Universidad Tecnológica Nacional
FACULTAD REGIONAL CORDOBA

PARADIGMAS DE PROGRAMACION

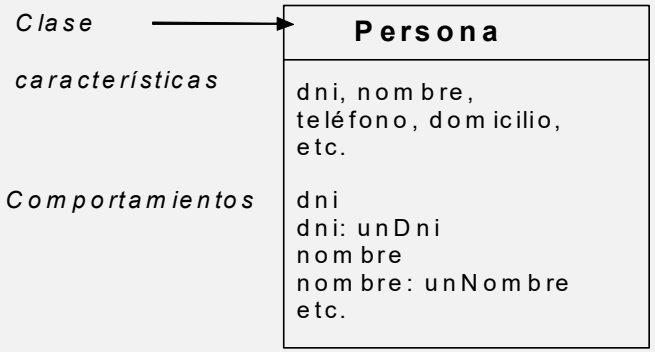
Unidad III
Paradigma Orientado a Objetos
Composición

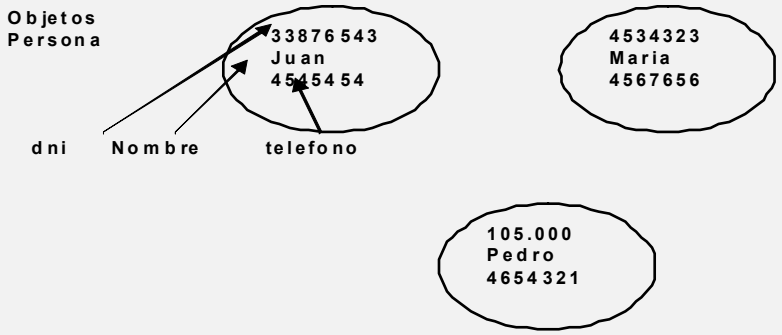
CONTENIDOS ABORDADOS

- Relaciones entre clases.
- Composición.
- Caso de estudio.

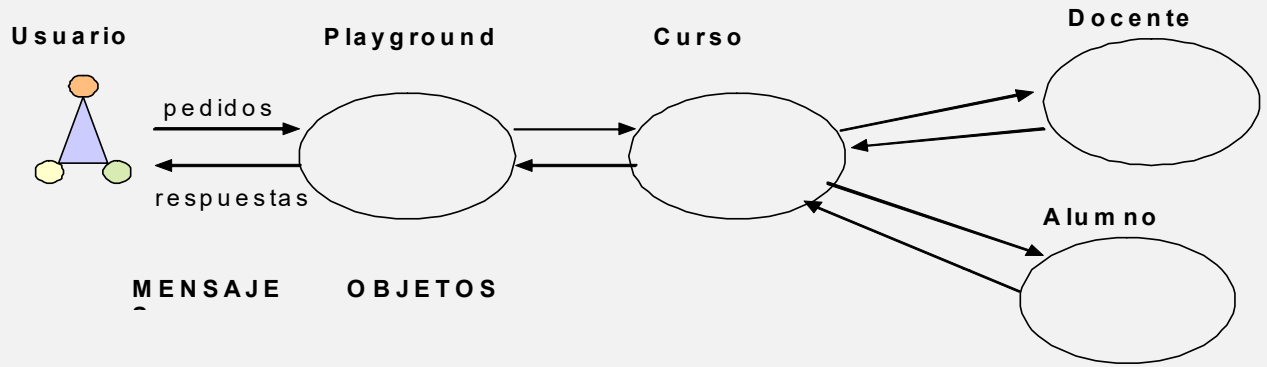
Clases

- Es un molde o modelo para construir objetos.
- En lugar de definir cada objeto por separado, defino una clase con las características que serán comunes a los objetos, y luego voy a crear los objetos a partir de esta clase.





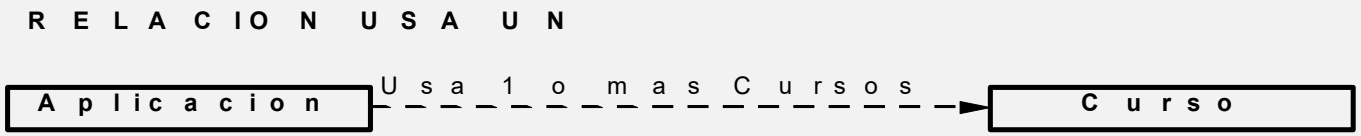
Relaciones entre clases



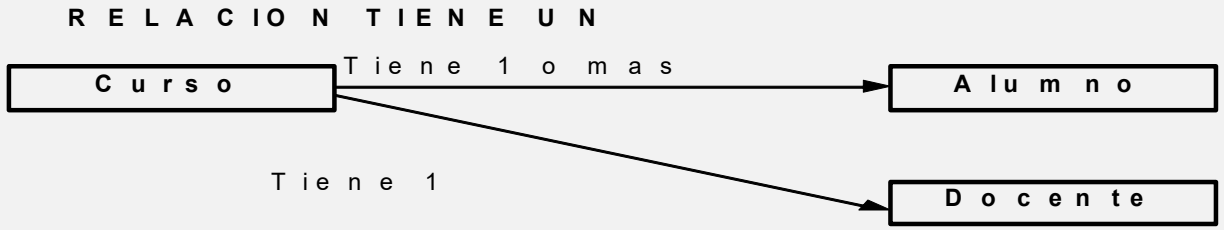
- Asociación
- Composición
- Especialización

Relaciones entre Clases

- Asociación (conexión entre clases)



- Agregación / Composición (relaciones de pertenencia)

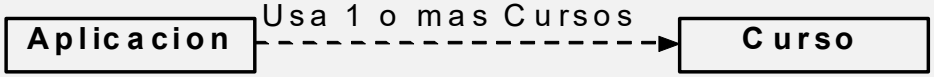


- Generalización/especialización (relaciones de herencia)

Relaciones entre Clases

- **Asociación:**
  - Permite asociar objetos que colaboran entre si.
  - No es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.
  - Ambos objetos son independientes entre sí.
  - Para validar la asociación, la frase *"Usa un"*, debe tener sentido:
    - **El usuario** *usa* **una aplicación**
    - **El cliente** *usa* **tarjeta de crédito**.

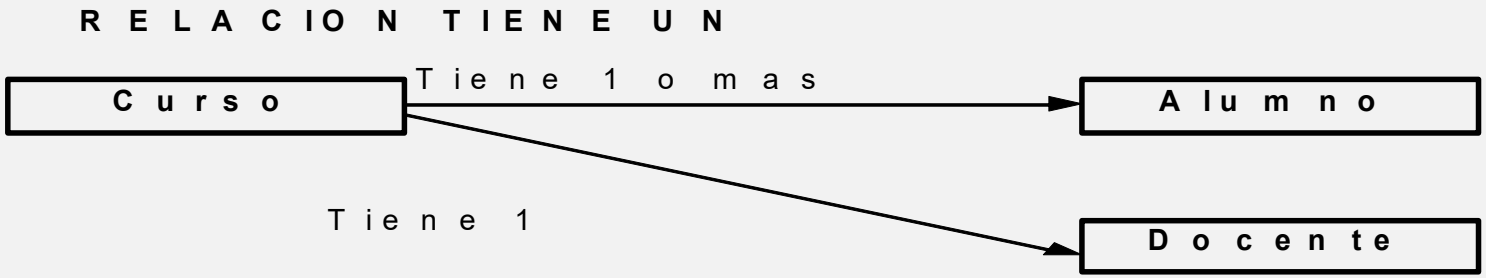
RELACION USA UN




```
"Calcula el total de los valores de una colección"
 |unaColeccion suma|

 unaColeccion := OrderedCollection new.
 unaColeccion add:1; add:2; add:3; add:4; add:5.

 suma := 0.
 unaColeccion do: [:unValor| suma := suma + unValor].

 Transcript show: 'La sumatoria: ', suma asString.
```

Relaciones entre Clases

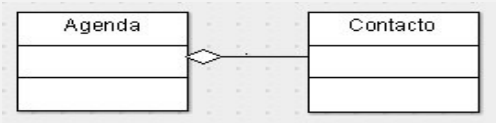
- **Composición:**
  - Es un tipo de relación **dependiente** en dónde un objeto más complejo es conformado por objetos más pequeños.
  - En esta situación, la frase “Tiene un”, debe tener sentido:
    - **El auto** tiene **ruedas.**
    - **El cliente** tiene **cuit.**

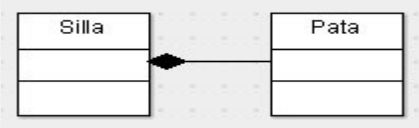


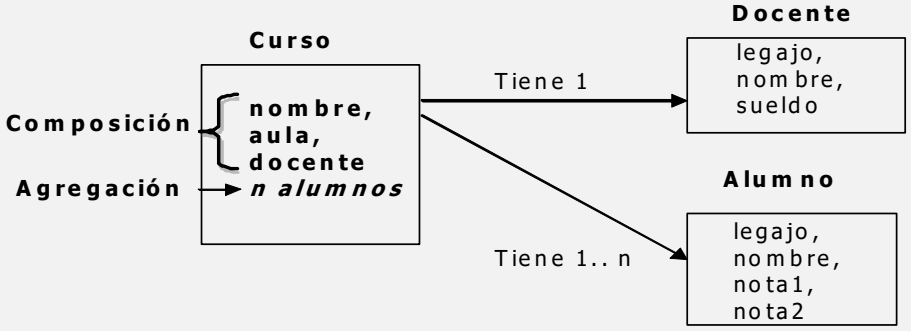
Relaciones entre Clases

Composición - Tipos de especializaciones:

- **Agregación:** Implica una composición débil, si una clase se compone de otras y quitamos alguna de ellas, entonces la primera seguirá funcionando normalmente.
- **Composición:** Es una forma fuerte de composición, donde la vida de la clase contenida debe coincidir con la vida de la clase contenedor. Los componentes constituyen una parte del objeto compuesto.

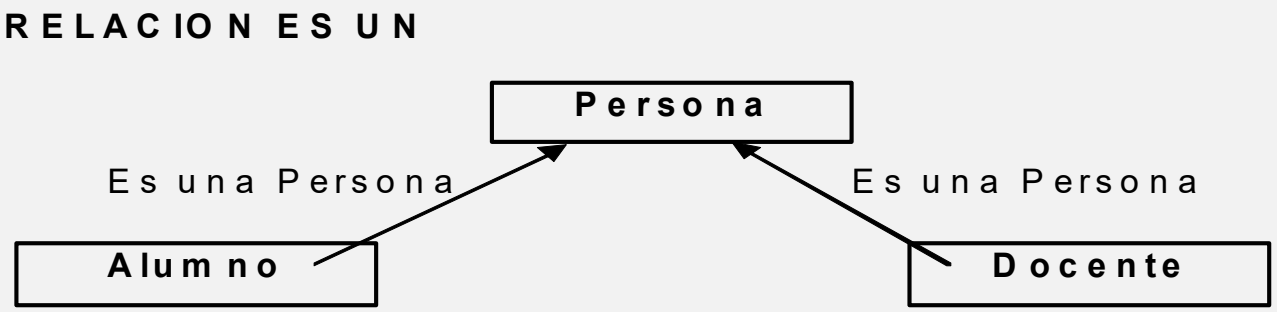






Relaciones entre Clases

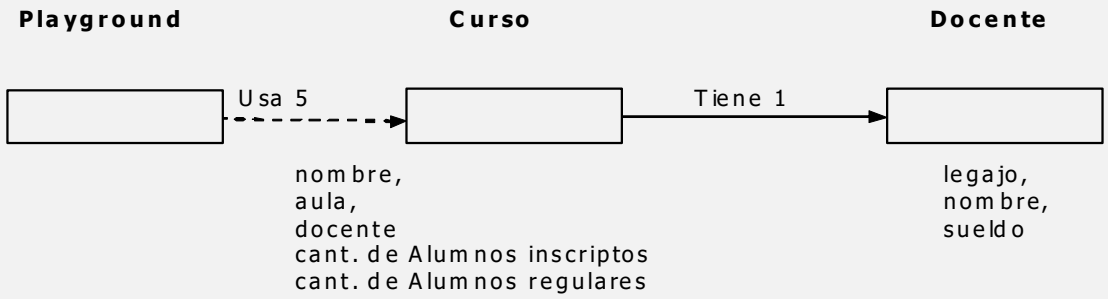
- **Generalización/Especialización**
  - También llamada **Herencia.**
  - Es un tipo de relación entre clases que comparten su estructura y el comportamiento.

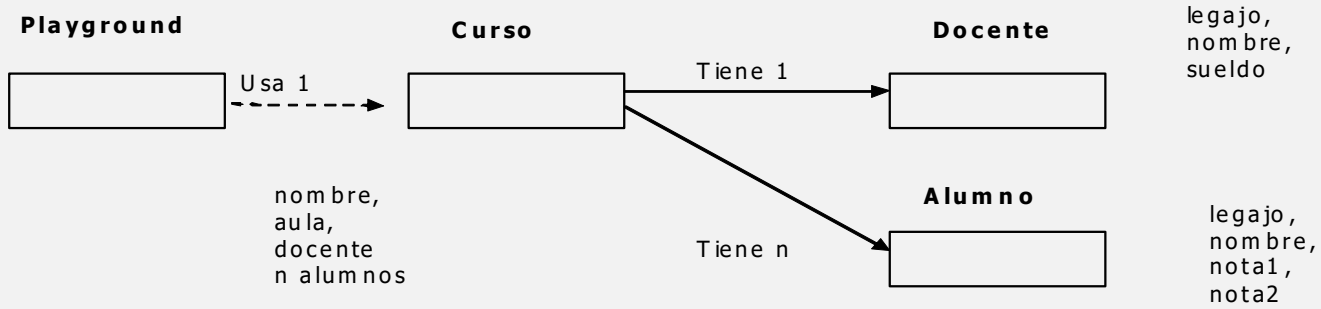


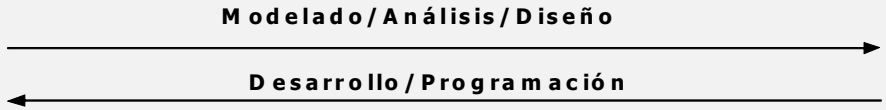
Reutilización de Clases

- Hay dos mecanismos en POO para aplicar reutilización, es decir, para construir clases utilizando otras clases:
  - **Composición:** Una clase posee objetos de otras clases (relación *tiene un*). Se puede reutilizar los atributos y métodos de otras clases, a través de la invocación de los mensajes correspondientes.
  - **Herencia:** Se pueden crear clases nuevas a partir de clases preexistentes (relación *es un*). Se puede reutilizar los atributos y métodos de otras clases como si fueran propios.

Ejemplos de composición

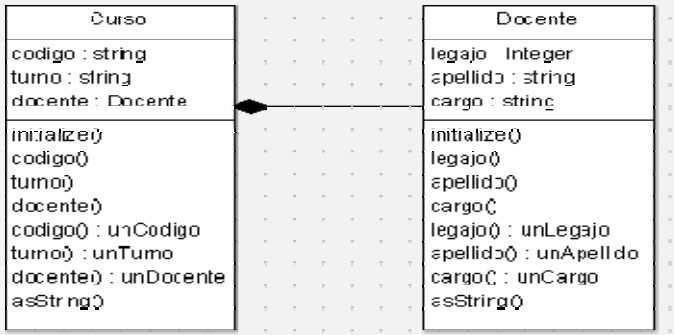






Implementación de composición

```
Object subclass: #Docente
    instanceVariableNames: 'legajo apellido
    cargo'
    classVariableNames: ''
    package: 'ComposicionPPR'
```

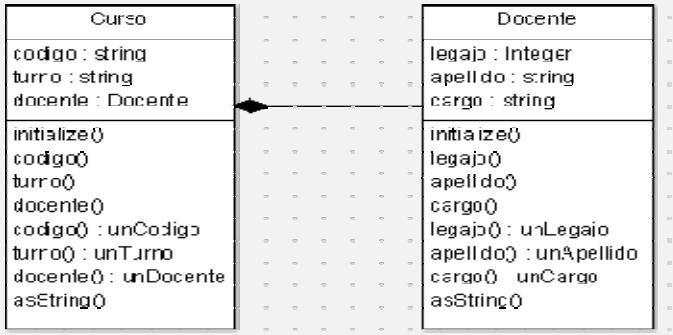



```
Object subclass: #Curso
    instanceVariableNames: 'codigo  turno docente'
    classVariableNames: ''
    package: 'ComposicionPPR'
```

Implementación de composición

- Implementación de ciertos métodos en clase Curso.

```
initialize
    codigo := ''
    turno := ''
    docente := nil.
docente
   ^docente.
docente:unDocente
    docente := unDocente.
asString
  |datos|
 datos:='Codigo curso:', self codigo asString, 'Turno: ', self turno
   asString, 'Docente: ', self docente asString.
^datos.
```

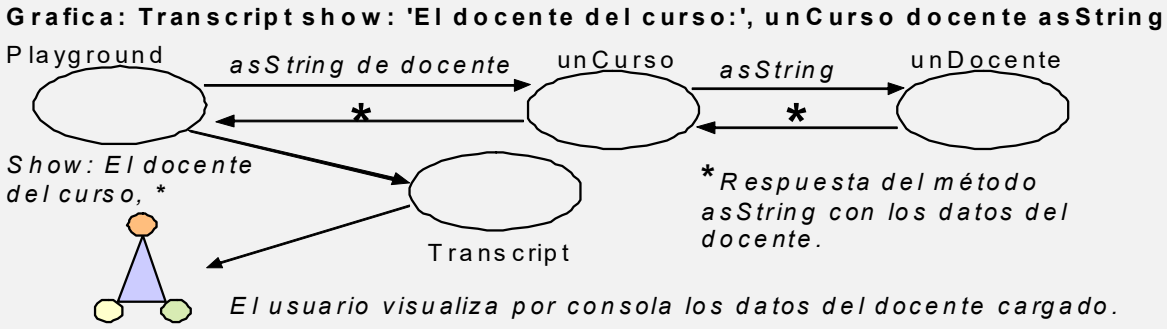


Se invoca el método **asString** de la clase Docente.

Implementación de composición

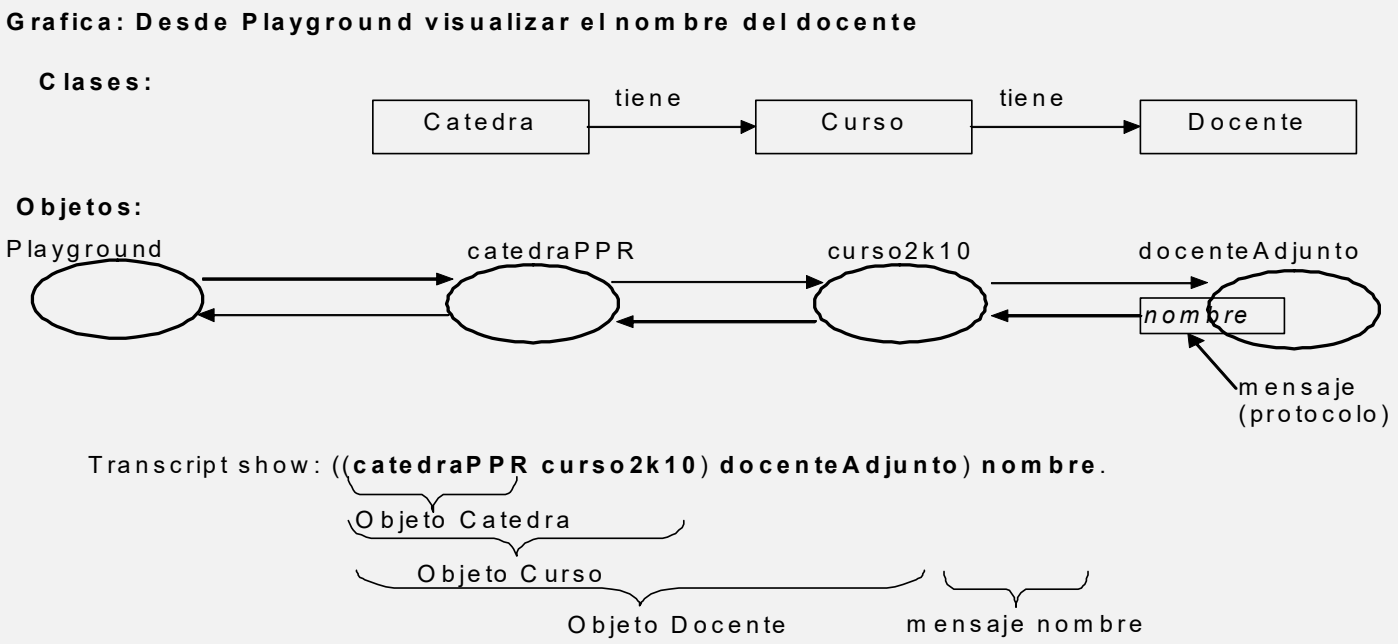
- Esquema de creación de objetos usando Composición.

```
|unDocente unCurso|
"creo objeto Docente para asignárselo a objeto curso"
unDocente := Docente new initialize.
unDocente legajo:12; apellido:'Pérez'; cargo:'JTP'.

"creo una instancia de Curso"
unCurso := Curso new initialize.
unCurso codigo:'2K3'; turno:'Mañana'; docente:unDocente.
Transcript show: 'El docente del curso:', unCurso docente asString.
```



Implementación de composición

- Esquema de invocación de objetos usando Composición.



Caso de estudio

1. Implementar en Smalltalk las clases Docente y Curso vistas anteriormente y crear varios objetos de tipo Curso y mostrar sus datos

2. Implementar la clase Alumno y adecuar la clase Curso según el diseño siguiente, las condiciones se enuncian en el documento adjunto CasoEstudioClase3.pdf.