1. *(3.5 points)* Using the following description, build the UML class diagram for the proposed system including class attributes and the names of all the relationships (**do not include any methods nor attribute types**).

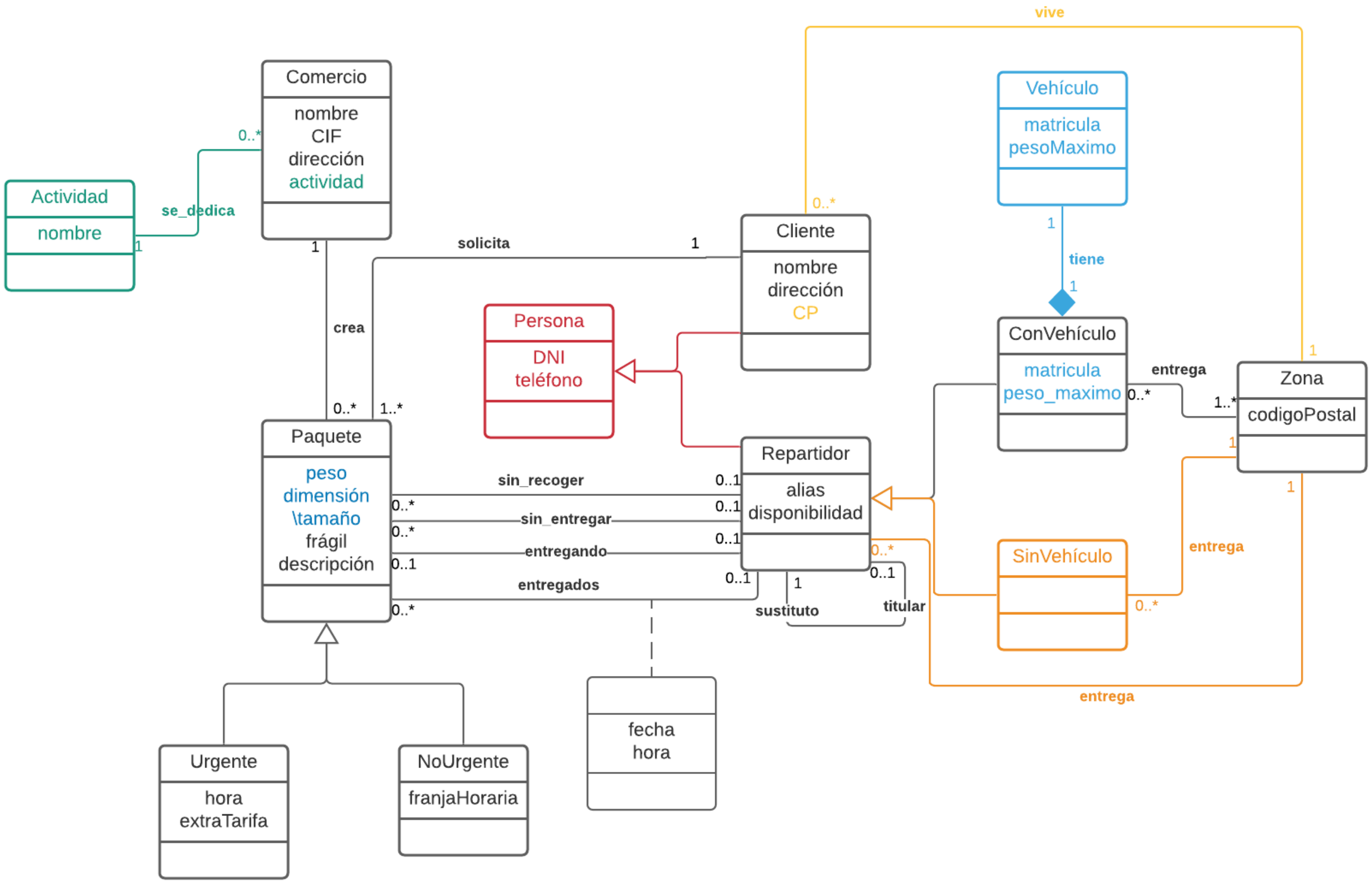
The company ISWSoft needs a software system for small and medium sized companies having customers with mobility restrictions. The app to be developed (*DeliveryApp*) manages the delivery of products (packages) to customers who make orders by using different mechanisms (phone, WhatsApp, Web, etc). In this first version of the product *DeliveryApp* will not handle payments because these will be handled by other means.

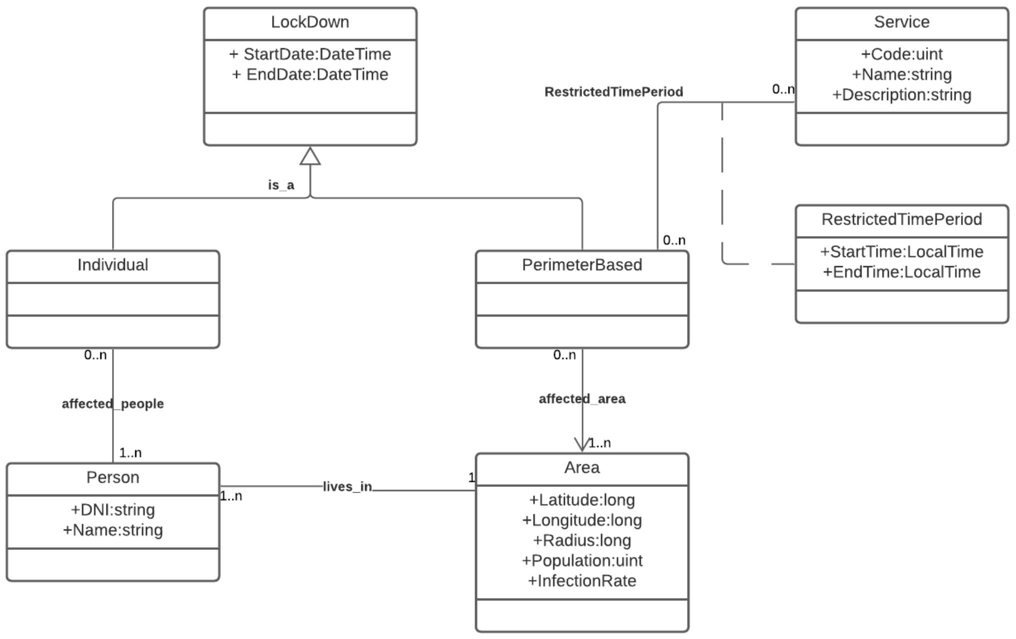
Any company can be registered in *DeliveryApp* by indicating its name, address, Company Id, and its activity. The possible activities of companies that can be selected when registering it are indicated by *DeliveryApp* such as "grocery store", "cafeteria", "restaurant", etc. Customers are registered by companies as soon as they use the system by using an ID number (if a customer already exists it is not registered twice). In addition to the ID number, it is also provided the customer` s name, phone number, address and postal code.

Whenever a delivery must be done, the company inserts the delivery in the system indicating whether it is an urgent delivery or not. If it is an urgent delivery the company must indicate the deadline delivery time (in the same day) to deliver it and the extra cost that the company will pay to the deliverer in addition to the normal pre-agreed delivery cost. In the case of a non-urgent delivery, a delivery time span is assigned (this time span has been previously agreed with the customer). All deliveries are labelled as small, medium size or large depending on its weight and it is also indicated whether the delivery is fragile or not. A description of the delivery may also be included.

Deliverers are also registered in the system by providing an ID number, alias, phone number and delivery zones (expressed as postal codes) in which they provide delivery services. In case the deliverer has its own vehicle (motorbike, car, van) he/she will indicate the vehicle's plate number and the maximum weight that can be delivered. In this case more than one delivery zone can be registered. Deliverers without a vehicle may only deliver small deliveries and in just one delivery zone (postal code). In addition, every deliverer must have a substitute deliverer who will oversee the deliveries in case any unexpected event occurs to the initially assigned deliverer. A substitute deliverer may only be substituting one deliverer.

DeliveryApp implements an algorithm to assign deliveries to deliverers depending on the delivery zone(s) of each deliverer and the deliverer's availability when the delivery is created. The availability is indicated by a deliverer in real time. If a deliverer is available, the system can assign him/her deliveries. The system must have information about all deliveries assigned to each deliverer (but not picked up by the deliverer yet), picked-up deliveries for each deliverer but not delivered, the current delivery being delivered by each deliverer and, finally, the delivered deliveries of each deliverer. As soon as a delivery is delivered the system will record the date and time of delivery. Every company will be able to display the status of its deliveries and the assigned deliverer. The administrator of *DeliveryApp* will issue the payments to deliverers and the charges to companies every 15 days


vive
Comercio
nombre
CIF
dirección
actividad
Vehículo
matricula
pesoMaximo
0..*
Actividad
nombre
se_dedica
1
0..*
1
tiene
1
Cliente
nombre
dirección
CP
solicita
1
1
crea
Persona
DNI
teléfono
ConVehículo
matricula
peso_maximo
entrega
0..*
1..*
Zona
codigoPostal
1
0..*
1..*
Paquete
peso
dimensión
\tamaño
frágil
descripción
Repartidor
alias
disponibilidad
1
1
sin_recoger
0..1
0..*
sin_entregar
0..1
0..*
entregando
0..1
0..1
entregados
0..1
0..*
0..*
0..1
1
sustituto
titular
SinVehículo
entrega
0..*
entrega
Urgente
hora
extraTarifa
NoUrgente
franjaHoraria
fecha
hora



Note1. There is a navigation restriction between *PerimeterBased* and *Area*
Note 2. Do not write class methods only the attributes.

Note 3. *LockDown* is an abstract class.

a) *(1.5 points)* Obtain the C# design using the design patterns studied in this course.
b) *(1 point)* Provide the header of the required constructors (without the body).
c) *(1 point)* Implement the needed C# code to create a LockDown object of type *PerimeterBased* with a restricted service affecting one area. The system must also have another Individual LockDown affecting just to one person. Use any arbitrary parameter values in the constructors to have a consistent system after its creation.

Area

```
public class Area
  {
    //Propiedades directas
    public long Latitude { get; set; }
    public long Longitude { get; set; }
    public long Radius { get; set; }
    public uint Population { get; set; }
    public long InfectionRate { get; set; }

    //Relationships
    public virtual ICollection<Person> LivesInPeople { get; set; }
    //Constructor header
    public Area(long Latitude, long Longitude, long Radius, uint Population,
                    long InfectionRate, Person LivesInPerson) { }
  }
```

LockDown

```
  public abstract class LockDown
  {
    //Propiedades directas
    public DateTime StartDate { get; set; }
    public DateTime EndDate { get; set; }

    //Constructor header
    public LockDown(DateTime StartDate, DateTime EndDate) { }
  }
```

Individual

```
  public class Individual:LockDown
  {
    //Relationships
    public virtual ICollection<Person> AffectedPeople { get; set; }

    //Constructor header
    public Individual(DateTime StartDate, DateTime EndDate, Person
                        AffectedPerson):base(StartDate, EndDate) { }
  }
```

PerimeterBased

```
  public class PerimeterBased:LockDown
  {
    //Relationships
    public virtual ICollection<Area> AffectedAreas { get; set; }
    public virtual ICollection<RestrictedTimePeriod> RestrictedTimePeriods { get; set; }

    //Constructor header
    public PerimeterBased(DateTime StartDate, DateTime EndDate, Area AffectedArea) :
                                                        base(StartDate, EndDate) { }
  }
```

Person

```
  public class Person
  {
    //Propiedades directas
    public string DNI { get; set; }
    public string Name { get; set; }

    // Relationships
    public virtual Area LivesInArea { get; set; }
```

```
        public virtual ICollection<Individual> AffectingIndividualLockDowns { get; set; }

        //relaxed constructor 1-1..n association
        //Constructor header
        public Person(string DNI, string Name) { }
    }
```

Service

```
  public class Service
    {
        //Propiedades directas
        public uint Code { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }

        //Relationships
        public virtual ICollection<RestrictedTimePeriod> RestrictedTimePeriods { get; set; }

        //Constructor header
        public Service(uint Code, string Name, string Description) { }
    }
```

RestrictedTimePeriod

```
   public class RestrictedTimePeriod
    {
        //Propiedades directas
        public LocalTime StartTime { get; set; }
        public LocalTime EndTime { get; set; }

        //Relationships
        public virtual Service RestrictedService { get; set; }
        public virtual PerimeterBased RestrictedPerimeterBasedLockDown { get; set; }

        //Constructor header
        public RestrictedTimePeriod(LocalTime StartTime, LocalTime EndTime, Service
             RestrictedService, PerimeterBased RestrictedPerimeterBasedLockDown) { }
}
```

C)

Main

```
        static void Main(string[] args)
        {
            Person affectedPerson = new Person("11111111A", "Javier Jaen");
            Individual individualLockDown = new Individual(new DateTime(2020, 12, 12), new
                                                         DateTime(2020, 12, 24), affectedPerson);
             Area area = new Area(2021, 1221, 23, 532123, 600, affectedPerson);
             affectedPerson.LivesInArea = area;
```

```
    affectedPerson.AffectingIndividualLockDowns.Add(individualLockDown);

    PerimeterBased perimeterBasedLockDown = new PerimeterBased(new
                        DateTime(2020, 11, 12), new DateTime(2020, 11, 21), area);
    Service affectedService = new Service(23, "Hotel Cristina", "Hotel Cerrado por foco
                                                                    Covid");
    RestrictedTimePeriod period = new RestrictedTimePeriod(new LocalTime(22, 00),
                          new LocalTime(23, 30), affectedService, perimeterBasedLockDown);
    perimeterBasedLockDown.RestrictedTimePeriods.Add(period);
   affectedService.RestrictedTimePeriods.Add(period);
}
```