Firefighter Training Simulation System (Case Study)

Case Study

Fire Department has hired you to make a training and simulation system for them.

In this system they have Fire Trucks. Where each Fire Truck contains a Ladder and a Hose Pipe.Hose pipes are detachable from the truck. Hose pipes are either made of synthetic rubber or soft plastic and they can be either be cylindrical or circular in shape. They have specific diameter and water flow rate.

Ladder has a specific length and colour and they are built right into the truck (i.e., they cannot be separated from the truck).

Each FireTruck has a Firefighter as its Driver. FireFighter has a name. He can drive the fire truck and can extinguish fire as well.

They have a Fire Chief as well. The fire chief is just another firefighter. He can drive a truck. He can put out fires. But he can also delegate responsibility for putting out a fire to another firefighter.

Solution

Step 1:
Identify the Classes which have no dependency on other Classes.
Note: We will only make the Classes for those who have distinctive Attributes.

Case Study

Fire Department has hired you to make a training and simulation system for them.

In this system they have Fire Trucks. Where each Fire Truck contains a Ladder and a Hose Pipe. Ladder has a specific length and colour and they are built right into the truck (i.e., they cannot be separated from the truck).

Hose pipes are detachable from the truck. Hose pipes are either made of synthetic rubber or soft plastic and they can be either be cylindrical or circular in shape. They have specific diameter and water flow rate.

Each FireTruck has a Firefighter as its Driver. FireFighter has a name. He can drive the fire truck and can extinguish fire as well.

They have a Fire Chief as well. The fire chief is just another firefighter. He can drive a truck. He can put out fires. But he can also delegate responsibility for putting out a fire to another firefighter.

Domain Model

Ladder

FireFighter

HosePipe

Solution

Step 2:
Identify the Classes which have dependency on other Classes.
Note: We will only make the Classes for those who have distinctive Attributes.

Case Study

Fire Department has hired you to make a training and simulation system for them.

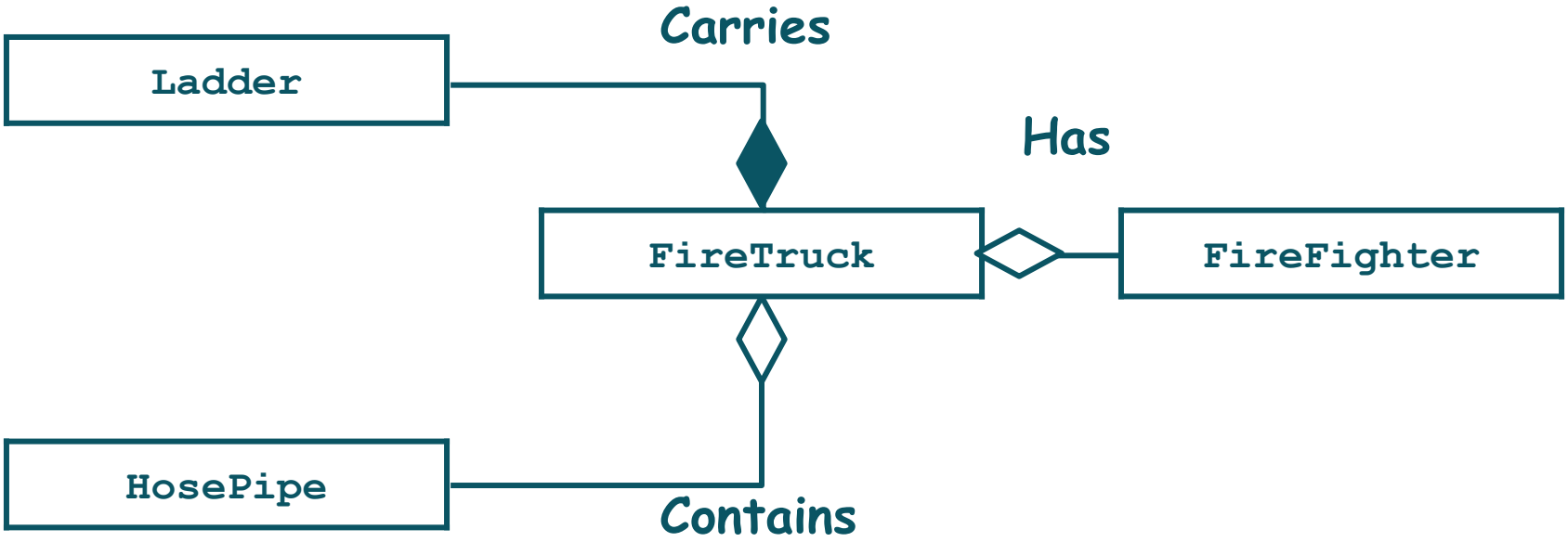
In this system they have Fire Trucks. Where each Fire Truck contains a Ladder and a Hose Pipe. Ladder has a specific length and colour and they are built right into the truck (i.e., they cannot be separated from the truck).

Hose pipes are detachable from the truck. Hose pipes are either made of synthetic rubber or soft plastic and they can be either be cylindrical or circular in shape. They have specific diameter and water flow rate.

Each FireTruck has a Firefighter as its Driver. FireFighter has a name. He can drive the fire truck and can extinguish fire as well.

They have a Fire Chief as well. The fire chief is just another firefighter. He can drive a truck. He can put out fires. But he can also delegate responsibility for putting out a fire to another firefighter.

Domain Model



Solution

Step 3:

Identify the Classes which are inherited from other Classes.

Case Study

Fire Department has hired you to make a training and simulation system for them.

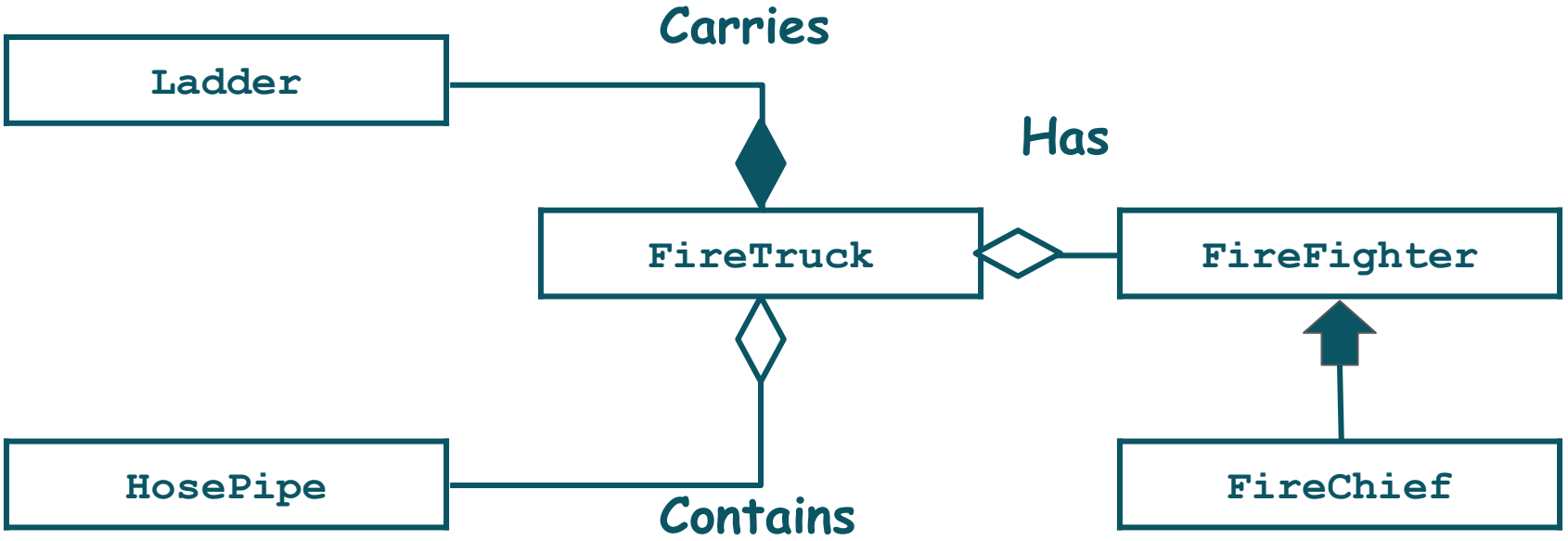
In this system they have Fire Trucks. Where each Fire Truck contains a Ladder and a Hose Pipe. Ladder has a specific length and colour and they are built right into the truck (i.e., they cannot be separated from the truck).

Hose pipes are detachable from the truck. Hose pipes are either made of synthetic rubber or soft plastic and they can be either be cylindrical or circular in shape. They have specific diameter and water flow rate.

Each FireTruck has a Firefighter as its Driver. FireFighter has a name. He can drive the fire truck and can extinguish fire as well.

They have a Fire Chief as well. The fire chief is just another firefighter. He can drive a truck. He can put out fires. But he can also delegate responsibility for putting out a fire to another firefighter.

Domain Model

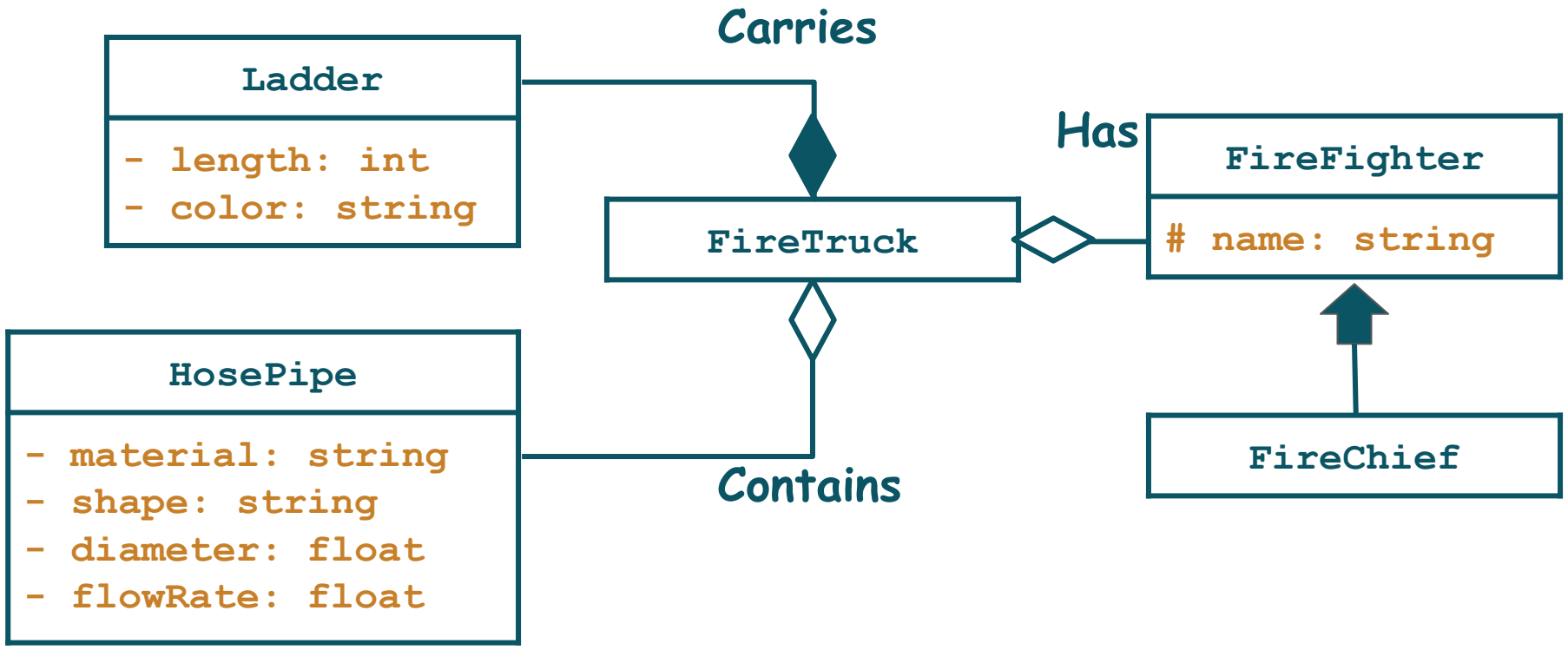


Solution

Step 4:

Draw the Class Diagram with Attributes.

Domain Model

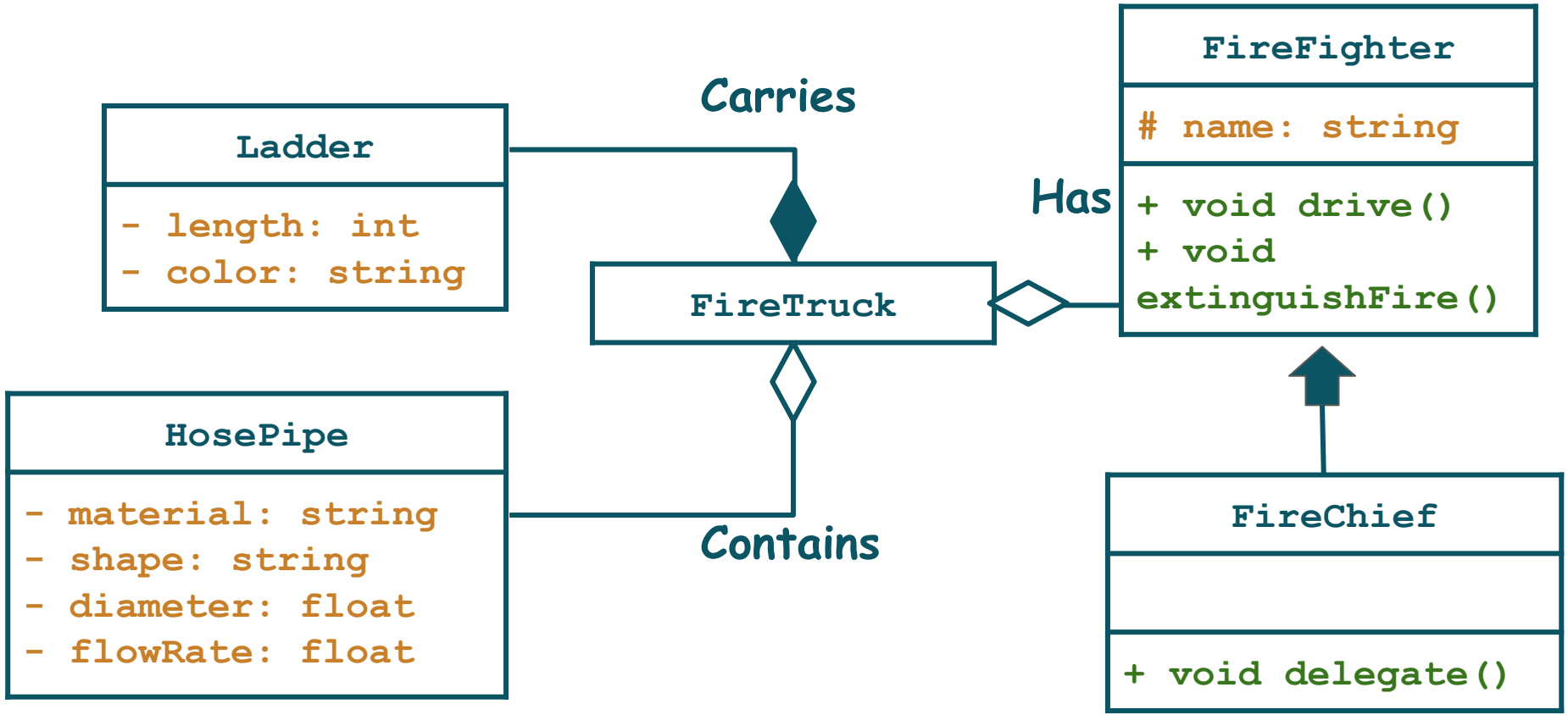


Solution

Step 5:

Draw the Class Diagram with Attributes and Functions.

Domain Model



Solution

Step 6:

Write the C# code for the Classes.

Classes Code: Ladder

```
class Ladder
{
    private int length;
    private string color;
    public Ladder(int length, string color)
    {
        this.length = length;
        this.color = color;
    }
}
```

Classes Code: HosePipe

```
class HosePipe
{
    private string material;
    private string shape;
    private float diameter;
    private float flowRate;

    public HosePipe(string material, string shape, float diameter, float flowRate)
    {
        this.material = material;
        this.shape = shape;
        this.diameter = diameter;
        this.flowRate = flowRate;
    }
}
```

Classes Code: FireFighter

```
class FireFighter
{
    private string name;
    public FireFighter(string name)
    {
        this.name = name;
    }
    public void drive()
    {
        Console.WriteLine(name + " is Driving the Truck");
    }
    public void extinguishFire()
    {
        Console.WriteLine(name + " is Extinguishing the Fire");
    }
}
```

Classes Code: FireChief

```
class FireChief : FireFighter
{
    public FireChief(string name) : base(name)
    {
    }

    public void delegateResponsibility(string FirefighteName)
    {
        Console.WriteLine("Tell " + FirefighteName + " to extinguish fire");
    }
}
```

Classes Code: FireTruck

```
class FireTruck
{
    private Ladder l1;
    private HosePipe h1;
    private FireFighter driver;

    public FireTruck(HosePipe h1, FireFighter driver)
    {
        l1 = new Ladder(34, "Black");
        this.h1 = h1;
        this.driver = driver;
    }
}
```

Task

Assume that BL, DL and UI functions are implemented.

Make a menu driven application with
1st option to add the Firefighter.
2nd option to add a Fire Truck (first show all the available drivers to the user and then user will choose which driver he will add for the fire truck).