Seminari

SET5- 1

Tema 5.

Disseny de la Capa Lògica

(Exercicis)

Enginyeria del Programari
ETS Enginyeria Informàtica
DSIC – UPV

Curs 2021-2022

Objectius

- Disseny d'Objectes- Treball en equip (2 – 4 persones)
  - Diagrames de Classe
- Treball individual / equip

Exercicis:

- Butlletí d'exercicis Tema 5.

Obtenir el disseny en C# del diagrama de classes proposat en els exercicis anteriors, d'acord amb les pautes de disseny vistes en classe.

Qüestió

UML disposa de dos mitjans per a la representació de propietats de les associacions: els atributs d'enllaç i les classes associació. Explica quina és la diferencia entre ells, i com es podrien dissenyar/implementar els atributs d'enllaç en associacions 1:1, 1:N i M:N.

Problema re-enginyeria (1/2).

Donat el següent disseny en C#. Es demana realitzar un procés de re-enginyeria i obtenir el diagrama de classes UML que es corresponga amb el disseny. Documentar les classes amb els atributs i les relacions amb el seu nom i rols (Nota: Els noms de les relacions apareixen en comentaris, els noms de rol es corresponen amb el nom dels atributs que descriuen la relació i les multiplicitats seran de 0..1 o 1..N).

```
class Persona {
    private String dni;
    private String nom;
    private String cognoms;
    private String direccio;
}

class Treballador : Persona {
    private Ocupació Empleat;
                  //Treballa_Per
 }

class Client : Persona {
    private List<Comanda> Realitza;  //Sol·licitar
    private List<Companyia> ServeisDemanats;  //Subministrar
}
```

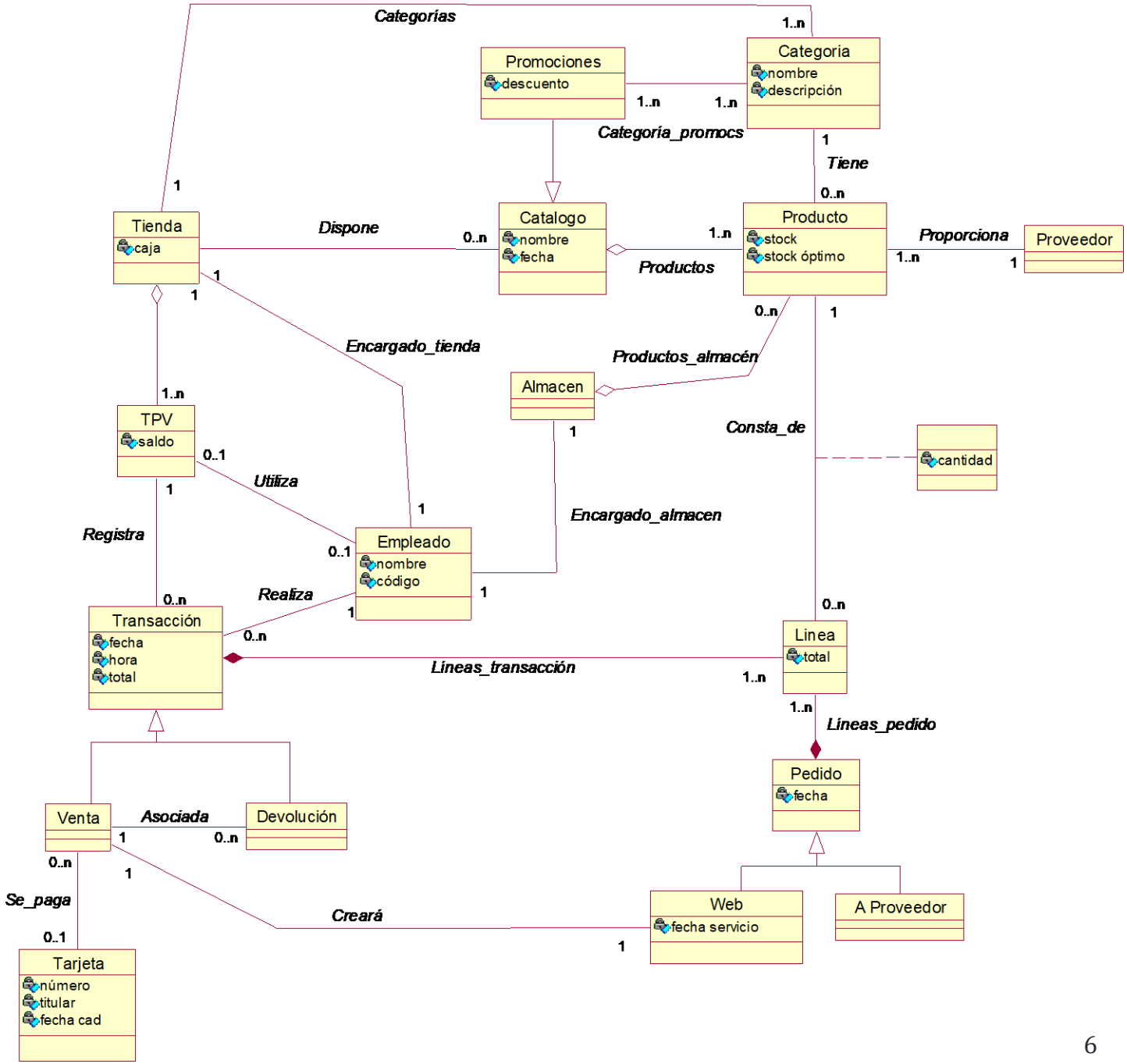
```
class Companyia {
    private String CIF;
    private String Nom;
    private List<Ocupació> empleats; //Treballa_Per
    private List<Producte> crea;    //Fabricar
    private List<Client> clients; //Subministrar
}

class Ocupació {
    private double Salari;
    private double Hores_Extra;
    private Companyia Lacompañia;   //Treballa_Per
    private Treballador ElTrabajador; //Treballa_Per
    private Ocupació Cap;    //Manar
    private List<Ocupació> Treballadors; //Manar
}
```

Problema re-enginyeria (2/2).

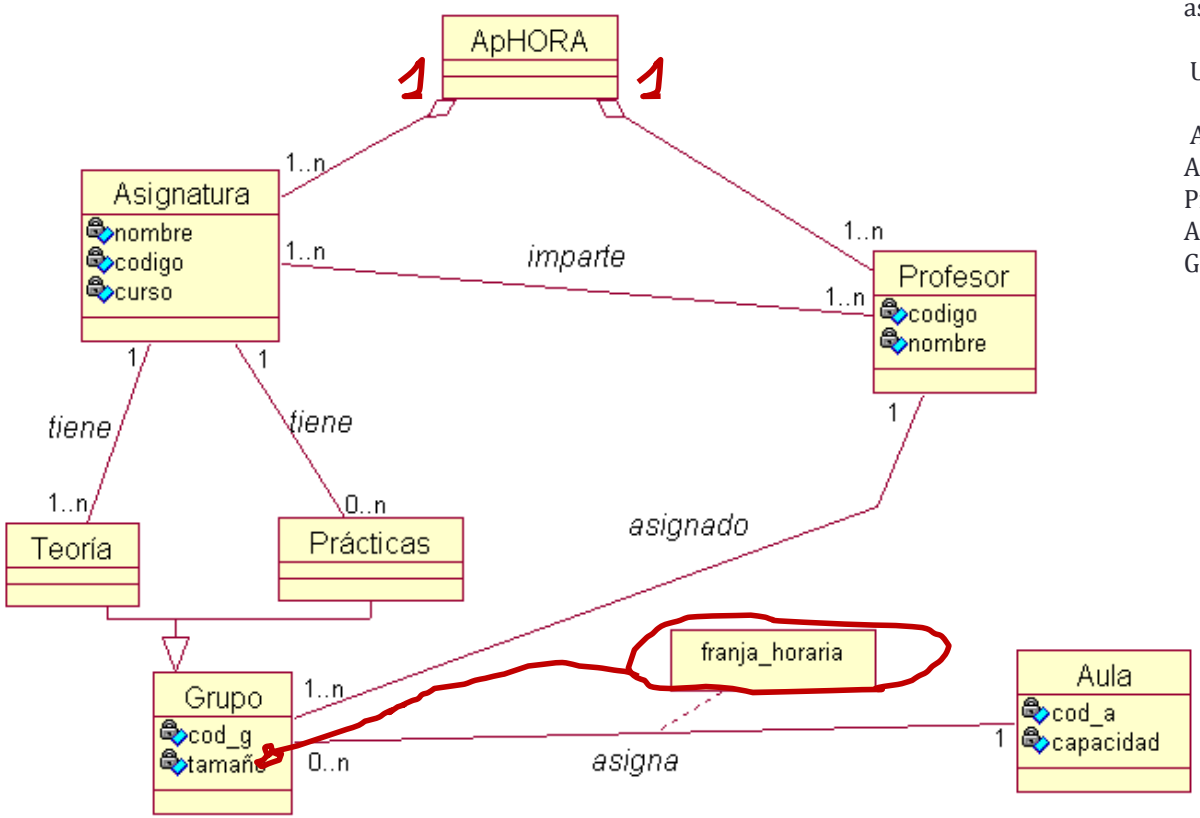
```
class Comanda {
          private int Num;
          private String Data;
          private double Preu;
          private Client ElClient;      //Sol·licitar
          private List<LineaComanda> EsComposa;    //Compondre
}

class LineaComanda {
        private int Num;
          private int Quantitat;
          private double Preu_Linea;
          private Producte Demana;    //Demanar
}

class Producte {
          private int Codi;
          private String Descripcio;
          private double Preu_Unitat;
          private Companyia Fabricat;    //Fabricar
}
```

Exercici

Fer el disseny en C# del diagrama de classes



Butlletí – Exercici 1 (ApHora)



// Grupo es classe abstracta

- Disseny d'Objectes en C#
  - Falta multiplicitat
  - Atribut d'enllaç
- Constructors (consideracions)
  - Ordre de construcció

// Relacions amb cardinalitat màxima 1 en els dos sentits -> Relaxar un sentit (considerar que siga 0 i després assegurar en el codi del main el 1.)

Una possible solució (hi ha altres vàlides)

ApHora
Aula
Professor
Assignatura
Grup

Butlletí – Exercici 1 (ApHora)

```
public class ApHora
  {
     private ICollection<Profesor> profesores;
     private ICollection<Asignatura> asignaturas;
    public ApHora()
     {
        this.profesores = new List<Profesor>();
        this.asignaturas = new List<Asignatura>();
     }
}
```

```
 public class Aula
  {
     private int cod_a;
     private int capacidad;
     private ICollection<Grupo> grupos;
     public Aula(int cod_a, int capacidad)
     {
        this.cod_a = cod_a;
        this.capacidad = capacidad;
        grupos = new List<Grupo>();
     }
```

```
public class Profesor
  {
    private int codigo;
    private string nombre;
    private ApHora enApHora;
    private ICollection<Asignatura> asignaturas;
    private ICollection<Grupo> grupos;
    public Profesor(int codigo, string nombre,
                    ApHora apHora)
    {
       this.codigo = codigo;
       this.nombre = nombre;
       this.enApHora = apHora;
       this.asignaturas = new List<Asignatura>();
       this.grupos = new List<Grupo>();
    }
```

Butlletí – Exercici 1 (ApHora)

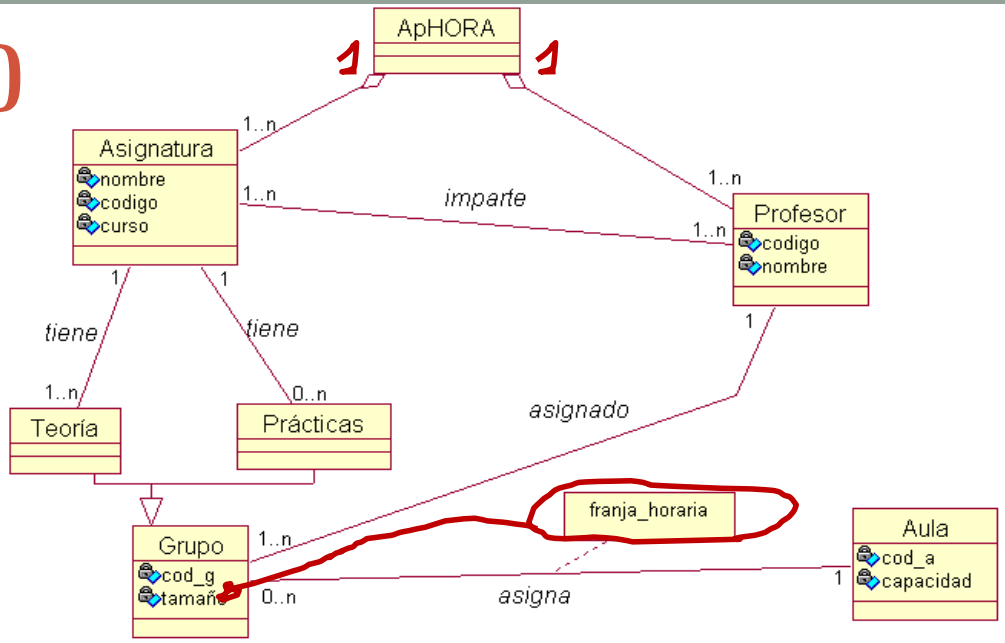
```
public class Asignatura
  {
    private string nombre;
    private int codigo;
    private string curso;
    private ApHora enApHora;
    private ICollection<Profesor> profesores;
    private ICollection<Teoria> gruposTeoria;
    private ICollection<Practica> gruposPracticas;

    public Asignatura(String nombre, int codigo, String curso, Profesor profesor,  ApHora apHora)
    {
      this.nombre = nombre;
      this.codigo = codigo;
      this.curso = curso;
      this.enApHora = apHora;
      this.profesores = new List<Profesor>();
      this.profesores.Add(profesor);
      this.gruposTeoria = new List<Teoria>();
      this.gruposPracticas = new List<Practica>();

    }
```

Butlletí – Exercici 1 (ApHora)

```
public abstract class Grupo
  {
    private int cod_g;
    private int tamanyo;
    private DateTime hora_desde;
    private DateTime hora_hasta;
    private Aula aula;
    private Profesor profesor;
    public Grupo(int cod_g, int tamanyo,
DateTime hora_desde, DateTime hora_hasta, Aula
aula, Profesor profesor)
    {
      this.cod_g = cod_g;
      this.tamanyo = tamanyo;
      this.hora_desde = hora_desde;
      this.hora_hasta = hora_hasta;
      this.aula = aula;
      this.profesor = profesor;
    }
  }
```

```
public class Teoria : Grupo
  {
    private Asignatura asignaturaT;

    public Teoria(int cod_g, int tamanyo, DateTime hora_desde,
DateTime hora_hasta, Aula aula,
      Profesor profesor, Asignatura asignatura) : base(cod_g,
tamanyo, hora_desde, hora_hasta, aula, profesor)
    {

      asignaturaT= asignatura;

    }
  }

  public class Practica : Grupo
  {
    private Asignatura asignaturaP;

    public Practica(int cod_g, int tamanyo, DateTime hora_desde,
DateTime hora_hasta, Aula aula,
      Profesor profesor, Asignatura asignatura) : base(cod_g,
tamanyo, hora_desde, hora_hasta, aula, profesor)
    {

      asignaturaP= asignatura;

    }
  }
```

Butlletí – Exercici 1 (ApHora)

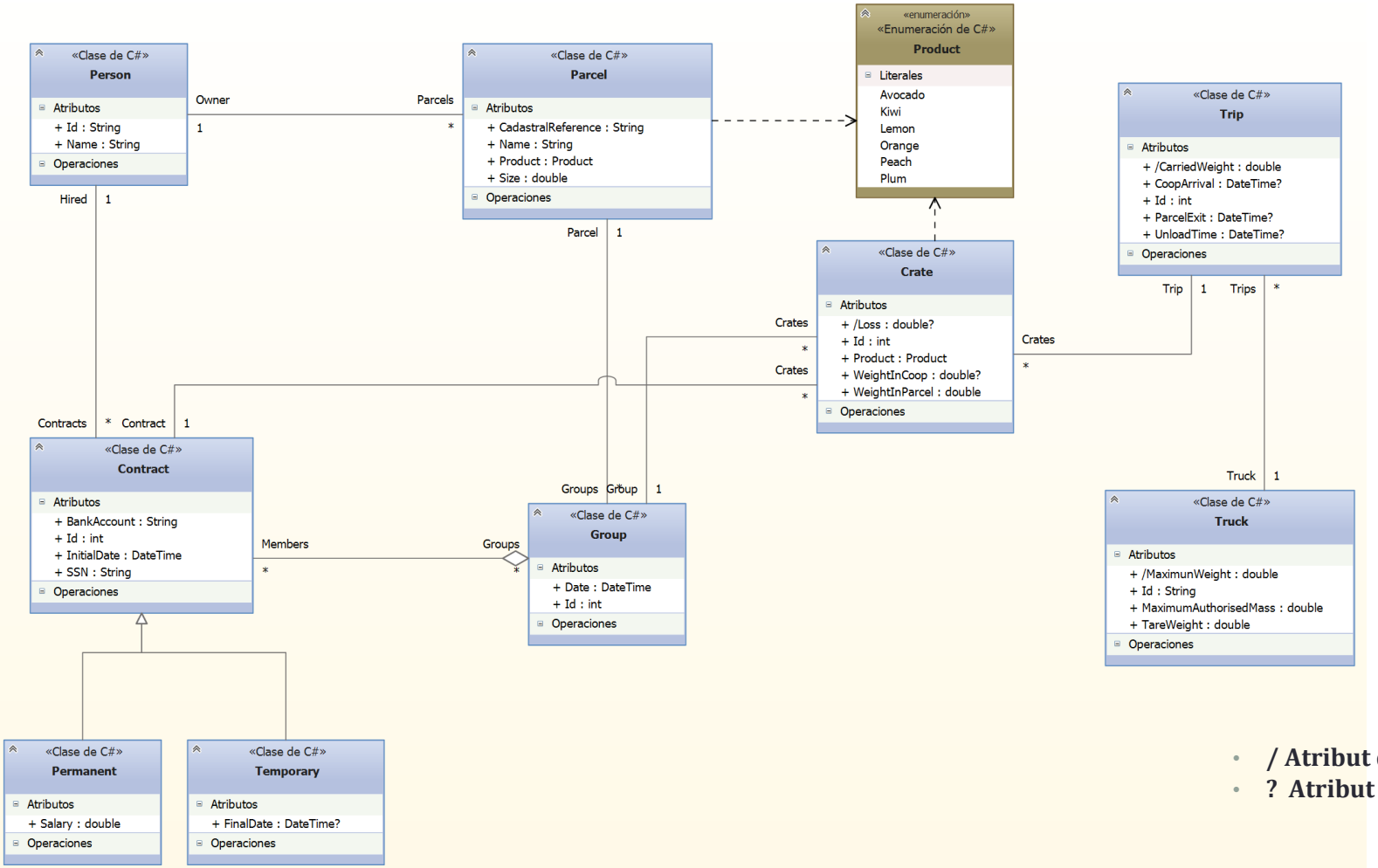



```
class Program
{
   static void Main(string[] args)
   {
     ApHora miApHora = new ApHora();

      Aula aula11 = new Aula(11,75);

      Profesor mc = new Profesor(1, “mc", miApHora);
      miApHora.AddProfesores(mc);

      Asignatura isw = new Asignatura("ISW", 1, "tercero", mc, miApHora);
      mc.AddAsignatura (isw);
      miApHora.AddAsignaturas(isw);

     Teoria teo_isw = new Teoria(11, 55, new DateTime(2021, 9, 14, 11, 30, 0), new DateTime(2021, 9, 14, 13, 0, 0), aula11, mc, isw);
      isw.AddTeoria(teo_isw);
      mc.AddGrupos(teo_isw);
     aula11.AddGrupo(teo_isw); //dejamos consistente el modelo, la navegación de la asociación es bidireccional
```

// en una inicialització mínima, no necessitem tenir grups de pràctiques . Si ens demanen crear una instància de cada classe, llavors s'invocaria el constructor de Pràctiques i es deixaria consistent el model (respecte a la navegabilitat), de forma similar a Teoria.

Cas d'Estudi Pràctiques

- Capa Lògica : Generar el codi per al cas d'estudi seguin les pautes de disseny vistes, a partir del diagrama de classes de disseny (solució de disseny)



- **/ Atribut derivat**
- **? Atribut que pot ser null**