Universidad Tecnológica Nacional
FACULTAD REGIONAL CORDOBA

PARADIGMAS DE PROGRAMACION

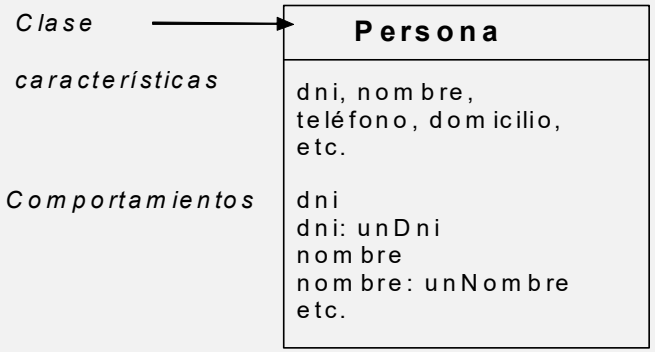
Unidad III
Paradigma Orientado a Objetos
Herencia

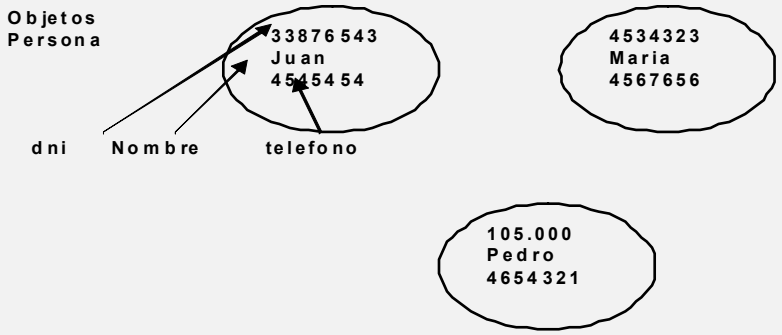
CONTENIDOS ABORDADOS

- Relaciones entre clases.
- Herencia.
- Casos de estudio.

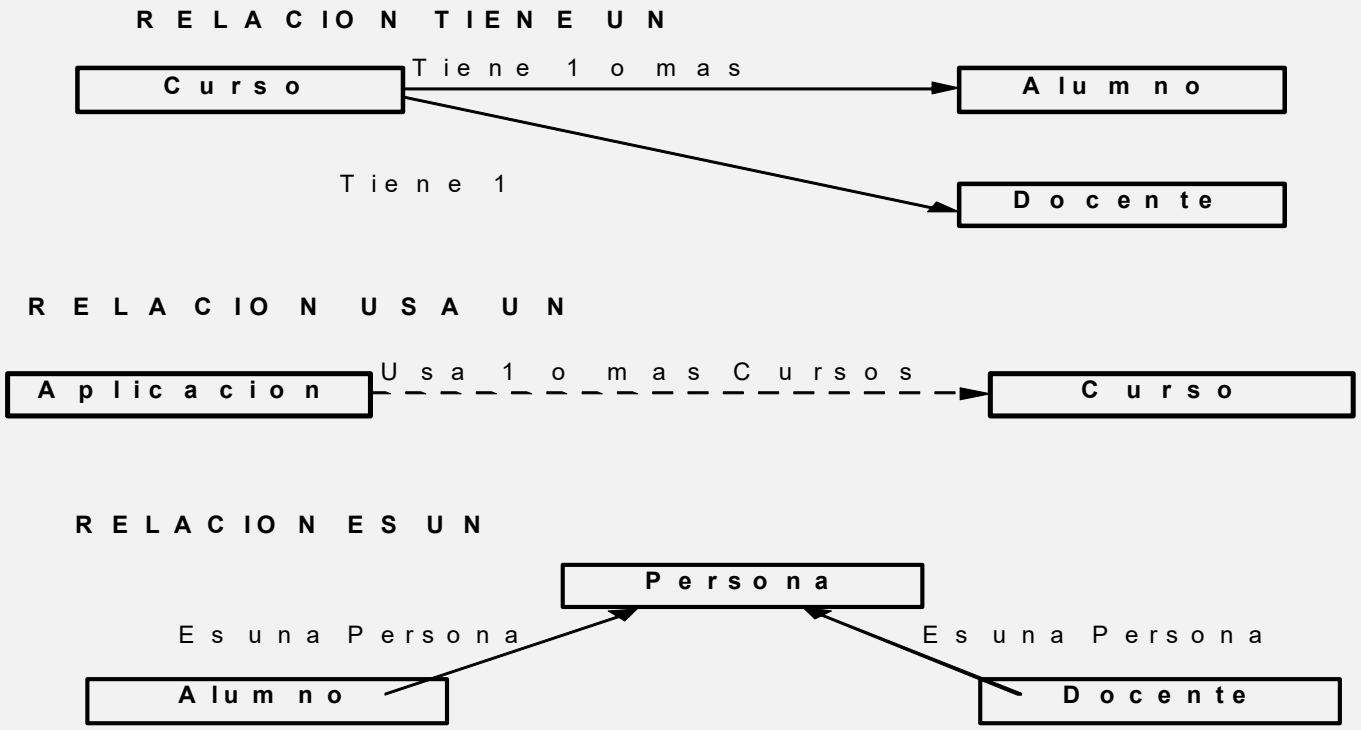
Clases

- 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.



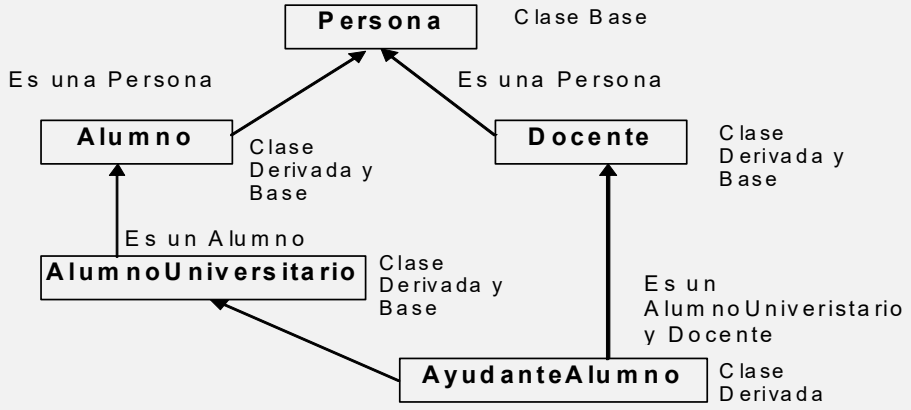


Relaciones entre clases



Herencia

- La herencia es un mecanismo que permite la definición de una clase a partir de la definición de otra ya existente.
- Es una característica clave en los sistemas orientados a objetos.
- Permite la reusabilidad.
- Mecanismo por el cual los objetos comparten conocimiento común, atributos y comportamientos.



Herencia: Estrategias

- Las estrategias para implementarla, pueden ser:
  - **Sistemas basados en clases**: Mediante una clase (molde, plantilla) es posible definir la estructura y el comportamiento de un conjunto de objetos. Lenguajes: Smalltalk, Haskell, C++, Java, entre otros.
  - **Prototipos**: La herencia se obtiene a través de la clonación de objetos ya existentes, que sirven de prototipos, extendiendo sus funcionalidades. Lenguajes:: Self, JavaScript, entre otros.

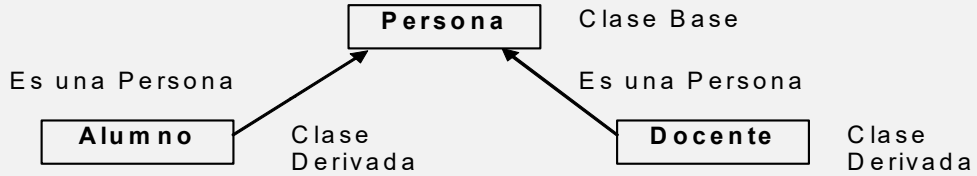
Herencia: Estrategias

Comparación de modelos

Modelo basado en clases	Modelo basado en prototipos
Existe el concepto de clase e instancia u objeto.	Existe el concepto de objetos considerados como clase e instancia.
Creación de objetos mediante el uso de métodos constructores definidos en las clases	Crea un objeto por medio de funciones constructoras, usando clonación.
Hay una distinción entre la estructura y el comportamiento.	Hay una distinción del estado.
No se pueden añadir propiedades a los objetos en tiempo de ejecución.	Se puede añadir o eliminar propiedades en tiempo de ejecución.

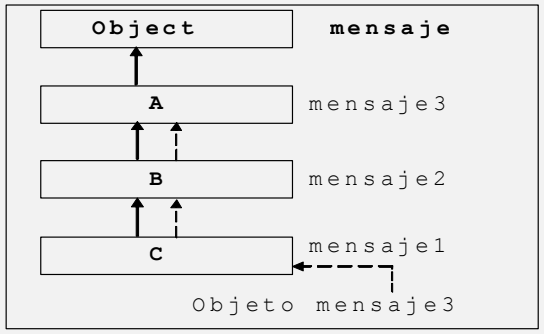
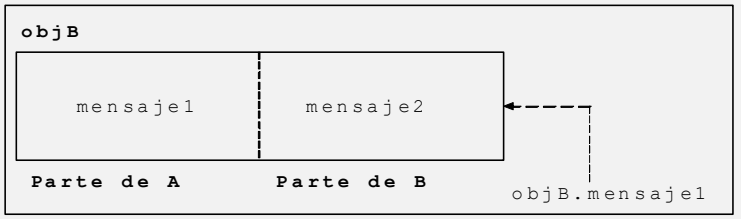
Herencia: Basada en Clases

- Formada por una clase llamada base, padre, ancestro, etc. y una clase llamada derivada, subclase, hija, descendiente, etc. que hereda las características y comportamientos de la clase base



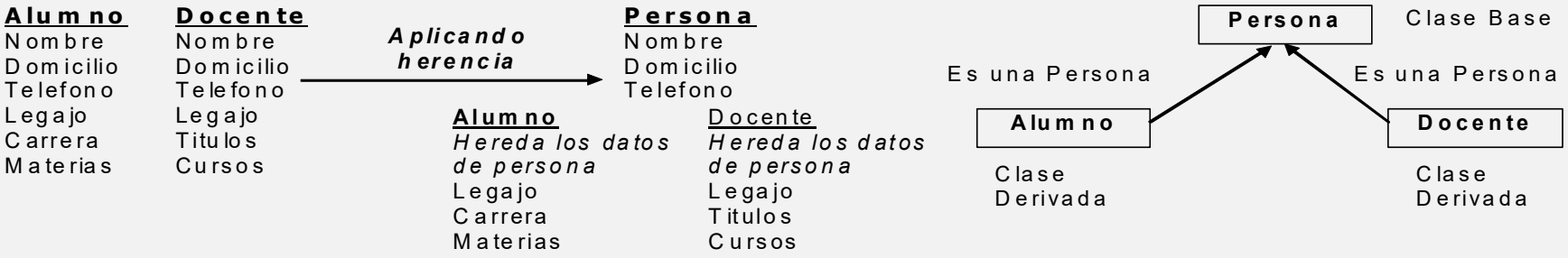
- Las clases se organizan mediante ***jerarquías de clases.***

Herencia: Modelos de implementación

Herencia por delegación	Herencia por Concatenación
El objeto tiene uno o más atributos parent, de forma que cuando no puede responder a un mensaje, le reenvía éste a su padre. Existe una cadena de objetos apuntando a sus padres, hasta llegar a un objeto padre de todos. 	El objeto está compuesto por las partes que define cada una de las clases de las que hereda. Todos los atributos y métodos heredados están disponibles en el mismo objeto. 

Herencia en Smalltalk

- Utiliza el mecanismo de herencia por delegación.
- Las subclases heredan atributos y métodos de sus clases base o superclases.
- La subclase puede añadir más variables de instancia pero no puede eliminar ninguno de los atributos heredados.
- En este curso se va a implementar solamente herencia simple (una subclase puede tener solo una clase base).

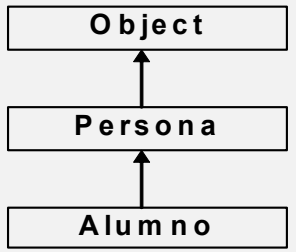


Herencia en Smalltalk

- Sintaxis para crear una clase jerarquía:

```
#clase base Persona
Object subclass: #Persona
    instanceVariableNames: 'dni nombre telefono'
    classVariableNames: ''
    category: 'HerenciaPPR'


#clase derivada Alumno
Persona subclass: #Alumno
    instanceVariableNames: 'legajo promedio'
    classVariableNames: ''
    category: 'HerenciaPPR'
```



Herencia en Smalltalk

Super

- El uso de **super** provoca que la búsqueda del método comience en la superclase del objeto receptor.
- Cuando **super** es encontrado en la ejecución de un programa, Smalltalk busca el método en la superclase del receptor.
- Ejemplo:

```
"invoca a un mensaje de la clase base"
super unMensaje.
"invoca a un mensaje de la propia clase"
self unMensaje
```

Herencia en Smalltalk

Herencia de Variables

- La jerarquía del lenguaje Smalltalk, ha sido diseñada para que las subclases hereden las variables de sus superclases. Las subclases también pueden poseer variables propias.
- Las variables de instancia están definidas en la definición de la clase. Los datos de la instancia se mantienen en un área de datos creada por Smalltalk.
- Las variables de clase también permiten compartir la información a través de todas las subclases de la clase en que fueron declaradas.

Herencia en Smalltalk

Herencia de Variables

- **Variables de Instancia**: Cada subclase tiene su propia copia (estado) de variables, tanto las propias como las heredadas.
- Por ejemplo: el objeto alumno tiene las **variables de instancia**: dni, nombre, telefono, legajo y promedio. Estas variables incluyen las variables de instancia heredadas de Persona.

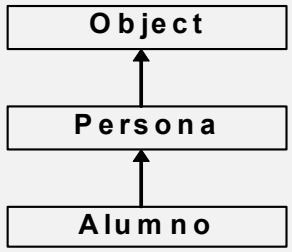
Herencia en Smalltalk

Inicialización de atributos en una clase derivada

- En el método de inicialización de la clase inicializa en primer lugar los atributos heredados de su superclase y seguidamente las variables de instancia propias.
- Ejemplo:

```
initialize
    "Inicializa atributos de la base"
    super initialize.

    "Inicializa atributos propios"
    atributoPropio1 := 0.
    atributoPropio2 := ' '.
```



Herencia en Smalltalk

Herencia de Métodos

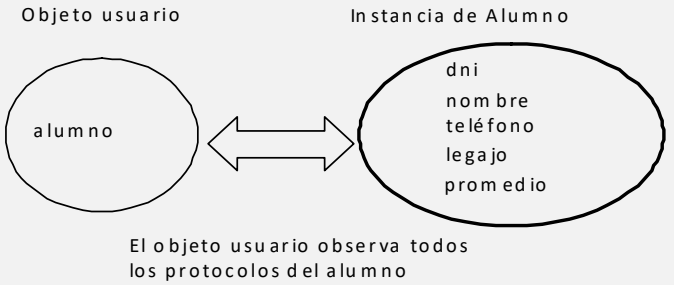
- Permite a una clase modificar su comportamiento respecto de su superclase. Puede ser por:
- **Agregación de métodos:** Se agregan nuevos nombres de métodos en la definición de la clase. Todo objeto de la subclase soporta los métodos de su superclase, más los nuevos métodos.
- **Redefinición de métodos:** Se redefine (vuelve a definir) algún método con un nombre existente en la superclase, con el objetivo de proveer una implementación diferente, por ejemplo: el método asString.

  Se declaran con la misma signatura (nombre y parámetros).

  En la invocación de métodos, si existen dos o más métodos con el mismo nombre o signatura, uno en la subclase y otros en las superclases, se ejecutará siempre el de la subclase.

Herencia en Smalltalk

Redefinición de Métodos



- Ejemplo en la clase Alumno:

```
asString
  | res |
  res := super asString,
          ' ,Legajo: ', self legajo asString,
    ' , Promedio: ', self promedio asString.

  ^res.
```

Datos de Persona

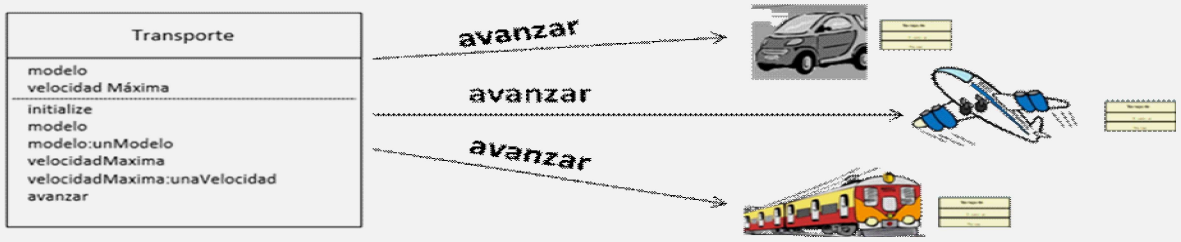
Datos de Alumno

```
asString
 ^ 'Dni: ', super dni asString, ' ,Nombre: ', self nombre asString,
' ,Telefono: ', self telefono asString, ' ,Legajo: ', self legajo
asString, ' , Promedio: ', self promedio asString.
```

Herencia en Smalltalk

Clases Abstractas

- Son clases genéricas que sirven para agrupar clases del mismo género.
- Definen comportamientos que se implementarán en las subclases.
- No se pueden instanciar, es decir no se pueden crear objetos a partir de ellas, ya que representan conceptos tan generales que no se puede definir como será la implementación de los mismos.



Herencia en Smalltalk

- Método abstracto es un método que no tiene implementación.
- Smalltalk posee mecanismos formales para implementarlos, pero el desarrollador puede hacerlo definiendo métodos sin implementación.

Clase abstracta - Método abstracto:

```
size
    ^ self subclassResponsibility.
```

ó

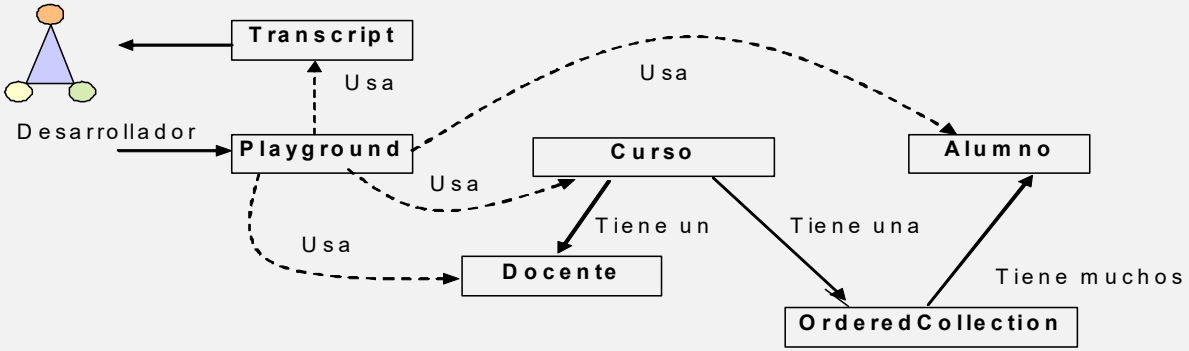
```
size
   "sin implementación"
```

Clase derivada - Método implementado:

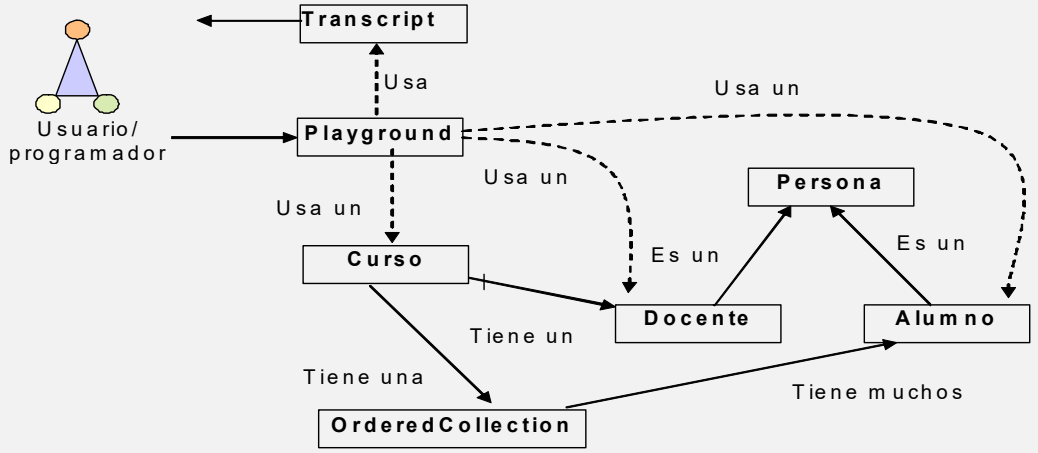
```
size
  ^ size
```

Caso de estudio

Modelo del caso de estudio de la clase anterior

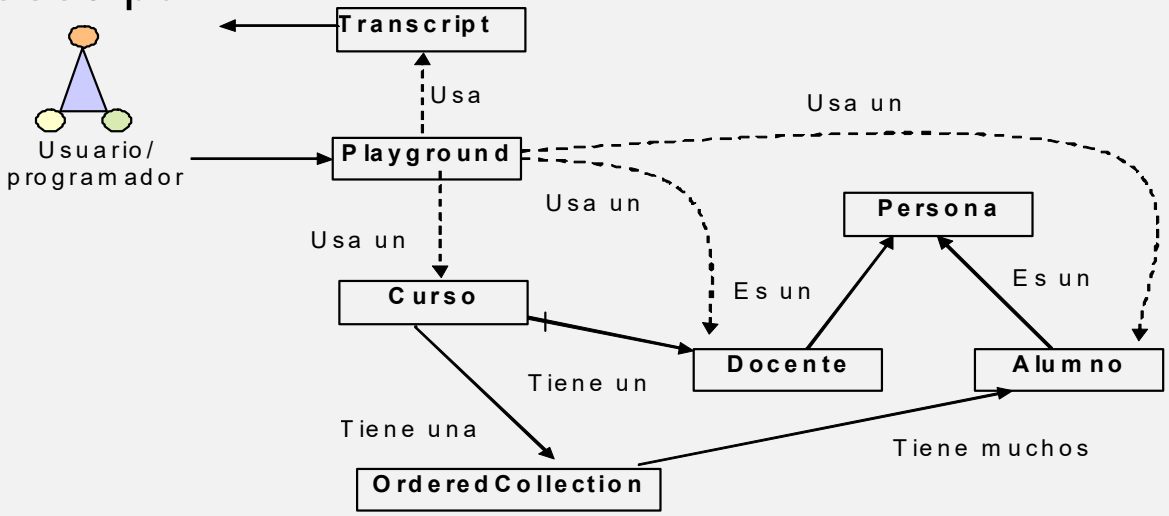


Modelo modificado utilizando herencia



Caso de estudio

1. Implementar las clases de la jerarquía descriptas en el diagrama.
2. Implementar la clase curso que utilice un docente y una colección de alumnos.
3. Las condiciones se enuncian en el documento adjunto CasoEstudioClase3.pdf.



Caso de estudio

Diseño de clases

Persona: Atributos: dni y nombre.
Métodos: Inicializador, acceso, modificadores y asString.
Docente: Es una Persona.
Atributos: legajo y sueldo.
Métodos: Inicializador, acceso, modificadores y asString
Alumno: Es una Persona.
Atributos: legajo, nota1, nota2.
Métodos: Inicializador, acceso, modificadores y asString.
Un método de control que retorne el promedio del alumno.
Curso: Atributos: nombre, docente, alumnos.
Métodos: Inicializador, acceso, modificadores y asString, además los siguientes métodos para el manejo de los alumnos: