Subí Que Te Llevo

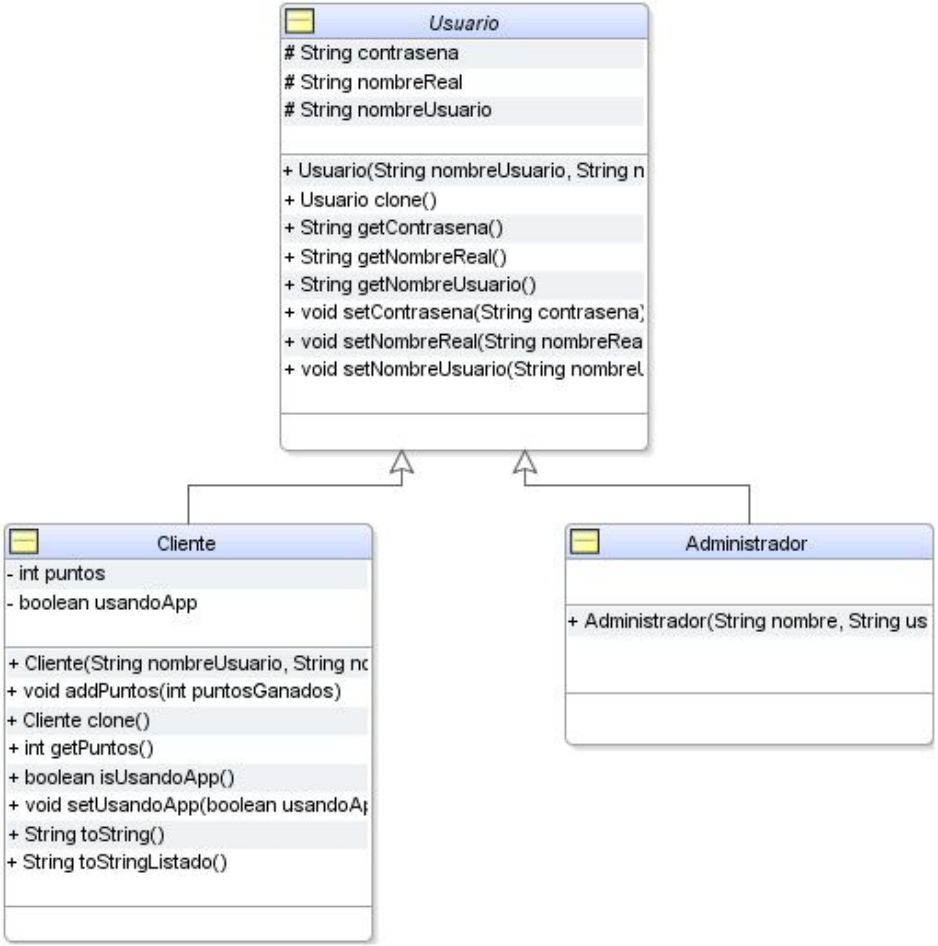
Programación III - Ahumada, Magliotti, San Pedro

INTRODUCCIÓN

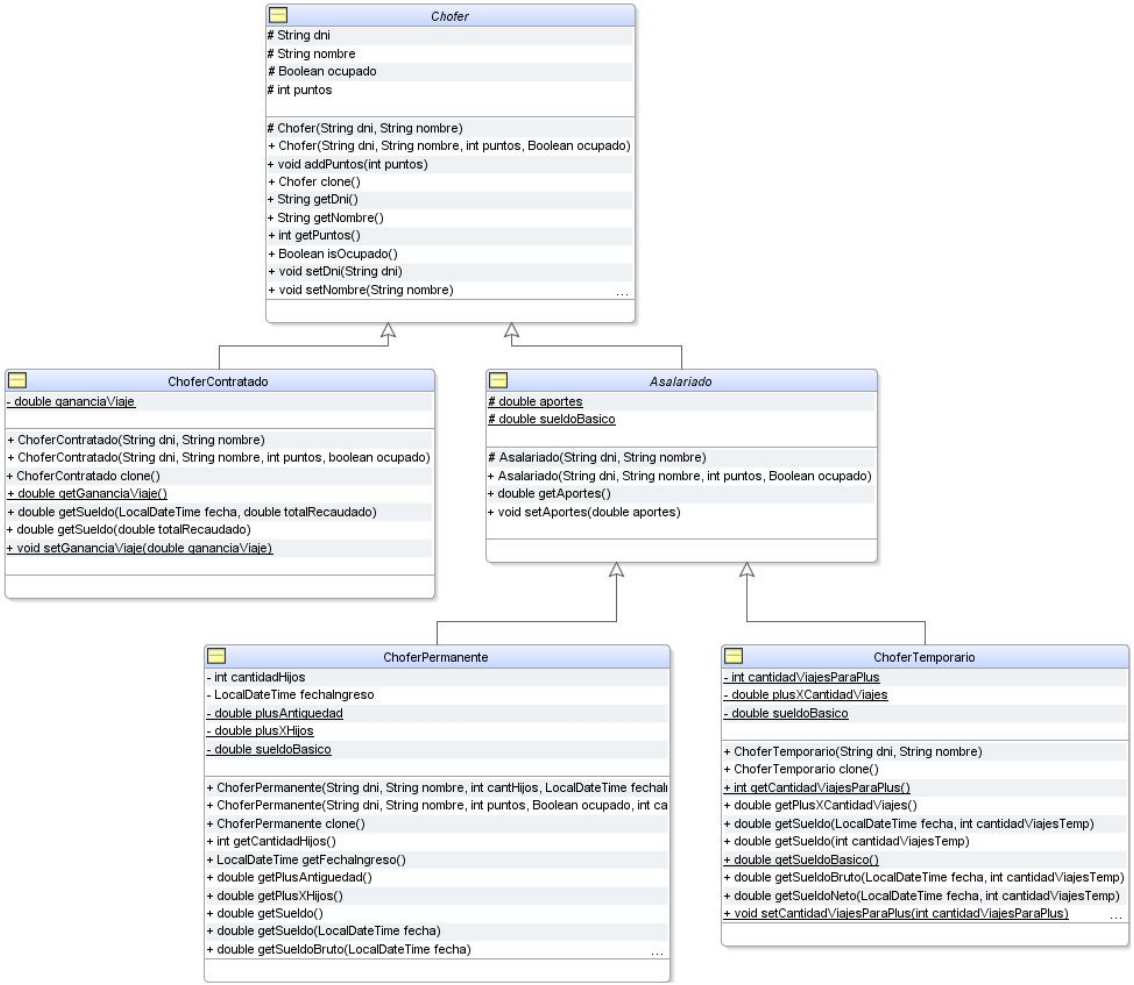
- Servicio de remises
- Aplicación de múltiples patrones
- Diseño de clases

MODELOS

USUARIO


Usuario
String contrasena
String nombreReal
String nombreUsuario
+ Usuario(String nombreUsuario, String n
+ Usuario clone()
+ String getContrasena()
+ String getNombreReal()
+ String getNombreUsuario()
+ void setContrasena(String contrasena)
+ void setNombreReal(String nombreRea
+ void setNombreUsuario(String nombreU
Cliente
- int puntos
- boolean usandoApp
+ Cliente(String nombreUsuario, String nc
+ void addPuntos(int puntosGanados)
+ Cliente clone()
+ int getPuntos()
+ boolean isUsandoApp()
+ void setUsandoApp(boolean usandoAp
+ String toString()
+ String toStringListado()
Administrador
+ Administrador(String nombre, String us

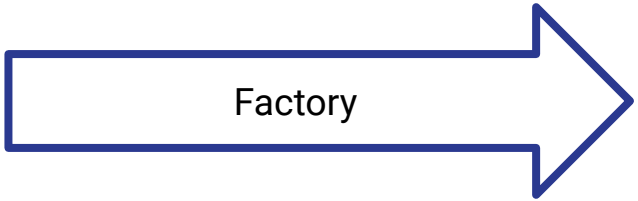
CHOFER


Chofer
String dni
String nombre
Boolean ocupado
int puntos
Chofer(String dni, String nombre)
+ Chofer(String dni, String nombre, int puntos, Boolean ocupado)
+ void addPuntos(int puntos)
+ Chofer clone()
+ String getDni()
+ String getNombre()
+ int getPuntos()
+ Boolean isOcupado()
+ void setDni(String dni)
+ void setNombre(String nombre)
ChoferContratado
- double gananciaViaje
+ ChoferContratado(String dni, String nombre)
+ ChoferContratado(String dni, String nombre, int puntos, boolean ocupado)
+ ChoferContratado clone()
+ double getGananciaViaje()
+ double getSueldo(LocalDateTime fecha, double totalRecaudado)
+ double getSueldo(double totalRecaudado)
+ void setGananciaViaje(double gananciaViaje)
Asalariado
double aportes
double sueldoBasico
Asalariado(String dni, String nombre)
+ Asalariado(String dni, String nombre, int puntos, Boolean ocupado)
+ double getAportes()
+ void setAportes(double aportes)
ChoferPermanente
- int cantidadHijos
- LocalDateTime fechaIngreso
- double plusAntiguedad
- double plusXHijos
- double sueldoBasico
+ ChoferPermanente(String dni, String nombre, int cantHijos, LocalDateTime fechaI
+ ChoferPermanente(String dni, String nombre, int puntos, Boolean ocupado, int ca
+ ChoferPermanente clone()
+ int getCantidadHijos()
+ LocalDateTime getFechaIngreso()
+ double getPlusAntiguedad()
+ double getPlusXHijos()
+ double getSueldo()
+ double getSueldo(LocalDateTime fecha)
+ double getSueldoBruto(LocalDateTime fecha)
ChoferTemporario
- int cantidadViajesParaPlus
- double plusXCantidadViajes
- double sueldoBasico
+ ChoferTemporario(String dni, String nombre)
+ ChoferTemporario clone()
+ int getCantidadViajesParaPlus()
+ double getPlusXCantidadViajes()
+ double getSueldo(LocalDateTime fecha, int cantidadViajesTemp)
+ double getSueldo(int cantidadViajesTemp)
+ double getSueldoBasico()
+ double getSueldoBruto(LocalDateTime fecha, int cantidadViajesTemp)
+ double getSueldoNeto(LocalDateTime fecha, int cantidadViajesTemp)
+ void setCantidadViajesParaPlus(int cantidadViajesParaPlus)

VEHÍCULO

- Factory
- Template

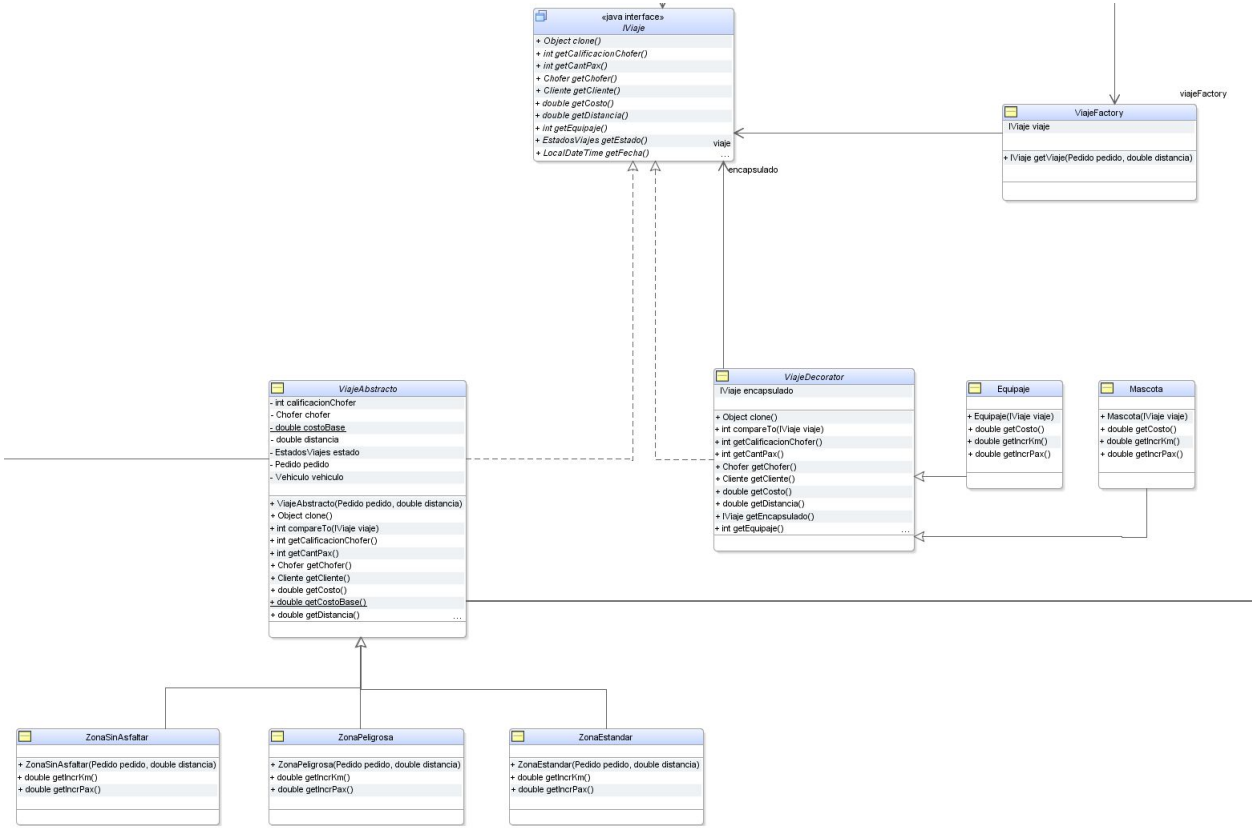
Método principal de VehiculoFactory




```
public Vehiculo getVehiculo(String tipo, String patente)
{
    tipo = tipo.toUpperCase();

    switch(tipo)
    {
    case "AUTO":
        return new Auto(patente);
    case "MOTO":
        return new Moto(patente);
    case "COMBI":
        return new Combi(patente);
    default:
        return null;
    }
}
```

VIAJE

- Interfaz
- Factory
- Decorator



Método principal de ViajeFactory

Factory

Decorator

```
public IViaje getViaje(Pedido pedido, double distancia)
{
    assert pedido != null : "Fallo Pre: El Pedido no puede ser Null ";
    assert distancia > 0 : "Fallo Pre: la distancia debe ser mayor a 0";

    switch(pedido.getZona().toUpperCase()) {
        case "ESTANDAR":
            viaje = new ZonaEstandar(pedido,distancia);
            break;
        case "SIN ASFALTAR":
            viaje = new ZonaSinAsfaltar(pedido,distancia);
            break;
        case "PELIGROSA":
            viaje = new ZonaPeligrosa(pedido,distancia);

    }

    if(pedido.getMascota() != 0) {
        viaje= new Mascota(viaje);
    }

    if(pedido.getEquipaje() != 0) {
        viaje = new Equipaje(viaje);
    }

    return viaje;
}
```

Cómo afecta el decorado

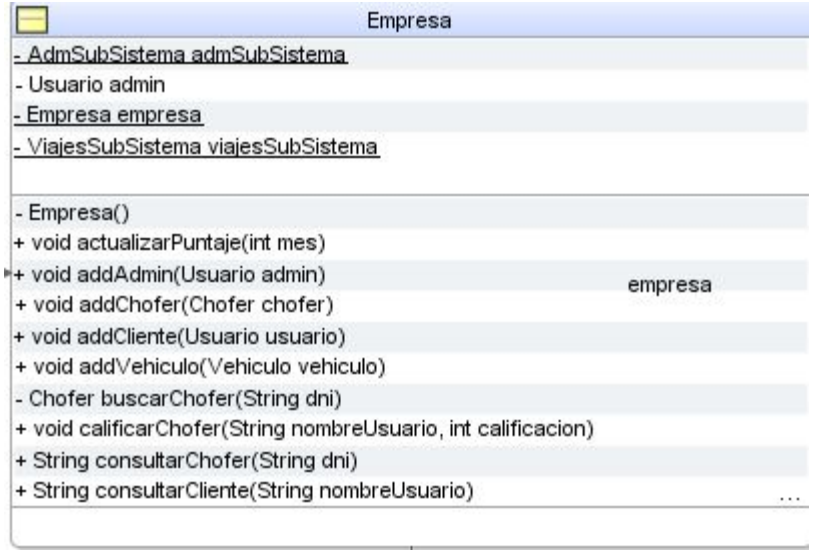
- Encapsulamiento
- Agrega funcionalidad

```java
@Override
public double getCosto() {
    return ViajeAbstracto.getCostoBase() * (1 + this.getIncrKm() + this.getIncrPax());
}

/**
 * Obtiene el incremento por pasajero asociado a la opcion Pet Friendly.<br>
 * @return El incremento por pasajero.
 */
public double getIncrPax() {
    return this.getEncapsulado().getIncrPax() + 0.1 * this.getEncapsulado().getCantPax();
}

/**
 * Obtiene el incremento por kilometro asociado a la opcion Pet Friendly.<br>
 * @return El incremento por kilometro.
 */
public double getIncrKm() {
    return this.getEncapsulado().getIncrKm() + 0.2 * this.getEncapsulado().getDistancia();
}
```

NEGOCIO

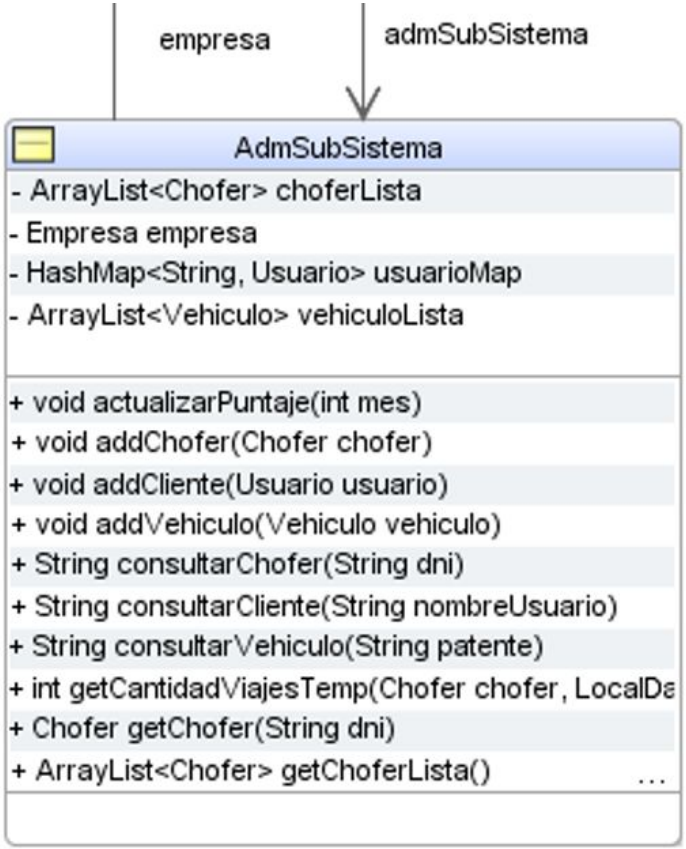
EMPRESA

- Singleton
- Facade
- Viajes subsistema
- Admin subsistema



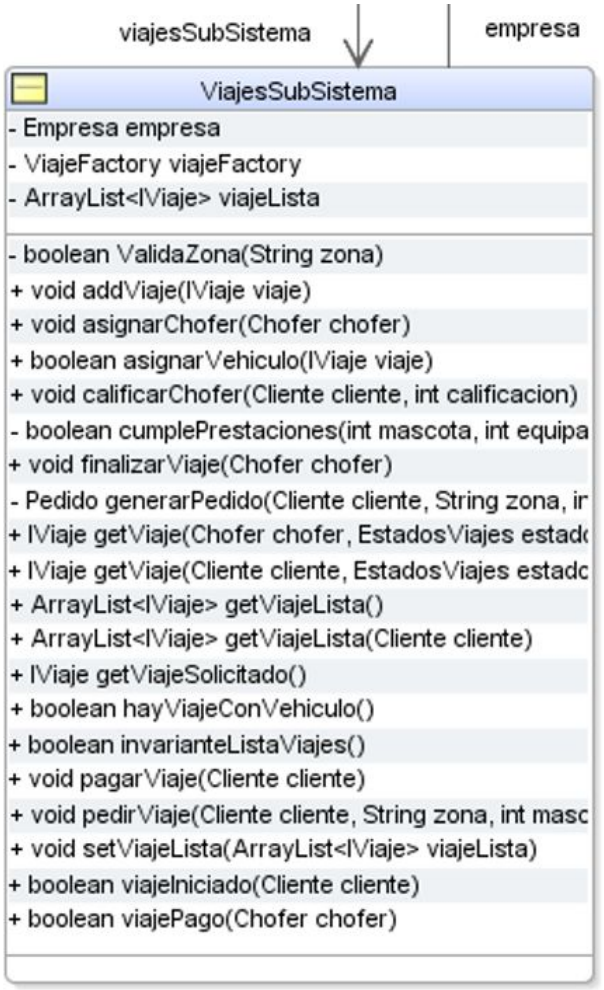
ADMIN SUBSISTEMA

- Maneja vehículos
- Maneja usuarios
- Maneja choferes
- Genera listados y reportes



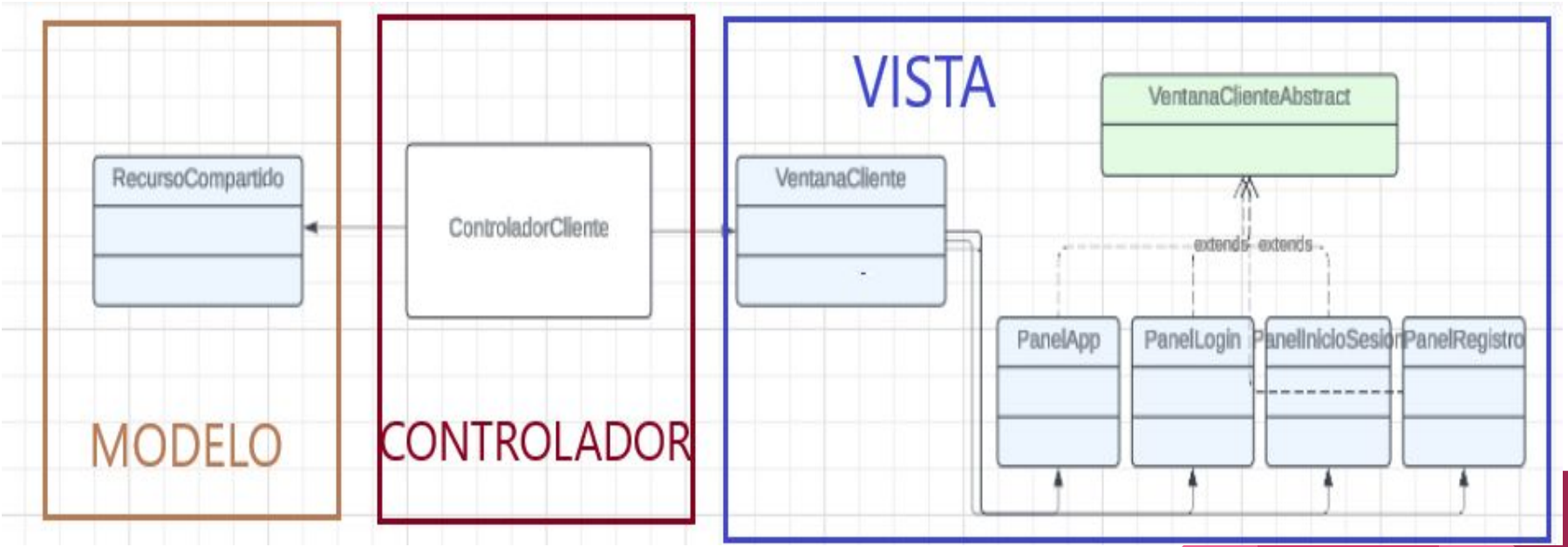
VIAJES SUBSISTEMA

- Crea pedidos y los valida
- Crea viajes
- Asigna choferes
- Finaliza viajes

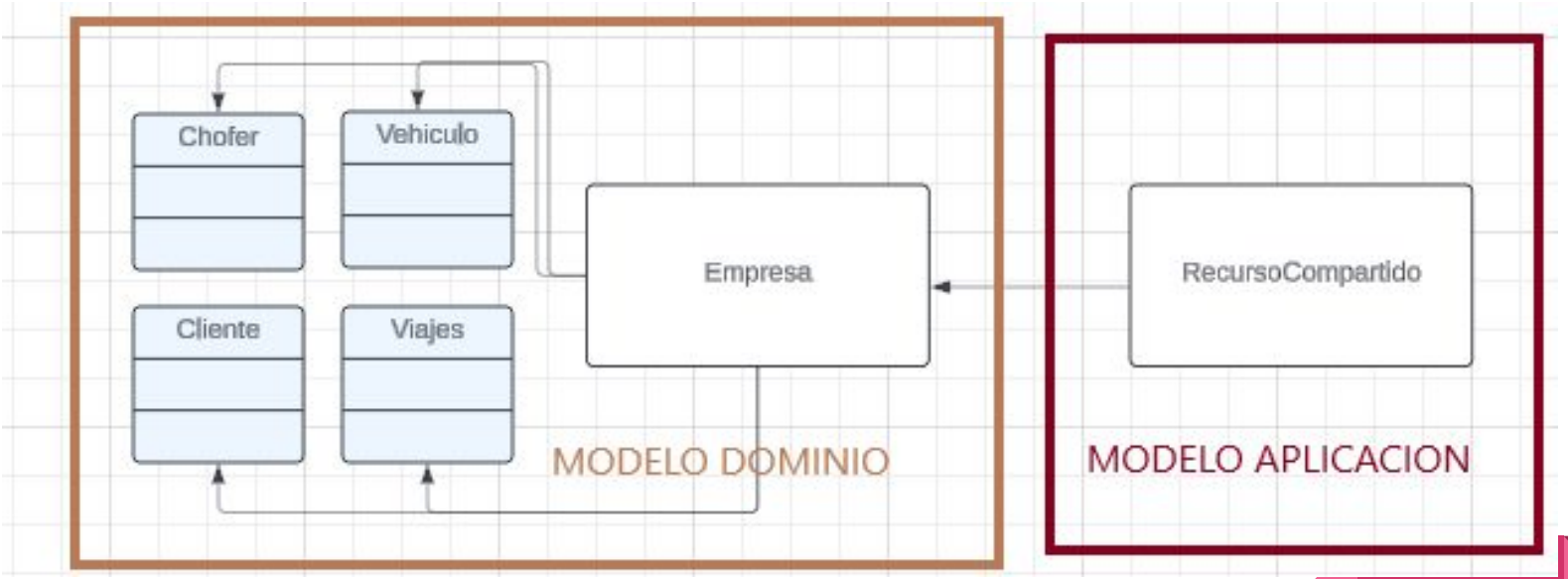


MVC

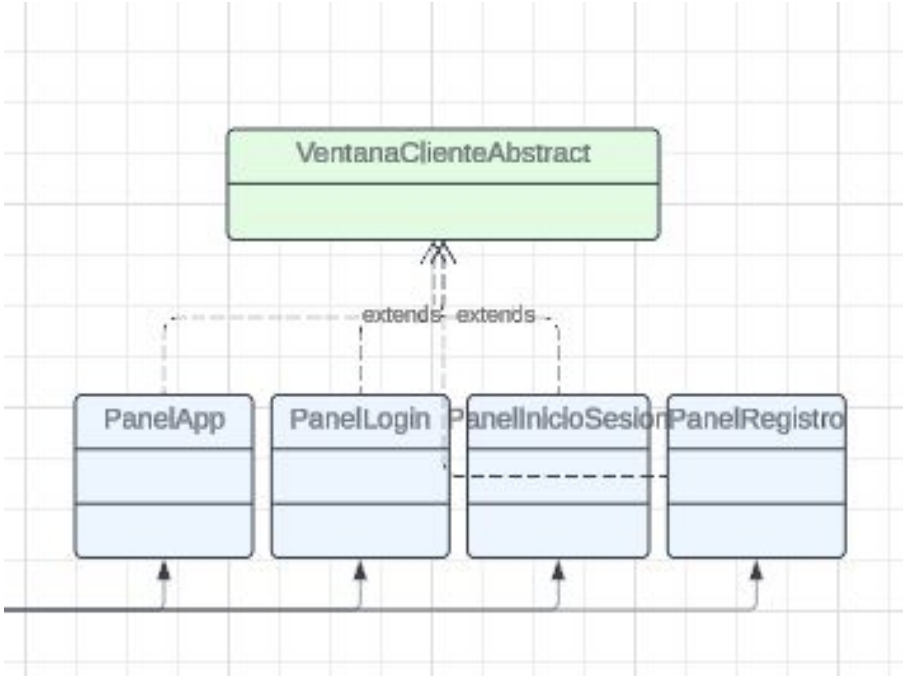
DIAGRAMA GENERAL



MODELO


Chofer
Vehiculo
Cliente
Viajes
Empresa
MODELO DOMINIO
RecursoCompartido
MODELO APLICACION

VISTA


VentanaClienteAbstract
extends
extends
PanelApp
PanelLogin
PanelInicioSesion
PanelRegistro

CONTROLADOR


```
@Override
public void actionPerformed(ActionEvent evento) {
    switch(evento.getActionCommand()) {
        case "ATRAS REG":
        case "ATRAS SESION":
            vista.setLogin();
            break;

        case "REGISTRAR":
            vista.setRegistro();
            break;

        case "INICIAR SESION":
            vista.setInicioSesion();
            break;

        case "INTENTO REGISTRO":
            intentoRegistro();
            break;

        case "INTENTO INICIO SESION":
            intentoInicioSesion();
            break;

        case "PAGAR":
            vista.setDialogFinViaje();
            break;
```

CONCURRENCIA

RECURSO COMPARTIDO

- Monitor
- Comunicación con la empresa
- Sincronismo

pedirViaje

```
*/
public synchronized void pedirViaje(Cliente cliente,String zona, int mascota, String tipoServicio, int equipaje, int cantPax, double distancia, Loc
        throws ExceptionPedido, ExceptionVehiculoDisp
{

   if(cantChoferes>0)
   {
    try{
         empresa.pedirViaje(cliente.getNombreUsuario(), zona, mascota, tipoServicio, equipaje, cantPax, distancia, fecha);
         evento = new EventoSimulacion("solicito viaje y espera",cliente,null,null,TipoEvento.CLIENTE);
    }
    catch(ExceptionVehiculoDisp e){
         evento = new EventoSimulacion("realizo pedido pero no habia auto que cumpla las especificaciones",cliente,null,null,TipoEvento.CLIENTE);
         setChanged();
         notifyObservers(evento);
         notifyAll();
         throw  e;
    }
    catch(ExceptionPedido e){
       evento = new EventoSimulacion("realizo pedido pero fue rechazado "+e.getMessage().toLowerCase(),cliente,null,null,TipoEvento.CLIENTE);
       setChanged();
       notifyObservers(evento);
       notifyAll();
       throw  e;
    } catch (ExceptionUsuario ex) {

    }
   }else
         evento = new EventoSimulacion("no pudo realizar pedido porque no hay choferes disponibles ",cliente,null,null,TipoEvento.CLIENTE);


   setChanged();
   notifyObservers(evento);
   notifyAll();
}
```

asignarVehiculo

```
public synchronized void asignarVehiculo()
{
   IViaje viajeSolicitado = getViajeSolicitado();

   while(simulacionIsActiva() && viajeSolicitado == null)
   {
       try {
            evento = new EventoSimulacion("El sistema intento asignar auto pero no hay viajes solicitados", null, null, null, TipoEvento.SISTEMA);
            setChanged();
            notifyObservers(evento);
            wait();

            viajeSolicitado = getViajeSolicitado();
        }
       catch (InterruptedException ex) {}
    }
```

asignarVehiculo

```
while(simulacionIsActiva()  && !empresa.asignarVehiculo(viajeSolicitado))
{
    evento = new EventoSimulacion("El sistema intento asignar vehiculo al viaje del cliente " + viajeSolicitado.getCliente().getNombreUsuario()
            +" pero no encontro uno libre, sigue en espera", viajeSolicitado.getCliente(), null, null, TipoEvento.SISTEMA);
    setChanged();
    notifyObservers(evento);
    notifyAll();
    viajeSolicitado = getViajeSolicitado();
    try {
        wait();
     } catch (InterruptedException ex) {}
}
```

- Precondición de asignarVehiculo -> viajeSolicitado != null

tomarViaje

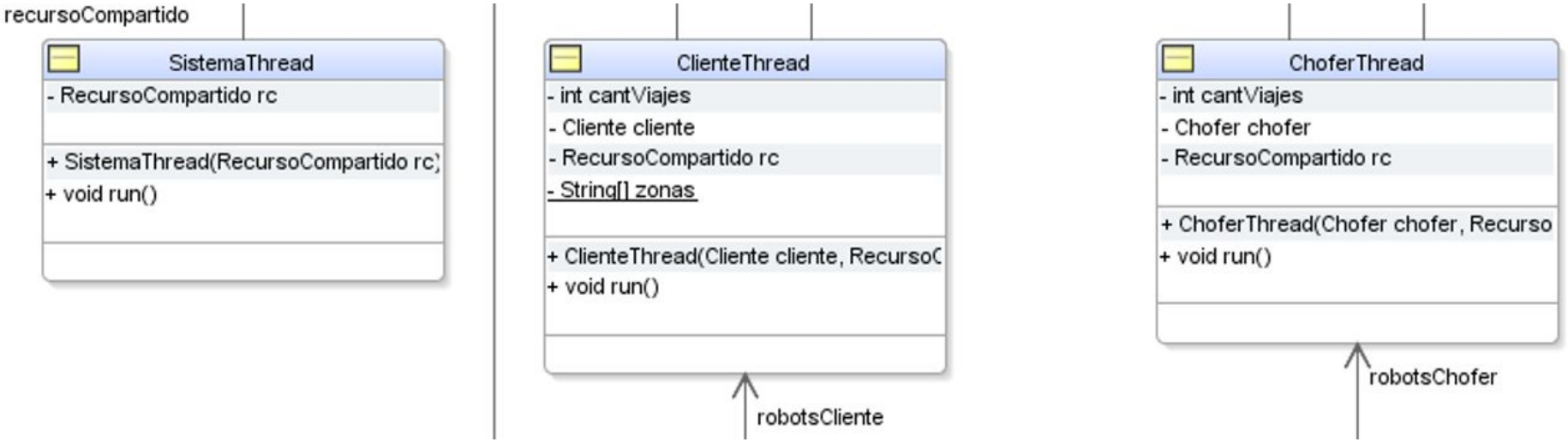
```
public synchronized void tomarViaje(Chofer chofer, int cantViajesPedientes)
{
    IViaje viaje;

    while(hayClientes() && !hayViajeConVehiculo() && (cantViajesPedientes > 0 || usuarioActivo))
    {
        try {
            wait();
        } catch (InterruptedException ex) {}
    }

    if (cantViajesPedientes > 0 || usuarioActivo) {
        if(hayClientes())
        {
            empresa.asignarChofer(chofer);
            viaje = getViaje(chofer,EstadosViajes.INICIADO);
            if(viaje!=null)
                 evento =  new EventoSimulacion("tomo el viaje del cliente "+
                         viaje.getCliente().getNombreUsuario(),viaje.getCliente(),chofer,null,TipoEvento.CHOFER);
            else
                evento =  new EventoSimulacion("intento tomar viaje pero no habia ninguno con vehiculo asignado",
                        null,chofer,null,TipoEvento.CHOFER);
        }
        else
            evento =  new EventoSimulacion("descubre que no hay mas clientes se retira de la empresa",null,chofer,null,TipoEvento.CHOFER);

        setChanged();
        notifyObservers(evento);
    }

    notifyAll();
}
```

pagarViaje / finalizarViaje

```
public synchronized void pagarViaje(Cliente cliente)
{

if(cantChoferes>0)
{
    while(cantChoferes>0 && !viajeIniciado(cliente))
    {
        try {
            wait();
        } catch (InterruptedException ex) {

        }
    }


    if(viajeIniciado(cliente))
    {
        try {
             empresa.pagarViaje(cliente);
             evento = new EventoSimulacion("pago el viaje y se retiro del vehiculo",
                      cliente,getViaje(cliente,EstadosViajes.PAGO).getChofer(),null,TipoEvento.CLIENTE);
        } catch (ExceptionSinViajeaPagar ex) {
            //no entramos nunca porque validamos q tenga viaje iniciado
        } catch (ExceptionUsuario ex) {

         }
    }else
           evento = new EventoSimulacion("cancela el viaje porque no hay choferes disponibles",
                    cliente,null,null,TipoEvento.CLIENTE);


    setChanged();
    notifyObservers(evento);
}
    notifyAll();
}
```

```
public synchronized void finalizarViaje(Chofer chofer, int cantViajesPendientes)
{

    while (simulacionIsActiva() && !viajePago(chofer) && (cantViajesPendientes > 0 || usuarioActivo))
        try {
            wait();
        } catch (InterruptedException ex) {}


    if (viajePago(chofer))
    {
        try {
            empresa.finalizarViaje(chofer);
            evento = new EventoSimulacion("finalizo el viaje y devolvio el vehiculo",
                    getViaje(chofer, EstadosViajes.FINALIZADO).getCliente(), chofer, null, TipoEvento.CHOFER);
            setChanged();
            notifyObservers(evento);
        }
        catch (ExceptionChofer ex) {}
        catch (ExceptionChoferSinViajesPagos ex) {}
    }
    notifyAll();

}
```

THREADS



- Todos extienden de Thread

ClienteThread

```
@Override
public void run()
{
    while(cantViajes>0 && rc.getCantChoferes()>0)
    {

       try{
             Util.espera();
             rc.pedirViaje(cliente, zonas[Util.rand(4)], Util.rand(0,1), "Transporte",
                      Util.rand(0,1),Util.rand(1,12),Util.rand(4000), LocalDateTime.now());

             Util.espera();
             rc.pagarViaje(cliente);
             this.cantViajes--;

       }catch(Exception e){}
     }
    rc.subCliente(cliente);
}
```

ChoferThread

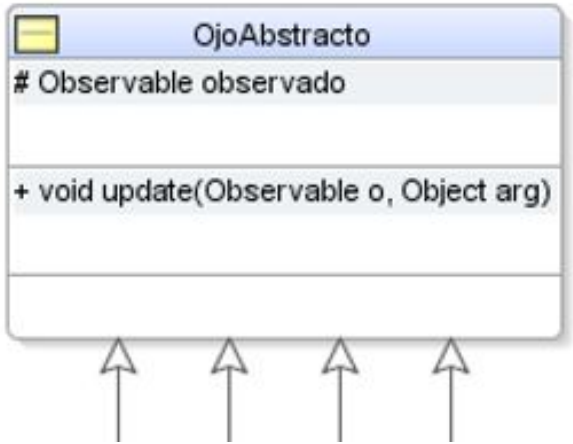
```
@Override
public void run()
{
    while(rc.getUsuarioActivo() || (cantViajes>0 && rc.hayClientes()))
    {
        Util.espera();
        rc.tomarViaje(chofer, cantViajes);

        Util.espera();
        rc.finalizarViaje(chofer, cantViajes);
        cantViajes--;
    }
    rc.subChofer(chofer);
}
```

SistemaThread

```
@Override
public void run()
{
    while(rc.simulacionIsActiva())
    {
        Util.espera();
        rc.asignarVehiculo();
    }
}
```

OBSERVER / OBSERVABLE

- Cambios en el recurso compartido
- Comunicación con múltiples vistas
- El observado solo avisa
- Recurso compartido no interactúa con vista
- Un observer por vista, todos extienden de OjoAbstracto



OjoGeneral

Se agrega como observador

```
public OjoGeneral(VentanaGeneral vista, Observable observado)
{
    this.vista = vista;
    this.observado = observado;
    observado.addObserver(this);
}
```

Cada vez que es notificado

```
@Override
public void update(Observable o, Object e) {
    super.update(o, e);

    EventoSimulacion evento;
    evento = (EventoSimulacion)e;

       switch(evento.getTipo())
       {

           case CLIENTE:  vista.appendGeneral(evento.getCliente().getNombreUsuario()+" "+evento.getMensaje()+"\n");
           break;
           case CHOFER:  vista.appendGeneral(evento.getChofer().getNombre()+" "+evento.getMensaje()+"\n");
           break;
           case SISTEMA: vista.appendGeneral(evento.getMensaje()+"\n");

       }


}
```

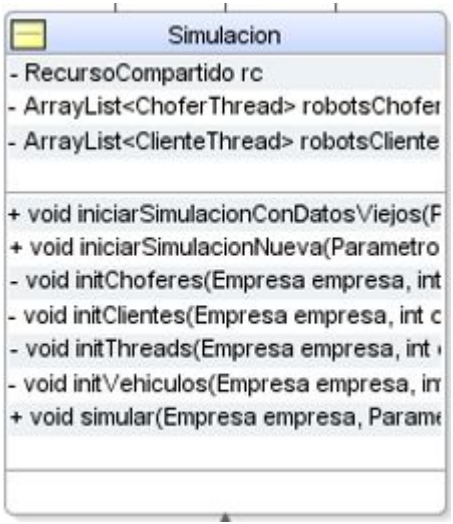
OjoClienteSimulacion

Filtra un cliente

```
@Override
public void update(Observable o, Object e) {
    super.update(o, e);

    EventoSimulacion evento = (EventoSimulacion)e;

    Cliente clienteEvento = evento.getCliente();
    if(clienteEvento != null && clienteEvento.equals(cliente))
    {
        switch(evento.getTipo())
        {
           case CLIENTE:  vista.appendCliente(evento.getCliente().getNombreUsuario()+" "+evento.getMensaje()+"\n");
           break;
           case CHOFER:  vista.appendCliente(evento.getChofer().getNombre()+" "+evento.getMensaje()+"\n");
           break;
           case SISTEMA: vista.appendCliente(evento.getMensaje()+"\n");
        }
    }
}
```

SIMULACION

- Encargada de unir todo lo anterior
- Inicia los hilos
- Tiene el recurso compartido



iniciarSimulacionNueva

```
public void iniciarSimulacionNueva(ParametrosSimulacion p)
{
      Empresa empresa = Empresa.getInstance();
      rc =  new RecursoCompartido(empresa,p.getCantClientes(),p.CantChoferes());

      initClientes(empresa,p.getCantClientes(), p.getCantMaxViajeCliente());

      initChoferes(empresa,p.getCantChoferTemporario(), p.getCantChoferContratado(),
            p.getCantChoferPermanente(), p.getCantMaxViajeChofer());

      initVehiculos(empresa,p.getCantAutos(), p.getCantMotos(), p.getCantCombis());
      simular(empresa,p);
}
```

- ParametrosSimulacion

simular

- Crea observers
- Persiste datos de configuración
- Inicia hilos

```
try {
    persisteEmpresa.abrirOutput("Empresa.xml");
    EmpresaDTO empresaDTO = UTILEmpresa.empresaDTOFromEmpresa(empresa);
    persisteEmpresa.escribir(empresaDTO);
    persisteEmpresa.cerrarOutput();
    if(parametros!=null)
    {
         persisteParametros.abrirOutput("Parametros.xml");
         persisteParametros.escribir(parametros);
         persisteParametros.cerrarOutput();
    }
} catch (IOException ex) {
    Logger.getLogger(Simulacion.class.getName()).log(Level.SEVERE, null, ex);
}
```

```
for(ClienteThread c : robotsCliente)
     c.start();

 SistemaThread sistema = new SistemaThread(rc);
 sistema.start();

for (ChoferThread c : robotsChofer)
     c.start();
```

simularConDatosViejos

- Genera simulacion

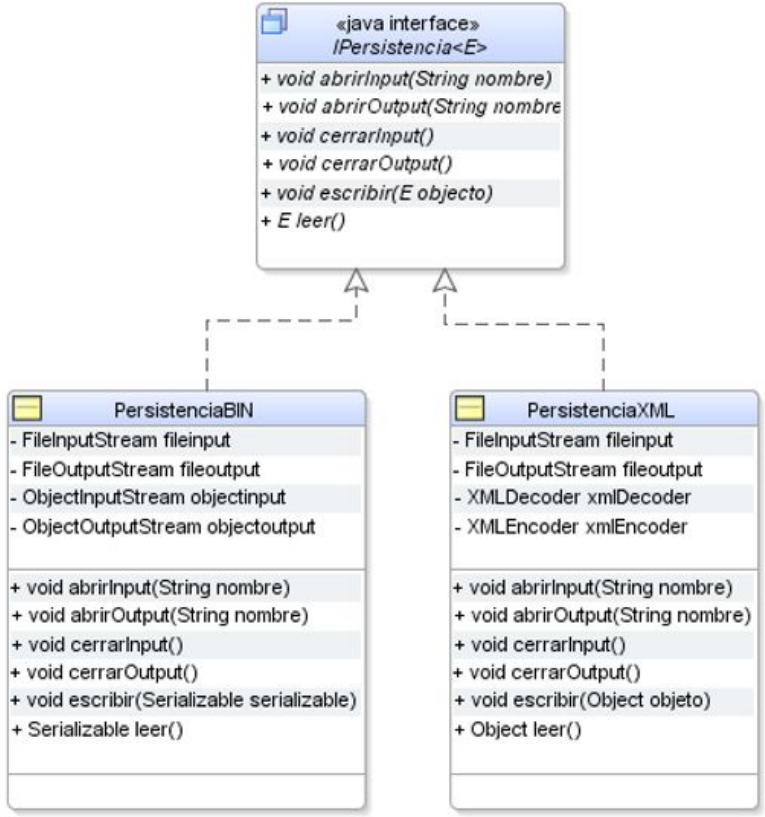
  con datos persistidos

```
public void iniciarSimulacionConDatosViejos(ParametrosSimulacion parametros)
{
    PersistenciaXML leeEmpresa = new PersistenciaXML();

    try {
        leeEmpresa.abrirInput("Empresa.xml");
        EmpresaDTO empresaDTO = (EmpresaDTO) leeEmpresa.leer();
        Empresa empresa = UTILEmpresa.empresaFromEmpresaDTO(empresaDTO);
        leeEmpresa.cerrarInput();


        rc = new RecursoCompartido(empresa, empresa.getChoferLista().size(), empresa.getUsuarioLista().size()-1);
        initThreads(empresa,parametros.getCantMaxViajeChofer(),parametros.getCantMaxViajeCliente());
        simular(empresa,null);

    } catch (IOException ex) {
        Logger.getLogger(Simulacion.class.getName()).log(Level.SEVERE, null, ex);
    }catch (ClassNotFoundException ex) {
        Logger.getLogger(Simulacion.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

PERSISTENCIA

XML y Serialización

- Interfaz IPersistencia define métodos de serialización.
- Clases PersistenciaBIN y PersistenciaXML implementan los métodos:
- *abrirInput(), abrirOutput(), cerrarOutput(), cerrarInput(), escribir() y leer().*



DAO

- Clases de mapeo para convertir entre objetos de dominio y DTOs

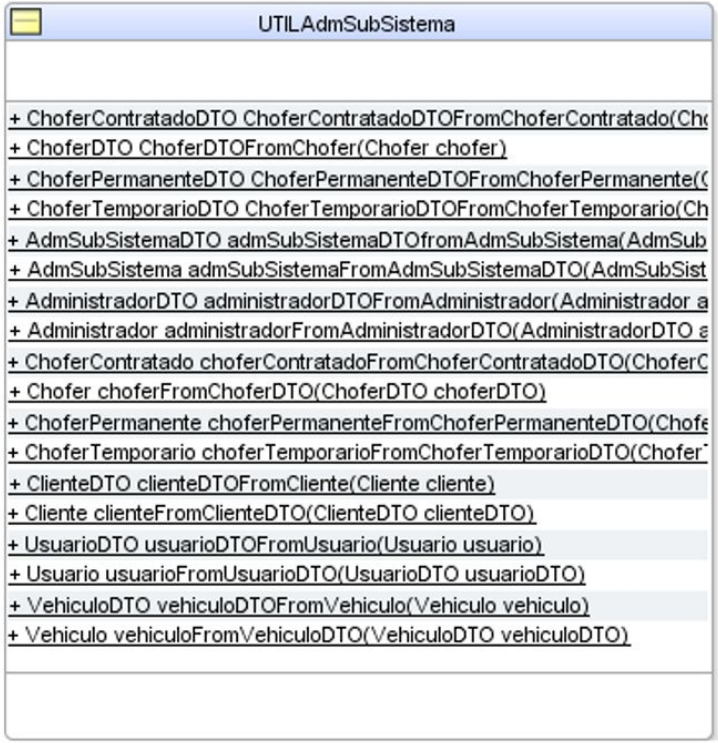
- No encapsulan directamente el acceso a la capa de persistencia.

- Se basan en dicha serialización para guardar y cargar datos.

    - ***UTILEmpresa***.
    - ***UTILAdmSubSistema***.
    - ***UTILViajesSubSistema***.

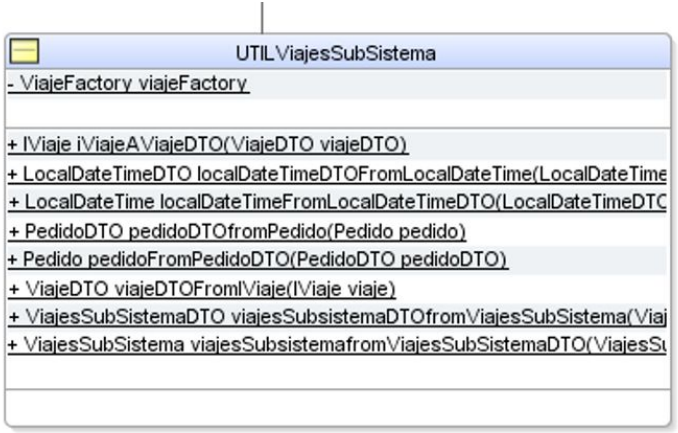
UTILAdmSubSistema

- Métodos para conversión de choferes, vehículos y usuarios.
- Crea AdmSubSistemaDTO a partir de AdmSubSistema.
- Convierte colecciones y objetos para persistencia.



UTILViajesSubSistema

- Métodos para conversión de pedidos y viajes.
- Crea ViajesSubSistemaDTO a partir de ViajesSubSistema.
- Convierte colección de viajes históricos.



UTILEmpresa

- Convierte Empresa y sus subsistemas a DTOs.
- Incluye constructor público debido al patrón Singleton.

DTO

- Las clases DTO se utilizan para transferir datos entre capas.

- Implementan Serializable para permitir la serialización.

- ***ChoferDTO*:**
  - *ChoferContratadoDTO.*
  - *AsalariadoDTO*.
  - *ChoferPermanenteDTO*.
  - *ChoferTemporarioDTO*.
- **VehiculoDTO:**
  - *AutoDTO.*
  - *MotoDTO.*
  - *CombiDTO.*
- **UsuarioDTO:**
  - *ClienteDTO.*
  - *AdministradorDTO*.
- **EmpresaDTO.**
- **AdmSubSistema.**
- **ViajesSubSistema.**
- **ViajeDTO.**
- **PedidoDTO.**
- **LocalDateTimeDTO.**

CONCLUSIONES