

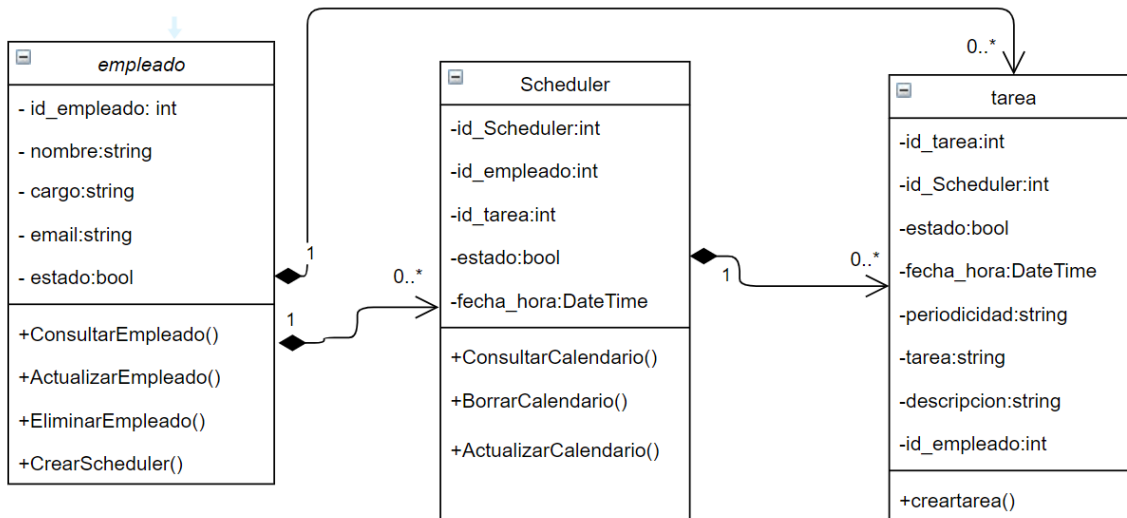
1. El estacionamiento ACME ubicado en la ciudad de Bogotá, tiene las siguientes reglas para calcular el valor total a pagar por servicio de estacionamiento brindado.

```
1 using System;
2
3 namespace Parqueadero
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             DateTime HoraIngreso;
10            DateTime HoraSalida;
11            DateTime FechaIngreso;
12            TimeSpan CntHoras;
13            int TotalHoras;
14            float ValorPagar;
15
16            Console.WriteLine("Ingrese la hora de inicio formato militar hh:mm");
17            HoraIngreso = DateTime.Parse(Console.ReadLine());
18            Console.WriteLine("Ingrese la hora de salida formato militar hh:mm");
19            HoraSalida = DateTime.Parse(Console.ReadLine());
20
21            if (HoraIngreso > HoraSalida)
22            {
23                Console.WriteLine("la hora de salida no puede ser menor a la de entrada");
24
25                Console.WriteLine("Ingrese la hora de inicio formato militar hh:mm");
26                HoraIngreso = DateTime.Parse(Console.ReadLine());
27                Console.WriteLine("Ingrese la hora de salida formato militar hh:mm");
28                HoraSalida = DateTime.Parse(Console.ReadLine());
29            }
30
31            Console.WriteLine("Ingrese la fecha formato dd/mm/yyyy");
32            FechaIngreso = DateTime.Parse(Console.ReadLine()).Date;
33
34            if (FechaIngreso.Month > 1 && FechaIngreso.Month <= 11)
35            {
36                CntHoras = HoraSalida - HoraIngreso;
37                TotalHoras = Convert.ToInt32(Math.Ceiling(CntHoras.TotalHours));
38
39                if (TotalHoras == 1)
40                {
41                    Console.WriteLine("-----");
42                    Console.WriteLine("Total de horas: " + TotalHoras);
43                    ValorPagar = 1000;
44                    Console.WriteLine("El Valor a pagar es: $" + ValorPagar);
45                    Console.WriteLine("-----");
46                }
47                else if (TotalHoras > 1)
48                {
49                    Console.WriteLine("-----");
50                    Console.WriteLine("Total de horas: " + TotalHoras);
51                    ValorPagar = ((TotalHoras - 1) * 500) + 1000;
```


```
52         Console.WriteLine("El Valor a pagar es: $" + ValorPagar);
53         Console.WriteLine("-----");
54     }
55
56 }
57 else {
58     CntHoras = HoraSalida - HoraIngreso;
59     TotalHoras = Convert.ToInt32(Math.Ceiling(CntHoras.TotalHours));
60
61     if (TotalHoras <= 2)
62     {
63         Console.WriteLine("-----");
64         Console.WriteLine("Total de horas: " + TotalHoras);
65         ValorPagar = 2500;
66         Console.WriteLine("El Valor a pagar es: $" + ValorPagar);
67         Console.WriteLine("-----");
68     }
69     else if (TotalHoras > 2)
70     {
71         Console.WriteLine("-----");
72         Console.WriteLine("Total de horas: " + TotalHoras);
73         ValorPagar = ((TotalHoras - 2) * 800) + 2500;
74         Console.WriteLine("El Valor a pagar es: $" + ValorPagar);
75         Console.WriteLine("-----");
76     }
77
78 }
79
80 }
81 }
```

2. La empresa ACME necesita un sistema online que le permita a sus empleados realizar la programación de sus tareas, el sistema a diseñar debe permitir planificar tareas futuras y mostrarlas en forma de planificador diario. Las tareas pueden ser elementos únicos, con una fecha y hora específicas, o recurrentes, como las que se repiten todos los días de la semana o se repiten semanal, mensual o anualmente. Con esta información proponga un diagrama de clases que modele el módulo de planificación de tareas (Scheduler)

Planificador de tareas →



3. Teniendo en cuenta la siguiente estructura de tabla realice las siguientes consultas en SQL ESTANDAR.

| Product | |
|---|---------------|
|  | Id |
| | ProviderId |
| | Code |
| | Description |
| | StockMin |
| | StockMax |
| | UnitCost |
| | UnitSale |
| | UserCreate |
| | CreateDate |
| | RequiredPhone |
| | Enabled |
| | Deleted |
| | Quantity |
| | ProductTypeId |
| | UserModify |
| | ModifyDate |

3.1. Seleccione el promedio de valor de venta (campo de UnitSale) de todos los productos activos en el sistema, cuyo valor máximo en stock (StockMax) se encuentra por debajo del valor del stock mínimo (StockMin) que debe tener el producto en inventario.

Nota: El campo Enabled indica si un producto se encuentra activo o inactivo donde **true** = Activo - **false** = Inactivo

```
SELECT P.Code AS CódigoProducto, AVG(P.UnitSale) AS PromedioValor
FROM Product P
WHERE P.Enabled='true'
GROUP BY P.Code
HAVING MAX(P.StockMax)< MAX(P.StockMin)
```

3.2. Seleccione todos los productos cuyo stock mínimo se encuentre entre los valores 25 y 30 (campo **StockMin** indica el valor mínimo que debe tener el producto en el inventario).

```
SELECT P.Code AS CódigoProducto, MIN(P.StockMin) AS StockMin
FROM Product P
GROUP BY P.Code
HAVING MIN(P.StockMin)>=25 AND MIN(P.StockMin)<=30
```

3.3. De todos los productos activos en el sistema, seleccione el producto que más unidades tiene en inventario y el producto que menos unidades tiene en inventario, todo en una misma consulta.

Nota: Las unidades en inventario se almacena en la columna “**Quantity**”.

```
SELECT P.Code AS CódigoProducto, P.Quantity
FROM Product P
WHERE P.Enabled='true' and P.Quantity =(SELECT MAX(P.Quantity) FROM Product P)
OR P.Quantity =(SELECT MIN(P.Quantity) FROM Product P)
```