)

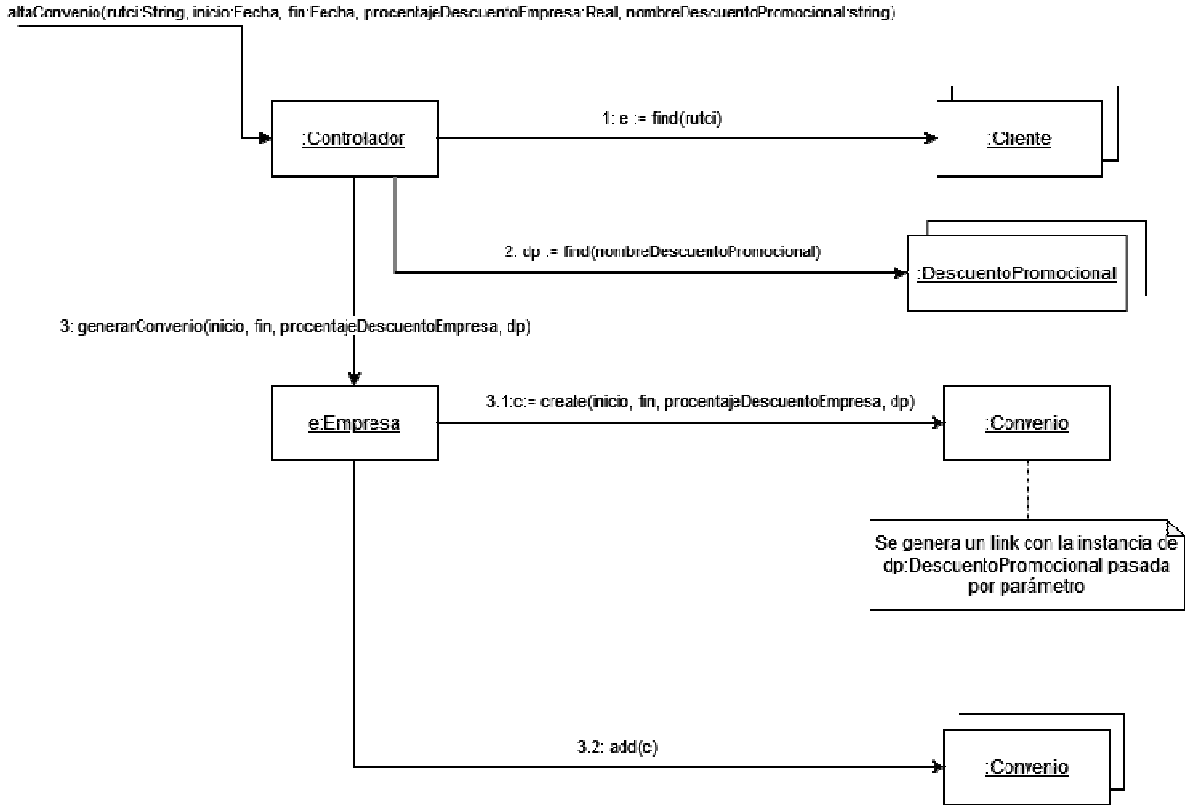

DSS AltaDeOrden
Usuario
:Sistema
[Sistema con Memoria]
listarVehiculos(mail:String) : Set(String)j
seleccionarVehiculo(matricula: String))
crearOrden(desc: String)
Recuerda vehiculo ingresado
alt
[Orden de Reparacion]
obtenerPartes() : Set(DTParte))
loop
[desea seguir seleccionando partes]
seleccionarParte(codigo: String))
Recuerda parte ingresada
alt
[Confirmar]
altaOrden())
cancelar()
Libera memoria


<<DataType>>
DTParte
codigo: String
costo: int

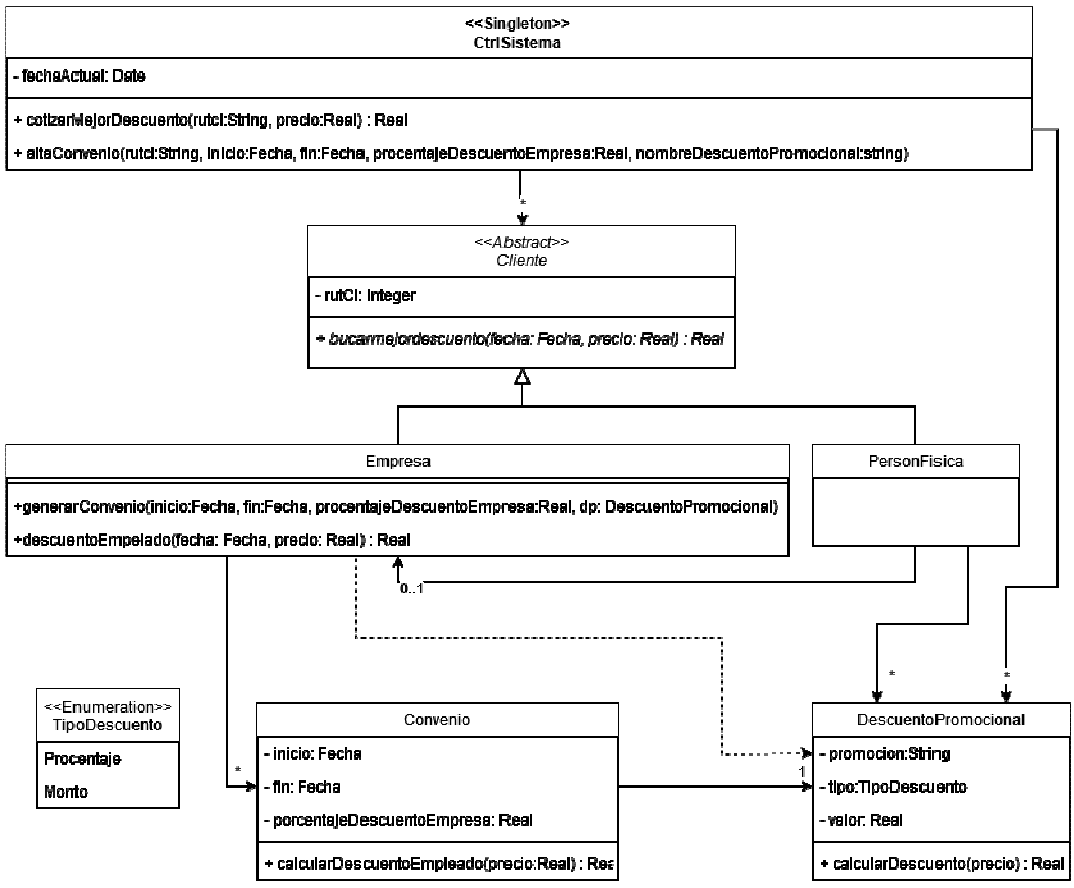
Problema 2 (35 puntos)

a)




altaConvenio(rutcl:String, inicio:Fecha, fin:Fecha, procentajeDescuentoEmpresa:Real, nombreDescuentoPromocional:string)
:Controlador
1: c := find(rutci)
:Cliente
2. dp := find(nombreDescuentoPromocional)
:DescuentoPromocional
3: generarConvenio(inicio, fin, procentajeDescuentoEmpresa, dp)
e:Empresa
3.1:c:= create(inicio, fin, procentajeDescuentoEmpresa, dp)
:Convenio
Se genera un link con la instancia de dp:DescuentoPromocional pasada por parámetro
3.2: add(c)
:Convenio

b)


<<Singleton>>
CtrlSistema
- fechaActual: Date
+ cotizarMejorDescuento(rutcl:String, precio:Real) : Real
+ altaConvenio(rutcl:String, inicio:Fecha, fin:Fecha, procentajeDescuentoEmpresa:Real, nombreDescuentoPromocional:string)
<<Abstract>>
Cliente
- rutCl: Integer
+ bucarmejordescuento(fecha: Fecha, precio: Real) : Real
Empresa
+generarConvenio(inicio:Fecha, fin:Fecha, procentajeDescuentoEmpresa:Real, dp: DescuentoPromocional)
+descuentoEmpelado(fecha: Fecha, precio: Real) : Real
PersonFisica
0..1
<<Enumeration>>
TipoDescuento
Procentaje
Monto
Convenio
- inicio: Fecha
- fin: Fecha
- porcentajeDescuentoEmpresa: Real
+ calcularDescuentoEmpleado(precio:Real) : Rea
DescuentoPromocional
- promocion:String
- tipo:TipoDescuento
- valor: Real
+ calcularDescuento(precio) : Real

Problema 3 (35 puntos)

Corrección de letra: en el diagrama y en la descripción de las operaciones, donde dice Estado debería decir EstadoViaje.

a)

Se utiliza el patrón state con los siguientes roles:

- Viaje es el Contexto.
- EstadoViaje es el Estado.
- Solicitado, EsperandoAuto, Viajando y Finalizado son los Estados Concretos.

b)

```
class CtldViajes {
    private:
        Map<int, Viaje *> viajes;
    public:
        void nuevoViaje(int);
        void cambiarEstado(int);
};

void CtldViajes::nuevoViaje(int idUsuario) {
    int idViaje = Utils::getInstance()->nuevoIdViaje();
    viajes[id] = new Viaje(idViaje, idUsuario);
}

void CtldViajes::cambiarEstado(int idViaje) {
    viajes[idViaje]->cambiarEstado();
}

class Viaje {
    private:
        int idViaje;
        int idUsuario;
        int idChofer;
        EstadoViaje *estado;
    public:
        Viaje(int, int);
        void cambiarEstado();
};

Viaje::Viaje(int idViaje, int idUsuario) {
    this->idViaje = idViaje;
    this->idUsuario = idUsuario;
    this->estado = new Solicitado();
    this->estado->setViaje(this);
}

void Viaje::cambiarEstado() {
    EstadoViaje *nuevoEstado = this->estado->darSiguiente();
```

```
    nuevoEstado->setViaje(this);
    this->estado = nuevoEstado;
    delete this->estado;
}

c)
class EstadoViaje {
    public:
        virtual EstadoViaje *darSiguiente() = 0;
        virtual void accion() = 0;
};

d)
class EsperandoAuto : public EstadoViaje {
    private:
        Viaje *v;
    public:
        EstadoViaje *darSiguiente();
        void accion();
};

EstadoViaje *EsperandoAuto::darSiguiente() {
    return new Viajando();
}

void *EsperandoAuto::accion() {
    Utils::getInstance()->reservarPago(v->getIdUsuario(),
                                       v->getIdChofer());
}

class Viajando : public EstadoViaje {
    private:
        Viaje *v;
    public:
        Viajando *darSiguiente();
        void accion();
};

EstadoViaje *Viajando::darSiguiente() {
    return new Finalizado();
}

void *Viajando::accion() {
    Utils::getInstance()->efectuarPago(v->getIdUsuario(),
                                       v->getIdChofer());
}
```