ISW CASE 1 Evaluation

25/10/2024

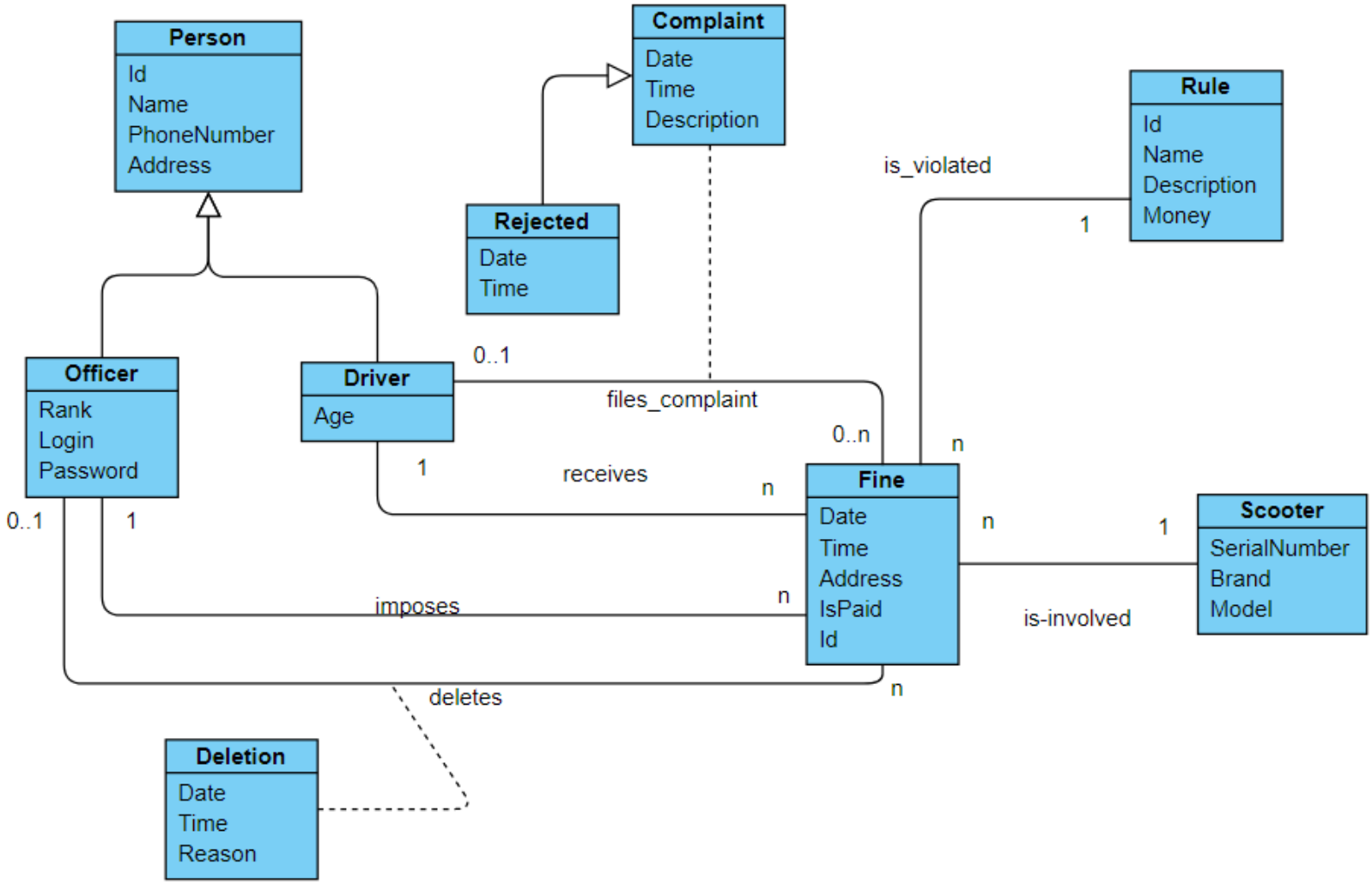
ISWSoft must develop a system for the local police department to manage fines imposed to scooter owners who violate the existing rules. The system is accessed by police officers who must be logged in using a personal login and password. Officers are recorded in the system with their personal Id number, name, phone number, address and rank.

If an officer detects a violation, he(she) immediately stops the scooter driver and asks him(her) his identification card. In order to record a fine, the officer must insert the info of the driver: personal Id number, name, phone number, address and age; the information about the scooter: Serial number, brand and model; the time, date and address when/where the violation was detected; the violated rule (each rule has an Id, a name, a description and the amount of money to be paid). The system automatically creates and assigns an Id for the fine.

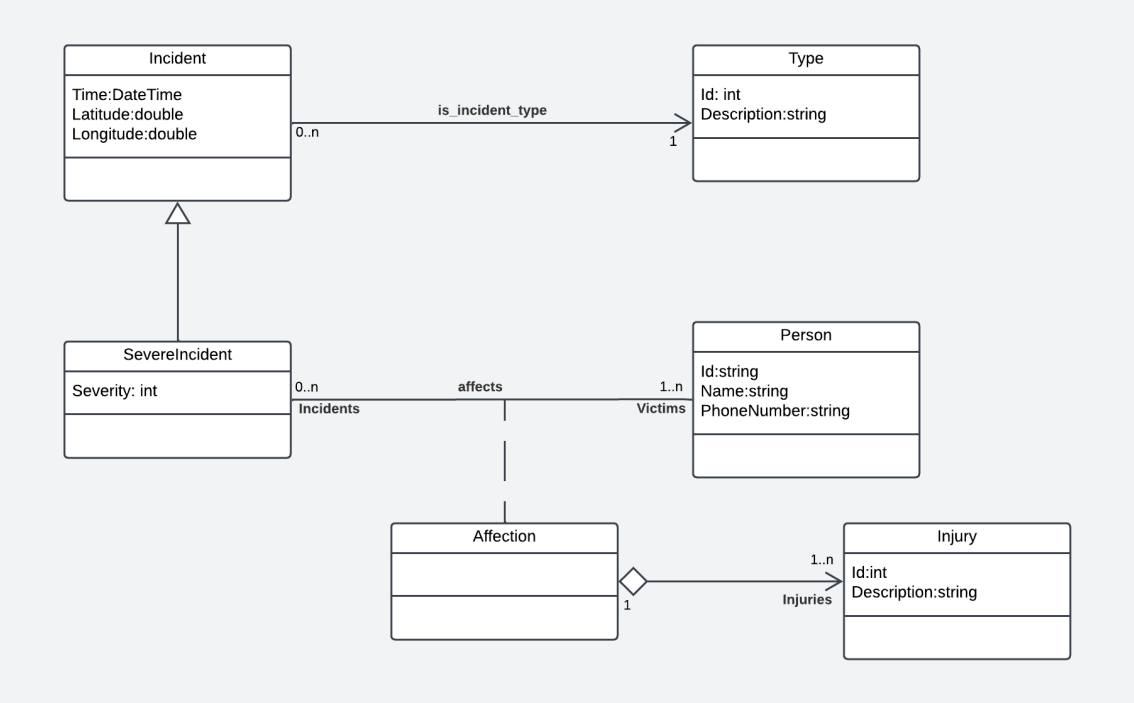
The officer who imposed the fine must also be recorded in case any complaint is filed. A complaint can be filed by a driver regarding a previous existing fine. In order to record a complaint, the driver must provide the Id of the fine and his(her) personal Id number (as an extra security crosscheck). The system validates that the fine exists and that it corresponds to the complaining driver and if this validation is successful, it allows the driver to describe his(her) complaint. When adding a complaint, the date and time in which the complaint was made, the driver who files the complaint, the reason explaining the complaint (text) and the associated fine must be recorded by the system.

If the complaint is favorably resolved, then the fine is marked as deleted by an officer (but the fine is not physically deleted from the system). If a fine is marked as deleted the system must record the officer who performed it, the time and date of the delete operation, and the reason (text description) why it was deleted. In case a complaint is not resolved favorable or there is no complaint, the fine must be paid. When the payment is externally notified (payment notifications are not part of this system), the fine is marked automatically by the system as paid.

Question 1 (5 points): Obtain the UML class diagram for the system described above


Person
Id
Name
PhoneNumber
Address
Complaint
Date
Time
Description
Rule
Id
Name
Description
Money
Rejected
Date
Time
is_violated
1
Officer
Rank
Login
Password
Driver
Age
0..1
files_complaint
0..n
n
1
receives
n
Fine
Date
Time
Address
IsPaid
Id
0..1
1
n
1
Scooter
SerialNumber
Brand
Model
imposes
n
is-involved
n
deletes
Deletion
Date
Time
Reason

Question 2 (5 points): Given the following class diagram of a Weather Incidents Management System:



a) (2 points) Obtain the associated C# design using C# properties according to the design patterns studied in this course.

b) (2 points) Define the headers of the constructors with parameters of the previous classes (**DO NOT** implement the constructors, Ids **MUST** be passed as parameters).

c) (1 point) Using the previous constructors write the code to create a Severe incident of severity 8, affecting two persons one with an injury described as ”leg bone injury” and another described as “head bleeding”. Use any arbitrary attribute values for the remaining properties.

```
public class Affection
{
    public virtual Person Victim { get; set; }
    public virtual SevereIncident Incident { get; set; }
    public virtual ICollection<Injury> Injuries { get; set; }
    public Affection( SevereIncident Incident, Injury injury, Person Victim) {}
}

public class Incident
{
    public DateTime Time { get; set; }
    public double Latitude { get; set; }
    public double Longitude { get; set; }
    public virtual Type Type { get; set; }
    public Incident(double Latitude,  double Longitude, DateTime Time, Type Type) { }
}

public class Injury
{
    public int Id { get; set; }
    public string Description { get; set; }
    public Injury(string Description,int Id) { }
}

public class Person
{
    public string Id { get; set; }
    public string Name { get; set; }
```

```
    public string PhoneNumber { get; set; }
    public virtual ICollection<Affection> Affections { get; set; }
    public Person(string Id, string Name, string PhoneNumber) { }
}

public class SevereIncident: Incident
{
    public int Severity { get; set; }
    public virtual ICollection<Affection> Affections { get; set; }

    //Relaxed 1-1 relationship with Affection
    public SevereIncident(double Latitude, double Longitude, DateTime Time, Type Type,
                    int Severity) : base( Latitude,  Longitude, Time, Type) { }
}

public class Type
{
    public int Id { get; set; }
    public string Description { get; set; }
    public Type(string Description,int Id) { }
}

public class Program
{
    static void Main(string[] args)
    {
        Person p1 = new Person("11111111A", "Javier Jaen", "961506644");
        Person p2 = new Person("22222222B", "Elena Navarro", "961403344");
        SevereIncident si = new SevereIncident(0.123456, 10.94567, DateTime.Now,
                                                new Type("Car accident", 1), 8);
        Affection aff1 = new Affection(si, new Injury("leg bone injury", 1), p1);
        p1.Affections.Add(aff1);
        si.Affections.Add(aff1);
        Affection aff2 = new Affection(si, new Injury("head bleeding", 2), p2);
        p2.Affections.Add(aff2);
        si.Affections.Add(aff2);

    }
}
```