UNIVERSIDAD DE LIMA | 60 años

DISEÑO ORIENTADO A OBJETOS USANDO UML

UNIDAD 2: DISEÑO E IMPLEMENTACIÓN

Temario

- Modelado y programación orientada a objetos.
- Diseño detallado.
- Modelos de diseño.
- Aspectos de implementación del software.
- Medidas de calidad del diseño OO.

UNIVERSIDAD DE LIMA | 60 años

Modelado y programación orientada a objetos.

A veces hay una etapa de diseño separada, y este **diseño se modela y documenta**. En otras ocasiones, un diseño está en la cabeza del programador o esbozado en una pizarra u hojas de papel.

Definiciones

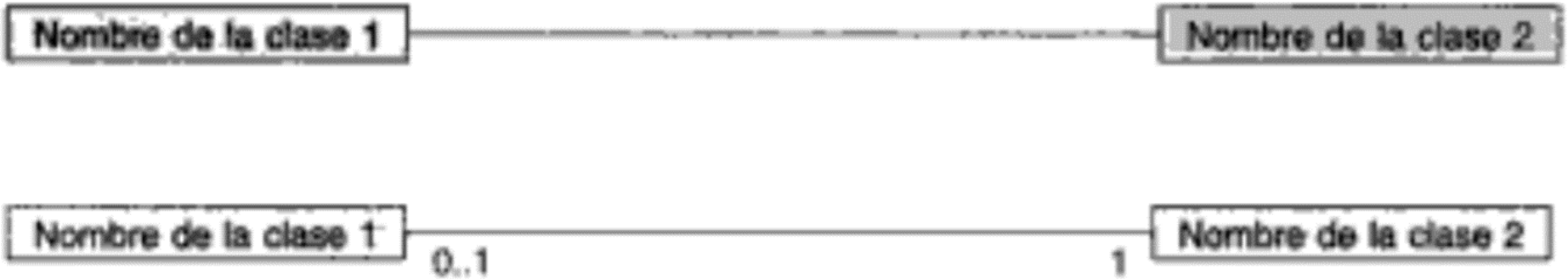
- **Objeto:**
  - Son las entidades básicas del modelo de objetos. Un objeto integra una estructura de datos (atributos) y un comportamiento (operaciones).
- **Clase:**
  - Describe un grupo de objetos con estructura y comportamiento común. Una clase se considera un “molde” a partir del cual se crean múltiples objetos.
  - Ejemplo: Una clase puede ser un molde de una cerámica.
- **Atributo:**
  - Definen la estructura de un objeto y de sus correspondientes objetos.
- **Operaciones:**
  - Son funciones o transformaciones que se aplican a todos los objetos de una clase en particular. La operación puede ser una acción ejecutada por el objeto o sobre el objeto.
- **Polimorfismo:**
  - Se define como una **misma operación** que toma diferentes formas. Una operación se considera polimórfica si esta se implementa en diferentes clases de forma distinta.

Definiciones

- **Relación:** vínculo entre objetos
- **Asociación:** vínculo entre clases de objetos
- **Tipos de asociaciones entre clases:**
    - Asociaciones de conocimientos (**Asociación estática**): Una instancia conoce de la existencia de otra instancia. Denota conocimiento entre clases durante largos periodos.
    - Asociaciones de comunicación (**Asociación dinámica**): representa el intercambio de información entre objetos. Un objeto envía y recibe eventos.
    - Las asociaciones y relaciones son bidireccionales
    - **Grado de una asociación:** Número de clases conectadas por la misma asociación.
    - **Asociaciones reflexivas:** Las asociaciones pueden ser reflexivas, y relacionan distintos objetos de una misma clase.
- **Multiplicidad: (Cardinalidad)** de una asociación especifica cuántas instancias de una clase se pueden relacionar a una sola instancia de otra clase.

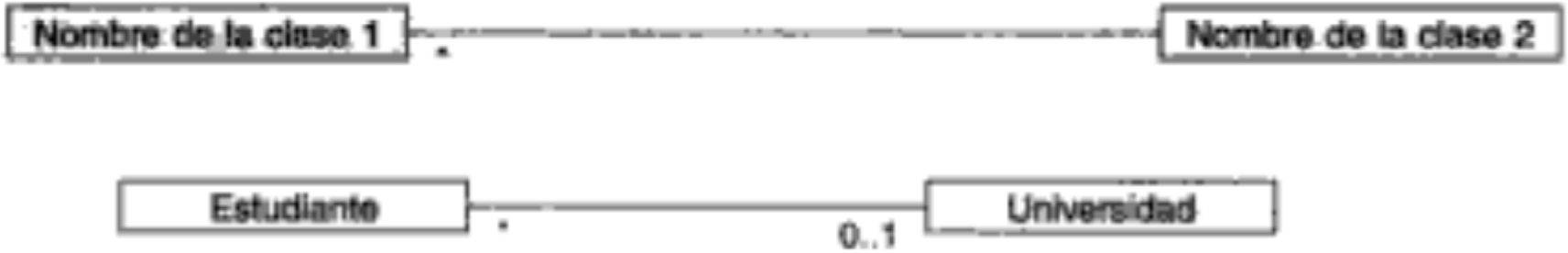
Multiplicidad: Uno a uno

Uno a uno: dos objetos se relacionan de forma exclusiva, uno con el otro.



Multiplicidad: Uno a muchos

- Uno a muchos: uno de los objetos está relacionado a muchos otros objetos.

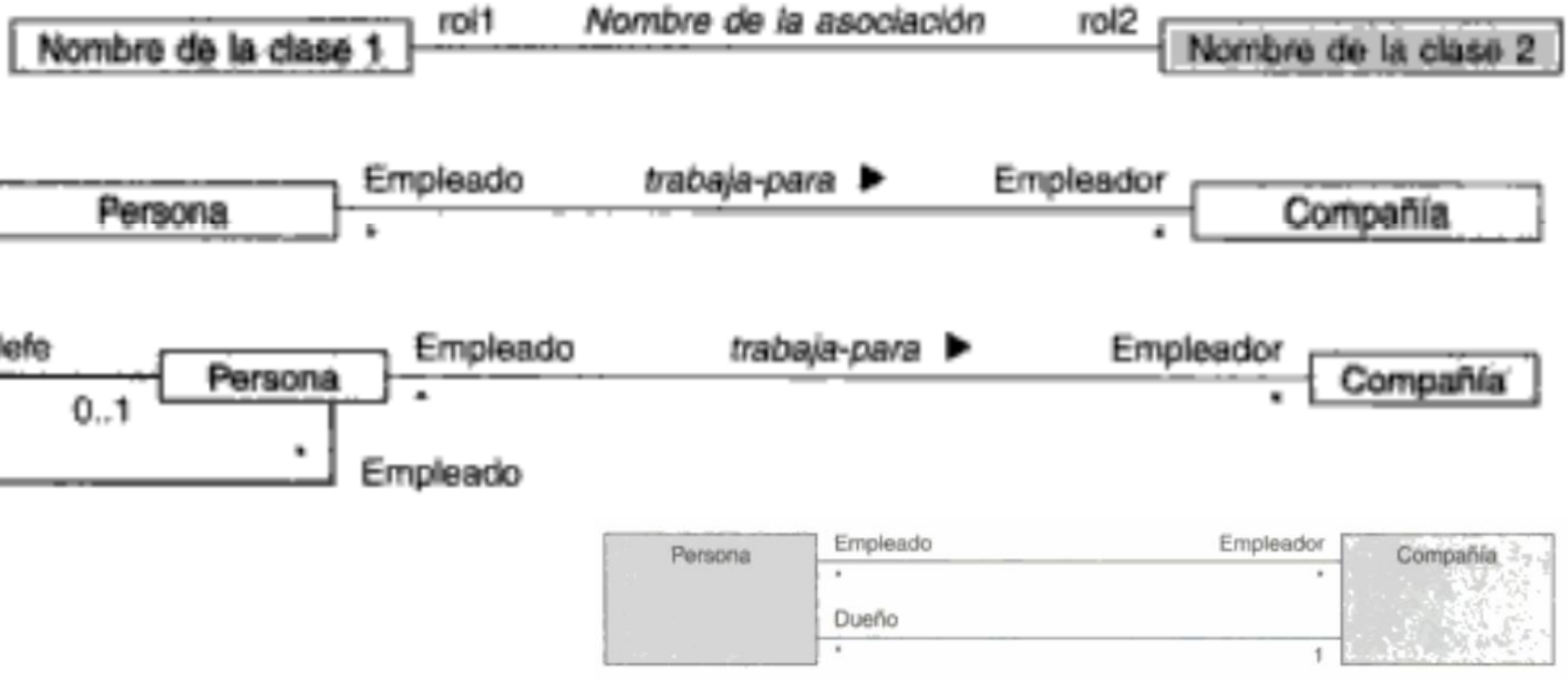


Multiplicidad: Muchos a muchos

- Muchos a muchos: cada objeto de cada clase puede estar vinculado a muchos otros objetos.

Rol

- Describe el papel que juega cada extremo de una asociación.



Ensamblados: agregación y composición

- Los **ensamblados,** en particular la **agregación** y la **composición** son formas especiales de asociación entre un todo y sus partes, en donde el **ensamblado** está compuesto por sus componentes. El ensamblado es el **objeto** central, y la estructura completa se describe como una **jerarquía de contenido**.
- Un ensamblado puede componerse de varias partes, donde cada relación *parte-todo* se considera una relación separada. En el caso de la *agregación*, las partes del ensamblado pueden aparecer en múltiples ensamblados. En el caso de la *composición*, las partes del ensamblado no pueden ser compartidas entre ensamblados.

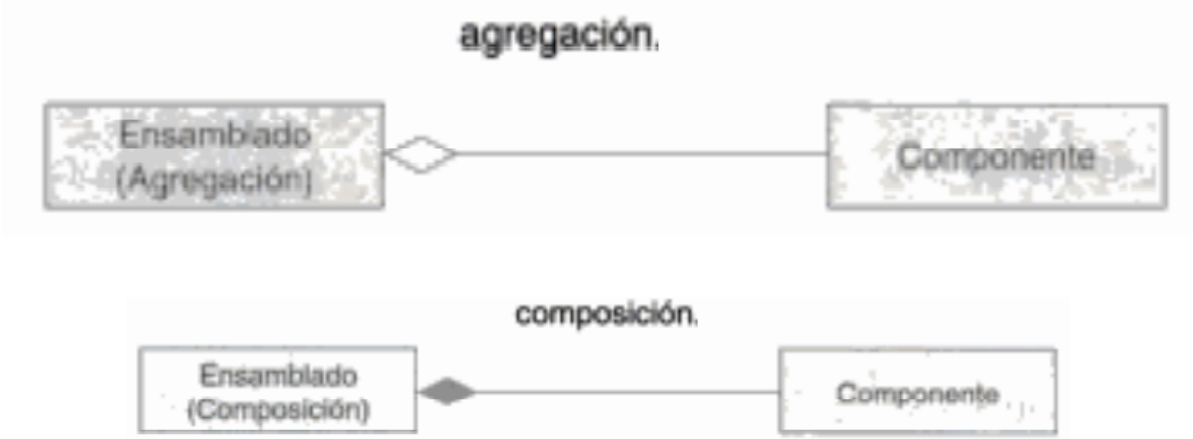
Ensamblados: agregación y composición

Ejemplo: Una *Red de computadoras* se puede considerar un ensamblado, donde las *Computadoras* son sus componentes. Éste también es un ejemplo de agregación, ya que las *Computadoras* pueden ser partes de múltiples *Redes de computadoras* a la vez. Además, las *Computadoras* pueden existir independientemente de la existencia de la *Red de computadoras*.

Ejemplo: Un *Automóvil* también se puede considerar un ensamblado, donde el *Motor* y la *Carrocería* son sus componentes. Éste es un ejemplo de composición, ya que el *Motor* y la *Carrocería* son partes del *Automóvil*, y a diferencia de la agregación, no pueden ser compartidos entre varios *Automóviles* a la vez. Además, no tiene mucho sentido que el *Motor* y la *Carrocería* existan de manera independiente al *Automóvil*, por lo cual la composición refleja de manera importante el concepto de *propiedad*.

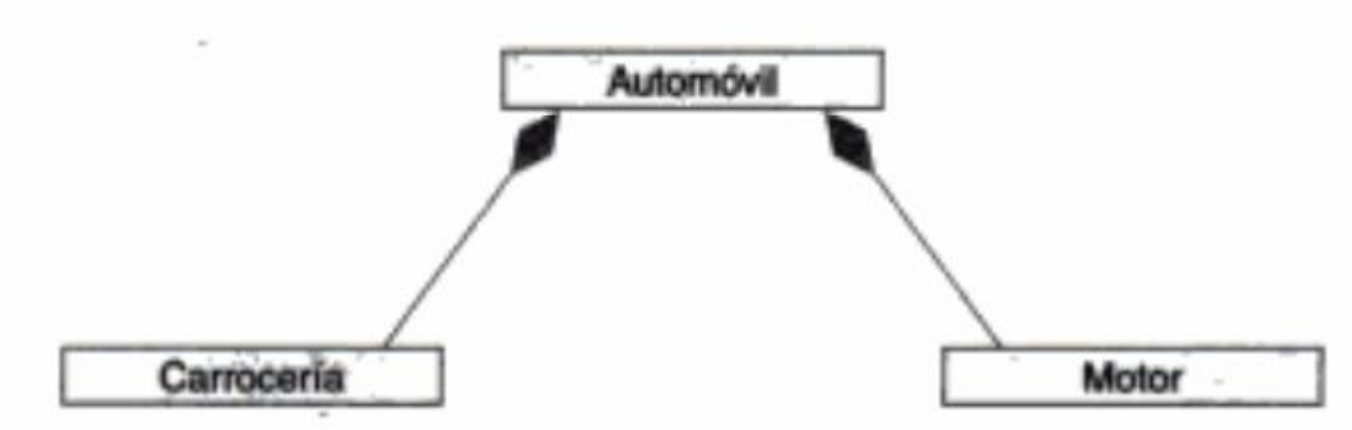
Ensamblados: agregación y composición

- Se considera un ensamblado y no una asociación regular:
  - Si se puede usar la frase "**parte-de**" o "**consiste-de**" o "**tiene**".
  - Si algunas operaciones en el todo pueden ***propagarse*** a sus partes.
  - Si algunos atributos en el todo se pueden **propagar** a sus partes.



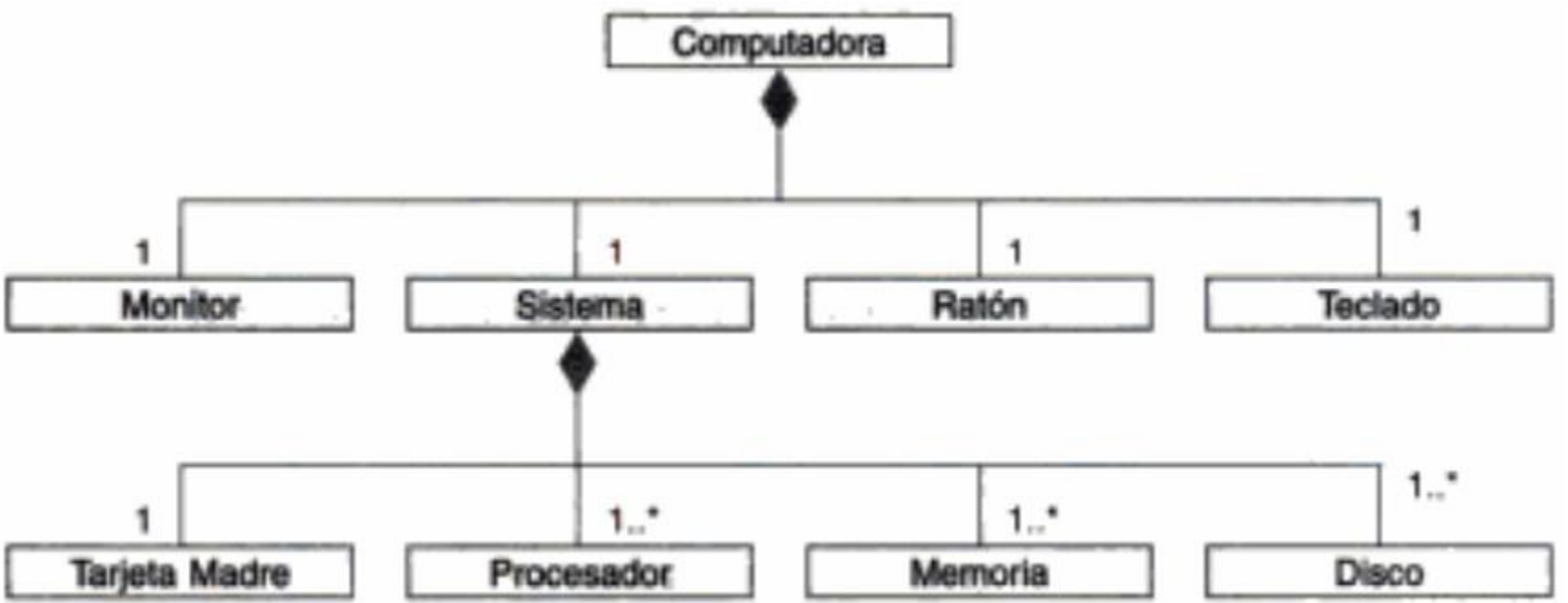
UNIVERSIDAD DE LIMA

Ensamblados: agregación y composición



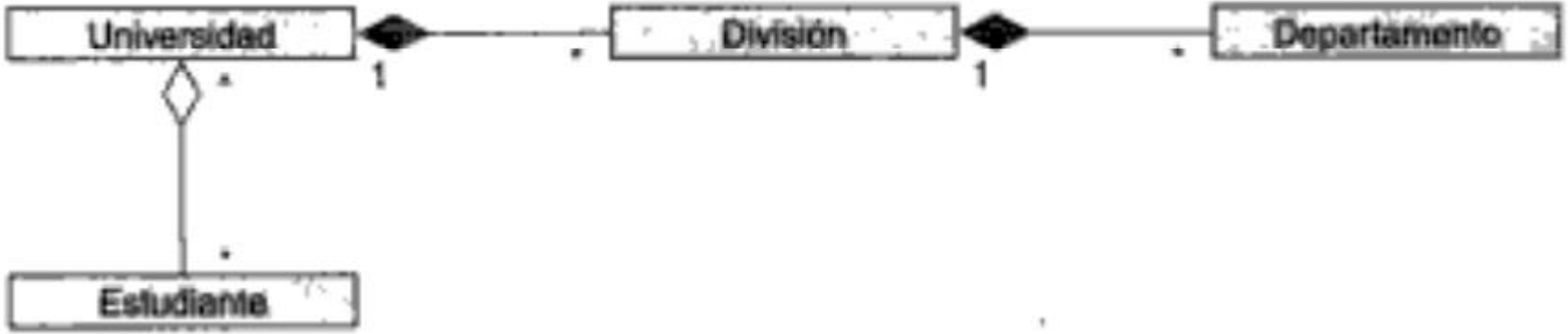
UNIVERSIDAD DE LIMA

Ensamblados: agregación y composición



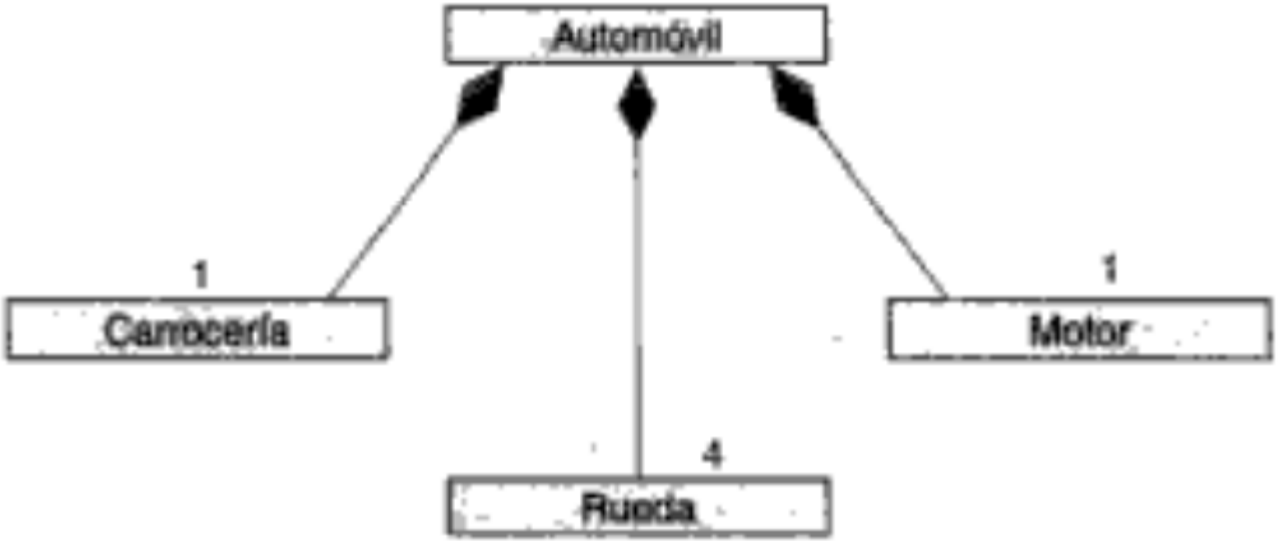
UNIVERSIDAD DE LIMA

Ensamblados: agregación y composición



Ensamblados: Tipos

- **Fijos**: Los que tienen una estructura fija donde el número de componentes está predefinido.



UNIVERSIDAD DE LIMA

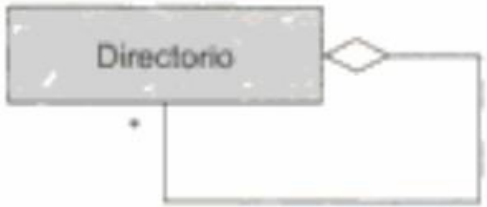
Ensamblados: Tipos

- **Variables**: Tienen un número finito de niveles, pero el número de componentes varía.

Ensamblados: Tipos

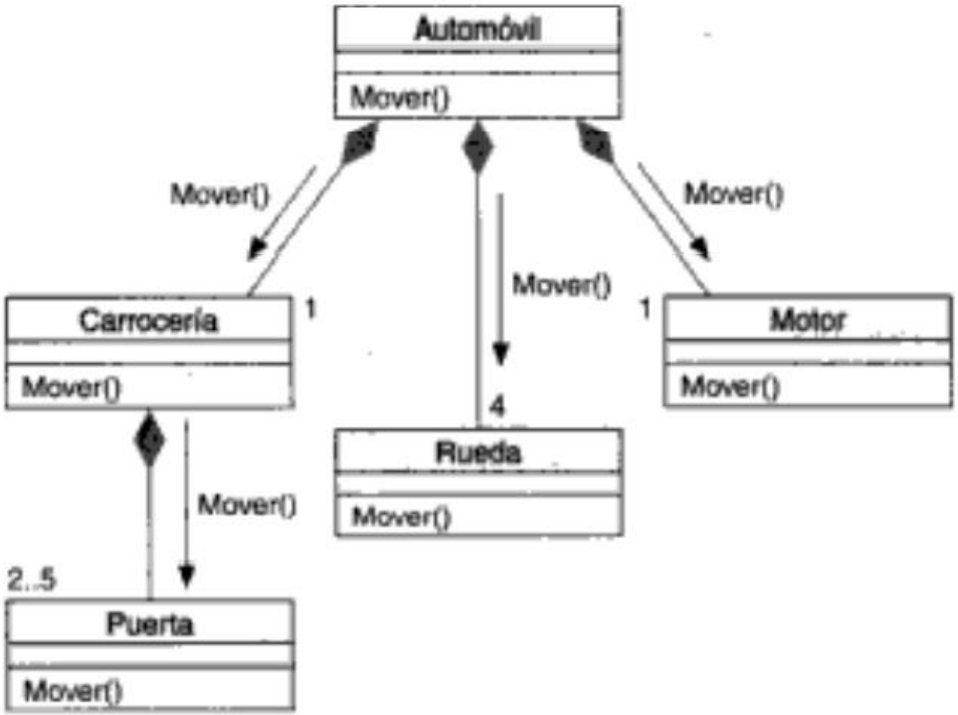
- **Recursivos**: Los ensamblados recursivos contienen de forma directa o indirecta una instancia del mismo tipo de agregado, donde el número de niveles de ensamblado es potencialmente ilimitado.

Ejemplo: Un *Directorio* en un sistema operativo está definido de forma recursiva, pudiendo contener otros directorios, que a su vez pueden incluir otros directorios, y así sucesivamente de forma indefinida, como se muestra en la figura 4.89.



Ensamblados: Propagación de operaciones

- Uno de los objetivos del ensamblado es que las operaciones aplicadas a éste puedan **propagarse** de forma automática a sus objetos componentes. La operación se propaga en una sola dirección, y puede ser aplicada a las partes del ensamblado de forma independiente.

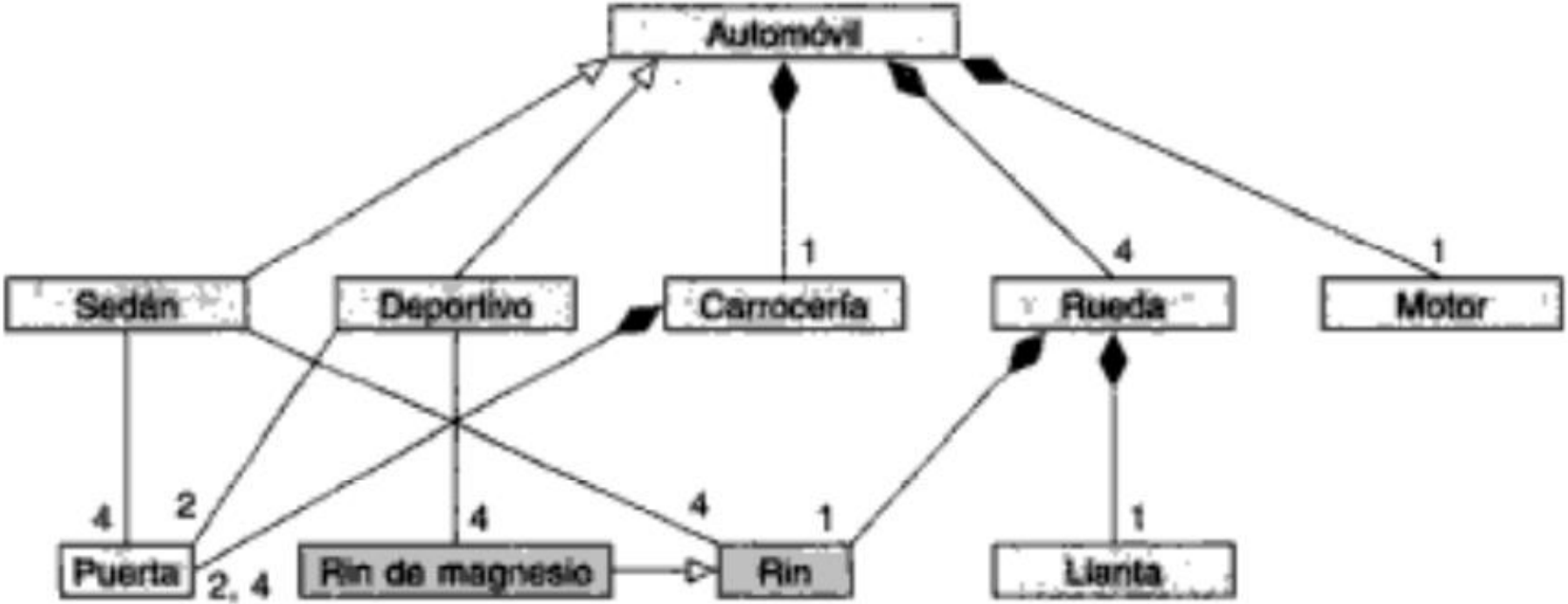


UNIVERSIDAD DE LIMA

Generalización y herencia

- La herencia es una relación "**es-una**" entre las clases más refinadas y generales.
- Las siguientes características se aplican a clases en una jerarquía de herencia:
  - Los valores de una instancia incluyen valores para cada atributo de cada clase ancestra.
  - Cualquier operación de cualquier clase ancestra se puede aplicar a una instancia.
  - Cada subclase no sólo hereda todas las características de sus ancestros, sino también añade sus propios atributos y operaciones.

Generalización y herencia



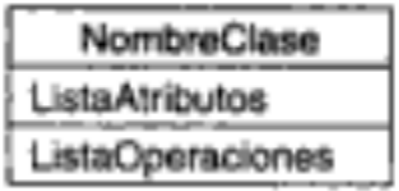
UNIVERSIDAD DE LIMA

UML y Java


```
class NombreClase {
}
```

UNIVERSIDAD DE LIMA

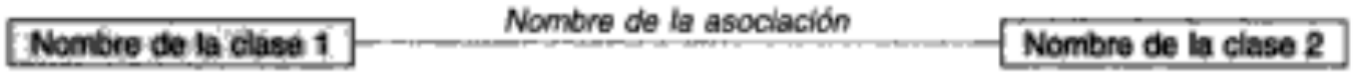
UML y Java




```
class NombreClase {
// atributos
    tipoAtributo1 nombreAtributo1;
    ...
    tipoAtributoi nombreAtributoi;
    ...
    tipoAtributoN nombreAtributoN;
// operaciones
    tipoRetorno1 nombreMétodo1 ( listaParámetrosMétodo1 )
    { cuerpoMétodo1 }
    ...
    tipoRetornoj nombreMétodoj ( listaParámetrosMétodoj )
    { cuerpoMétodoj }
    ...
    tipoRetornoM nombreMétodoM ( listaParámetrosMétodoM )
    { cuerpoMétodoM }
}
```

UNIVERSIDAD DE LIMA

UML y Java

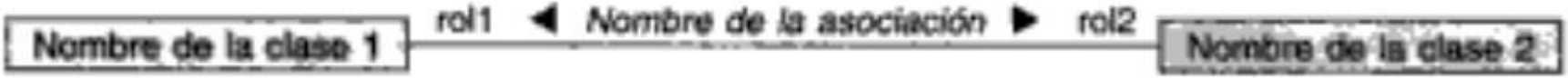



```
class Clase1 {
    Clase2 ref;
}
class Clase2 {
    Clase1 ref;
}
```

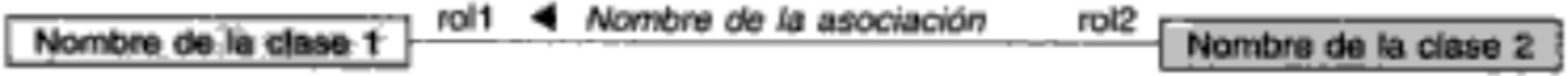

```
class Clase1 {
    Clase2 rol2;
}
class Clase2 {
    Clase1 rol1;
}
```

UNIVERSIDAD DE LIMA

UML y Java




```
class Clase1 {
    Clase2 rol2;
}
class Clase2 {
    Clase1 rol1;
}
```




```
class Clase1 {
}
class Clase2 {
    Clase1 rol1;
}
```

UNIVERSIDAD DE LIMA

UML y Java: Multiplicidad


```
class Clase1 {
    Clase2 rol2[];
}
class Clase2 {
    Clase1 rol1;
}
```

UNIVERSIDAD DE LIMA

UML y Java: Multiplicidad


```
class Clase1 {
    Clase2 rol2[];
}
class Clase2 {
    Clase1 rol1[];
}
```

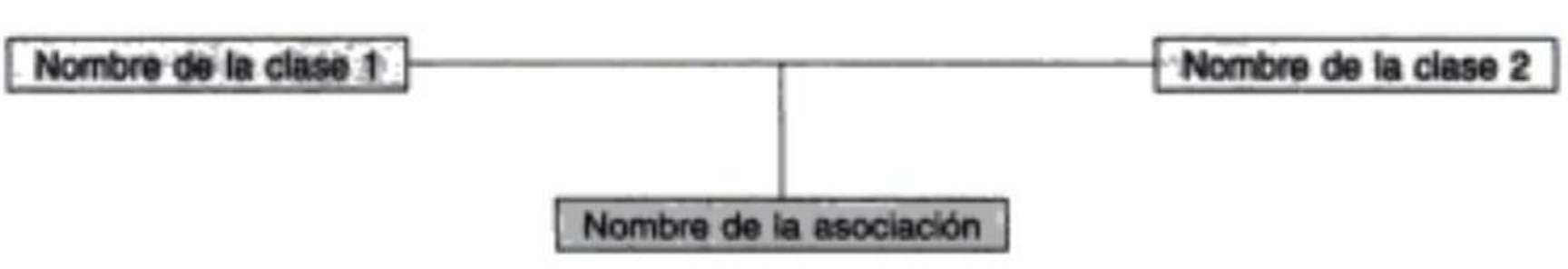
UNIVERSIDAD DE LIMA

UML y Java: Asociación reflexiva


```
class Clase {
    Clase rol1;
    Clase rol2[];
}
```

UNIVERSIDAD DE LIMA

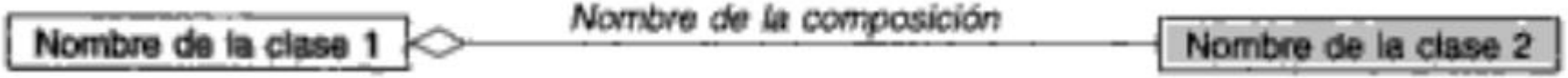
UML y Java: Asociación con clase




```
class Clase1 {
    Asociacion ref;
}
class Clase2 {
    Asociacion ref;
}
class Asociacion {
    Clase1 ref1[];
    Clase2 ref2[];
}
```

UNIVERSIDAD DE LIMA

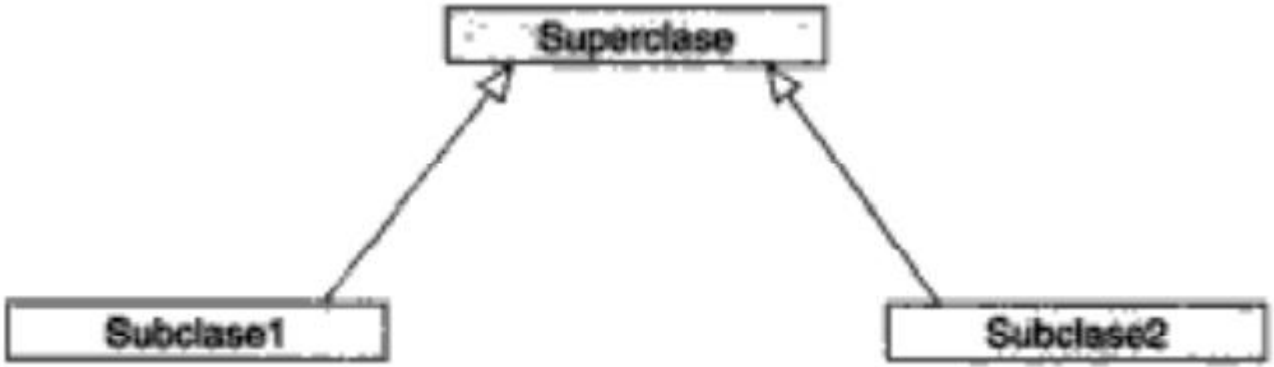
UML y Java: Agregación y composición




```
class Clase1 {
    Clase2 ref;
}
class Clase2 {
    Clase1 ref;
}
```

UNIVERSIDAD DE LIMA

UML y Java: Generalización y Herencia




```
class Superclase {
}
class Subclase1 extends Superclase {
}
class Subclase2 extends Superclase {
}
```

UNIVERSIDAD DE LIMA

UML y Java: Sobreescritura y polimorfismo

UML y Java: Clases Abstractas

- Las clases abstractas son un aspecto básico de la generalización, dado que definen clases que requieren subclases para poder utilizarse.

```
public abstract class FormaGrafica {
        ...
        public void desplegar(int x, int y) {
        }
        ...
}
```

```
public abstract class FormaGrafica {
        ...
        public abstract void desplegar(int x, int y);
        ...
}
```

Métodos Abstractos

UNIVERSIDAD DE LIMA

UML y Java: Interfases

- Las interfases son similares a clases abstractas, excepto que se utiliza la palabra *interface* en lugar de *abstract* y *class*.

```
public interface NombreInterface {
    listaMetodos
}
```

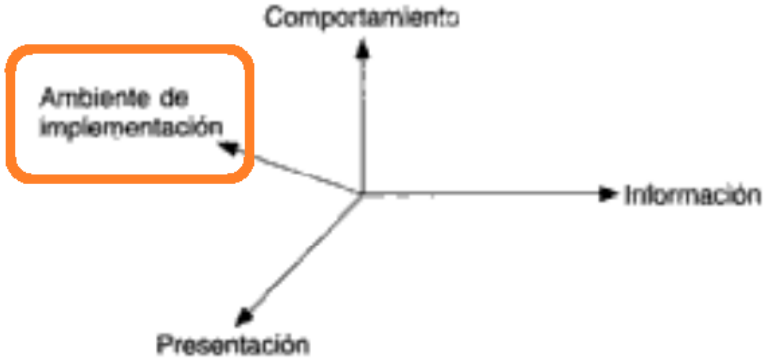
UNIVERSIDAD DE LIMA | 60 años

Modelo de diseño

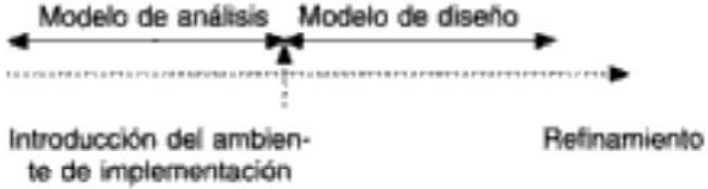
El modelo de diseño es un refinamiento y formalización adicional del modelo de análisis. El resultado del modelo de diseño son especificaciones muy detalladas de todos los objetos, incluyendo sus operaciones y atributos.

Formalización del Análisis

- El modelo de análisis no es lo suficiente formal para alcanzar el código fuente.
- El análisis se considera un mundo ideal para el sistema, en la realidad se debe adaptar el sistema al **ambiente de implementación**
- El modelo de diseño incluye una nueva dimensión adicional que refina el modelo de análisis hasta que el código fuente sea fácil de escribir



El diseño añade el ambiente de implementación como un nuevo eje de desarrollo.



El modelo de diseño es una continuación del modelo de análisis.

Nivel de detalle del modelo de diseño

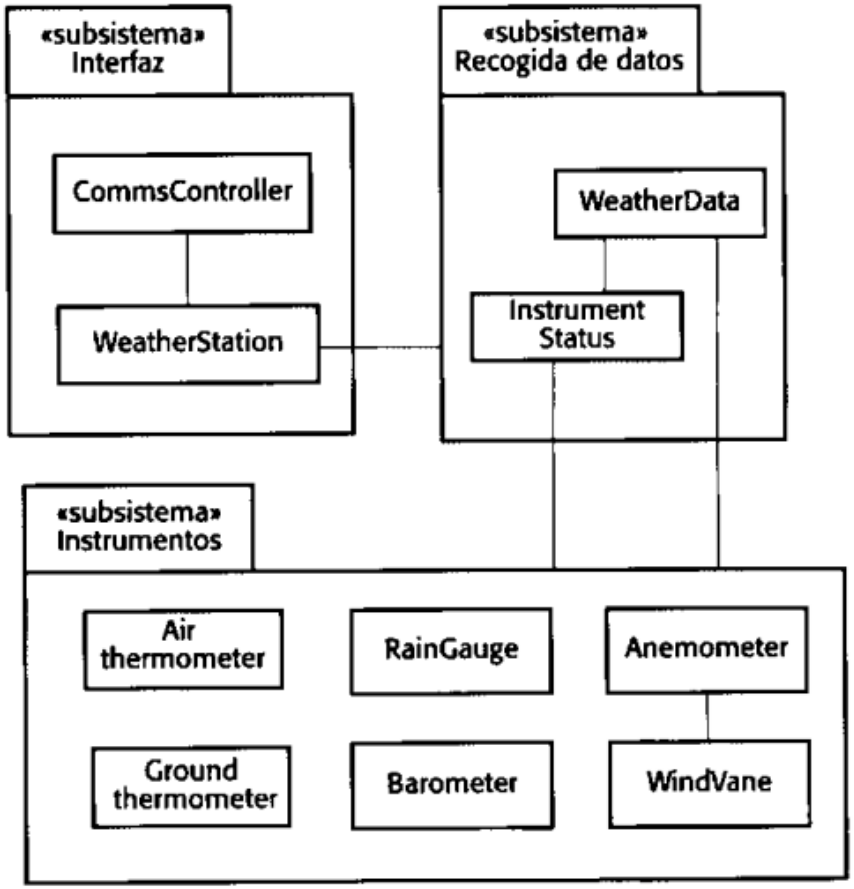
- El proceso de diseño es decidir qué modelos de diseño son necesarios y el nivel de detalle de los mismos.
- Existen dos tipos de modelos de diseño OO:
  - Modelos estáticos: Estructura estática del sistema en términos de clases y sus relaciones (Relaciones importantes: Generalización, utiliza/utilizado-por y composición).
  - Modelos dinámicos: Estructura dinámica del sistema. Muestra las **interacciones entre los objetos y no entre clases**. Se documentan las interacciones de **secuencia** de los servicios solicitados por los objetos y la forma en que el **estado** del sistema se relaciona con estas interacciones de objetos

Diagramas UML

- **Diagramas UML**: Sirve para representar las clases con diagramas de clases y tiene muchos otros diagramas.

Modelos de diseño en UML

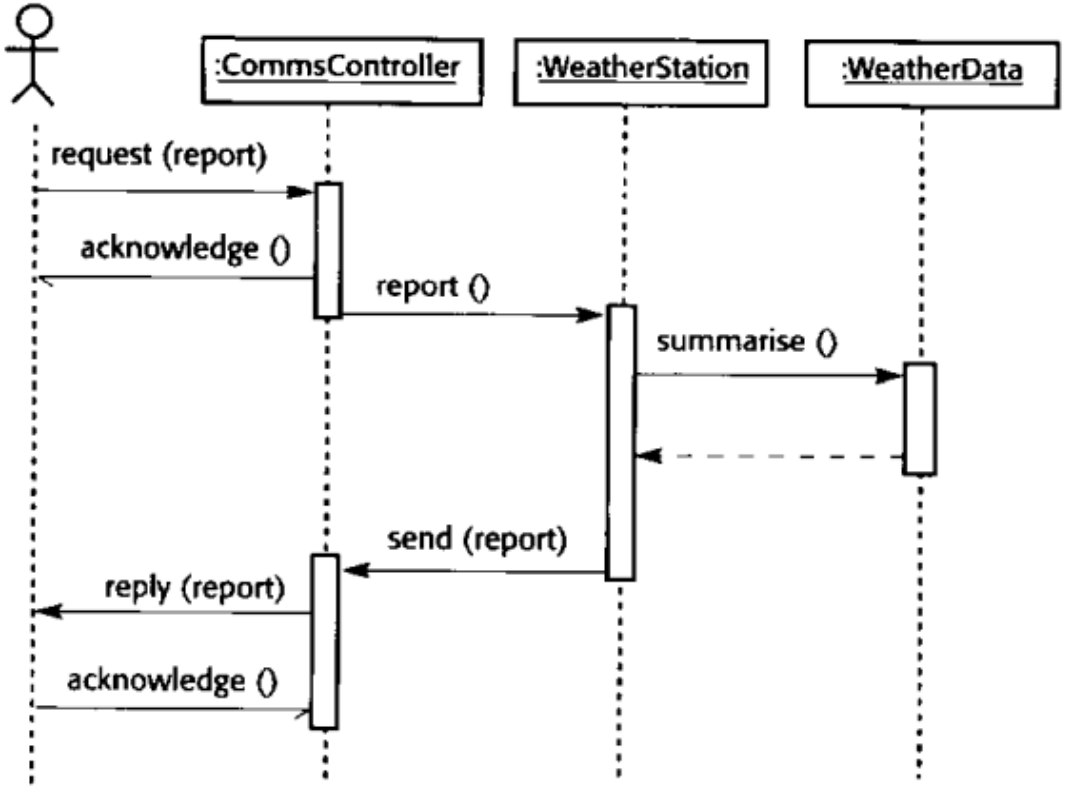
- **Modelos de subsistemas**: Utiliza diagrama de clases y paquetes (*Estáticos*)
- **Modelos de secuencia**: Muestra la secuencia de interacciones de objetos. Utiliza diagramas de secuencia o colaboración. (*Dinámicos*)
- Modelos de máquinas de estado: muestra cómo los objetos individuales cambian su estado en respuesta a los eventos. Utiliza diagrama de estado (*Dinámicos*)



Modelos de diseño en UML: Diagrama de interacción

- Representan la interacción entre objetos.
  - Los **tipos** de diagramas de interacción son:
    - Diagramas de secuencia
    - Diagramas de comunicación (colaboración).
- Muestra la interacción con respecto al **tiempo**.
  - El diagrama muestra:
    - Los objetos que participan en la interacción
    - La secuencia de mensajes que intercambian

Ejemplo de un diagrama de secuencia



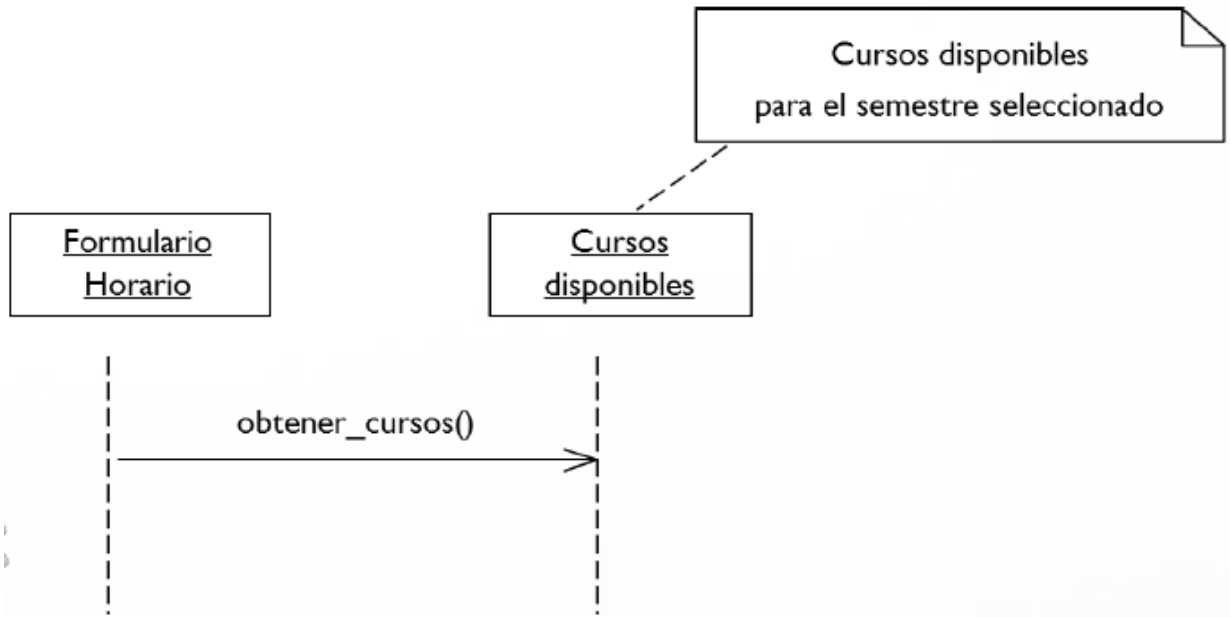
UNIVERSIDAD DE LIMA

Colocando el Nombre de los Objetos

- Los **objetos** son dibujados como rectángulos
  - Las **"líneas de vida"** son representadas por líneas verticales punteadas:

Notas

- Las **notas** pueden ser añadidas para agregar información al diagrama:



Modelos de diseño en UML: Diagrama de estados

Conceptos:

- Diagrama que describe los diferentes estados de cada objeto. Modela la naturaleza dinámica del sistema. Por lo general, no se requiere elaborar diagramas de estado para todos los objetos.
- **Estado**: Condición de un objeto en un momento determinado.
- **Evento**: Acontecimiento sobre un objeto que puede generar un cambio en su estado.

Modelos de diseño en UML: Diagrama de estados

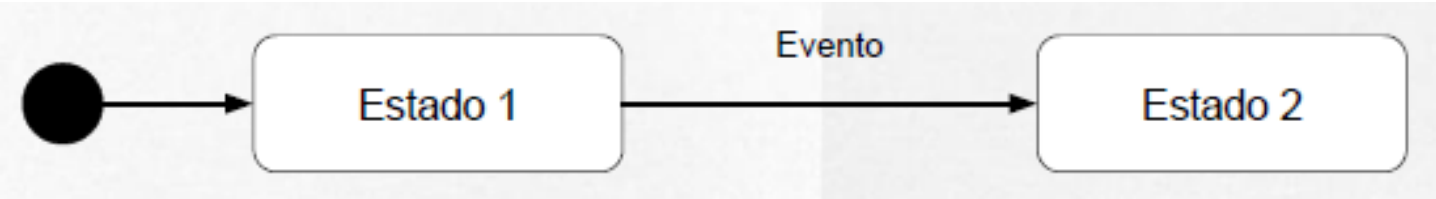
Tipos de Eventos:

- **Evento Interno:** Factor interno del sistema. Evento iniciado a través de una operación invocado por otro objeto.
- **Evento externo:** Llamado evento del sistema. Iniciado por un factor externo al sistema (Ejemplo: un actor).
- **Evento temporal**: Puede ser:
  - Evento iniciado al alcanzar una fecha u hora específica
  - Evento iniciado luego de que ha transcurrido un tiempo establecido

Modelos de diseño en UML: Diagrama de estados

Notación de transición:

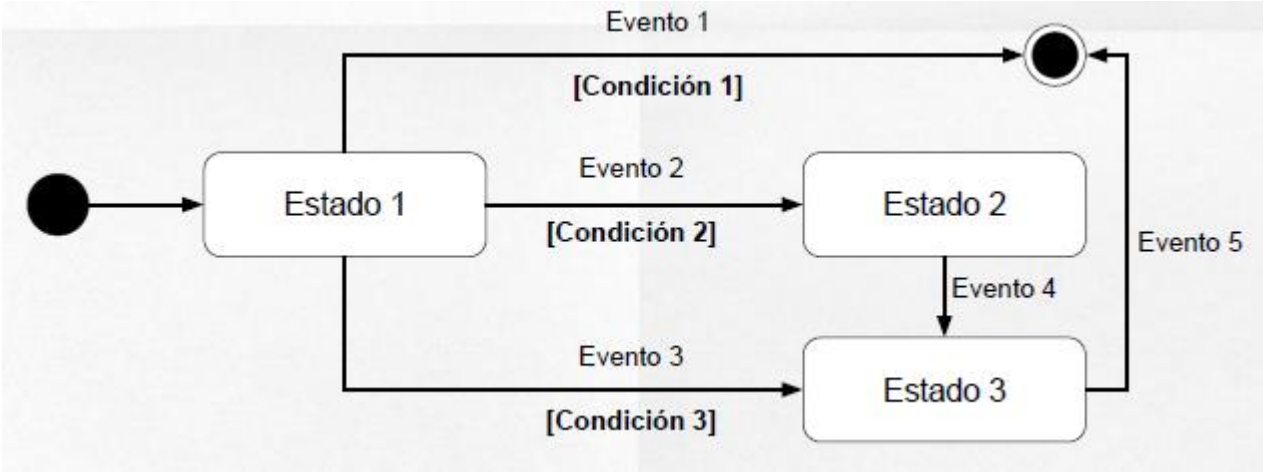
- Expresa relación entre estados
- Se produce al ocurrir un evento que promueve el cambio del estado de un objeto.



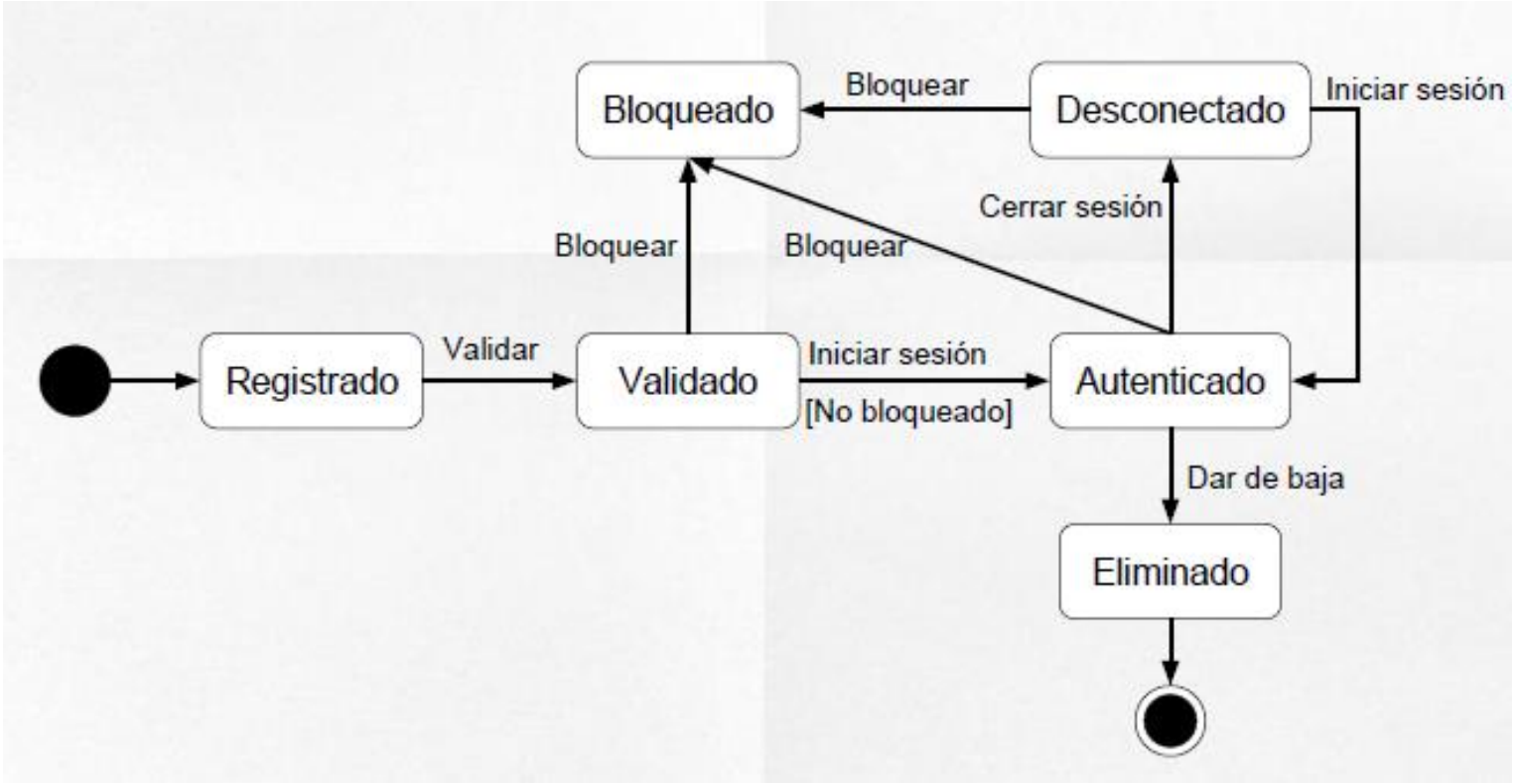
Modelos de diseño en UML: Diagrama de estados

Notación: Condición de guarda

- Condición que debe de cumplirse para que un evento pueda promover un cambio de estado.



Ejemplo: Objeto Usuario



Modelos de diseño en UML: Diagrama de estados

Pasos para elaborar un Diagrama de Eventos:

1. Identificar los principales objetos a analizar.
2. Identificar los posibles estados por cada objeto.
3. Identificar los eventos asociados a las transiciones de cada estado.

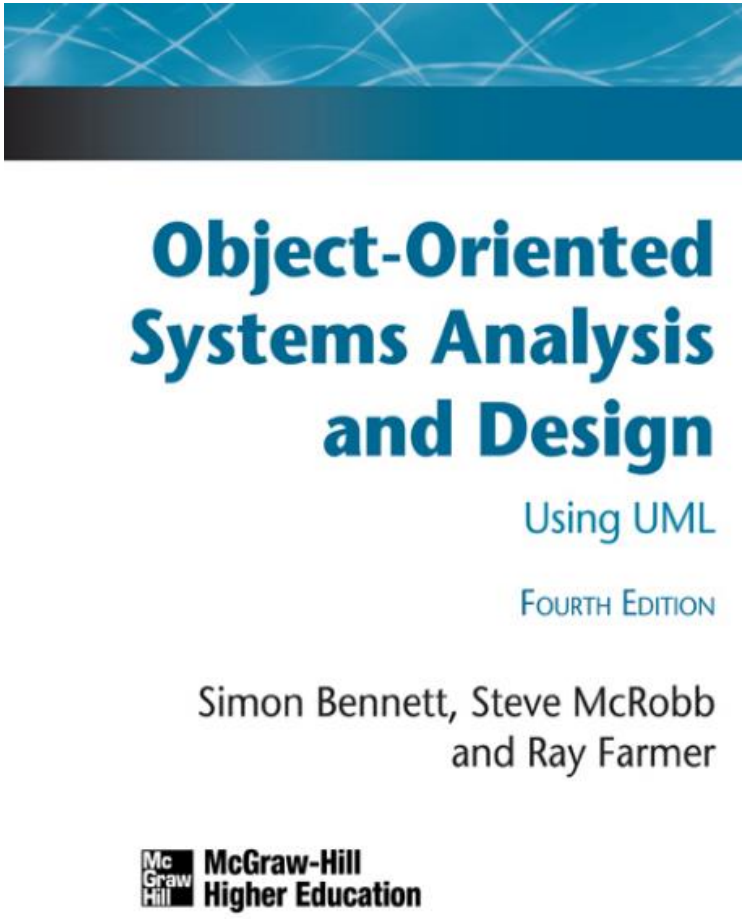
UNIVERSIDAD DE LIMA | 60 años

Diseño detallado

Se agrega detalle al modelo de clases. Se puede modificar las clases y relaciones. Agrega nuevas clases relacionadas a la interacción humano-Computador y gestión de datos.

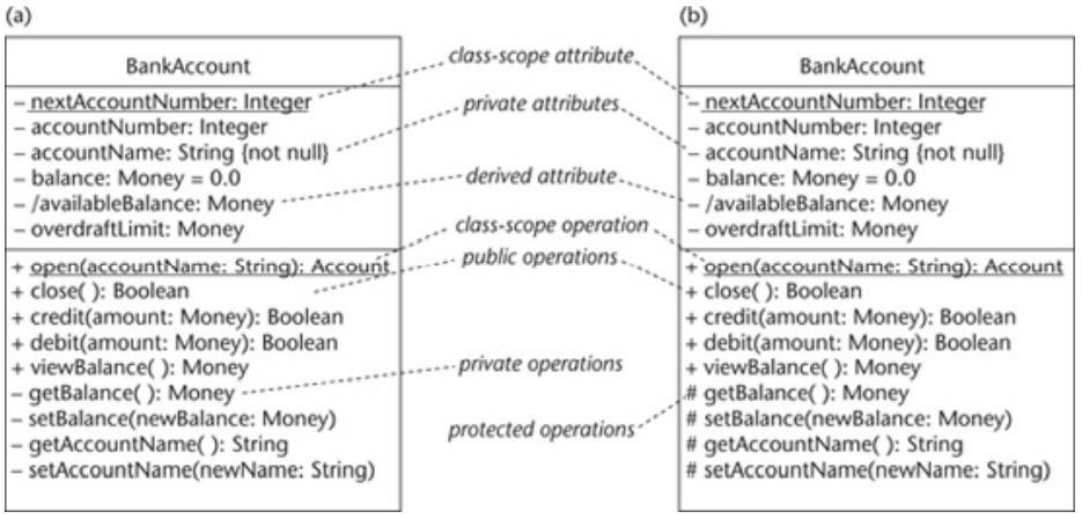
Diseño detallado

- Especificación de atributos y operaciones
- Agrupar atributos y operaciones en clases
- Diseño de Asociaciones
- Restricciones de integridad
- Diseño de algoritmos de operación



Especificación de atributos y operaciones

- Decidir el tipo de datos de cada atributo;
- Decidir cómo manejar los atributos derivados;
- Agregar operaciones primarias;
- Definir la firma de operaciones incluyendo los tipos de parámetros;
- Decidir la visibilidad de atributos y operaciones.

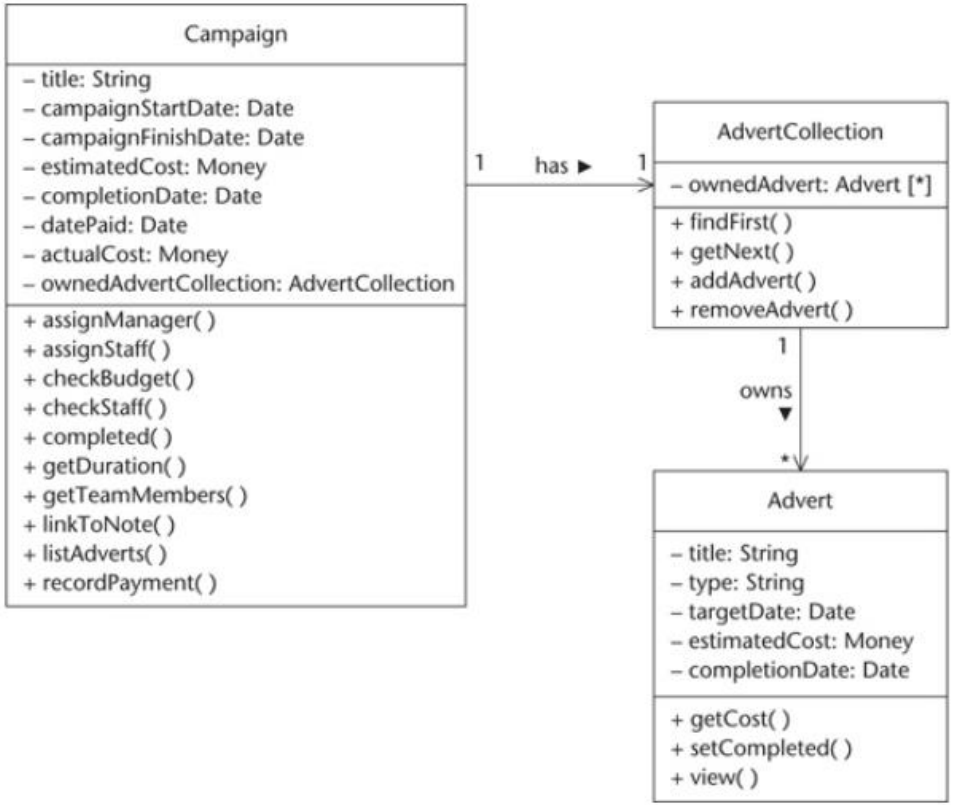


Agrupar atributos y operaciones en clases

- Verificar que las **responsabilidades** se hayan asignado a la clase correcta;
- Definir o usar interfaces para agrupar comportamientos estándar bien definidos;
- Aplicar los conceptos de acoplamiento y cohesión **(Ver siguiente semana)**;
- Aplicar el principio de sustitución de Liskov. **(Ver siguiente semana)**

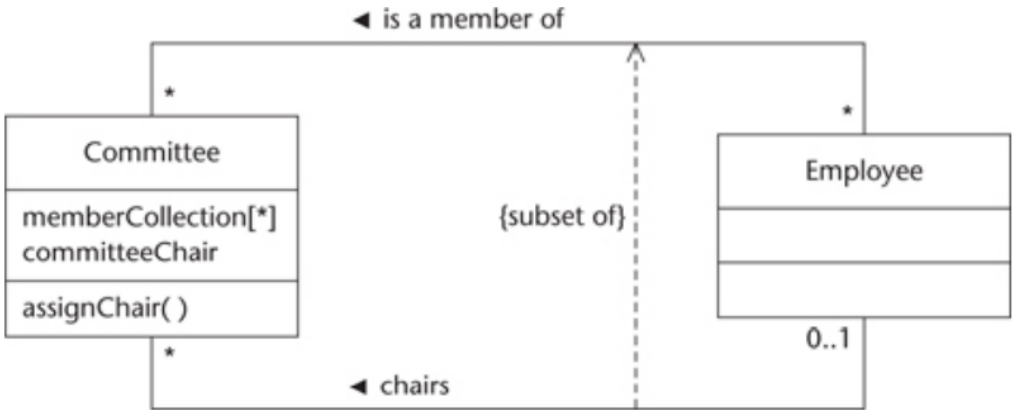
Diseño de Asociaciones

- Asociaciones uno a uno
- Asociaciones de uno a muchos
- Asociaciones de muchos a muchos

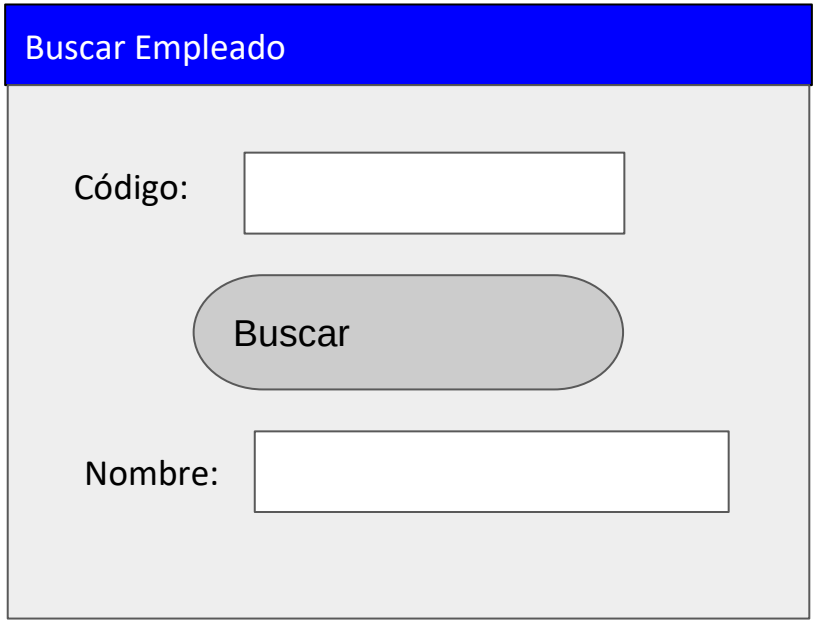


Integrity Constraints

- ***Integridad referencial:*** garantiza que un identificador de objeto en un objeto se refiera realmente a un objeto que existe;
- ***Restricciones de dependencia***: Aseguran que las dependencias de atributos, donde un atributo puede calcularse a partir de otros atributos, se mantengan consistentemente;
- ***Integridad del dominio:*** Garantiza que los atributos solo tengan valores permitidos.

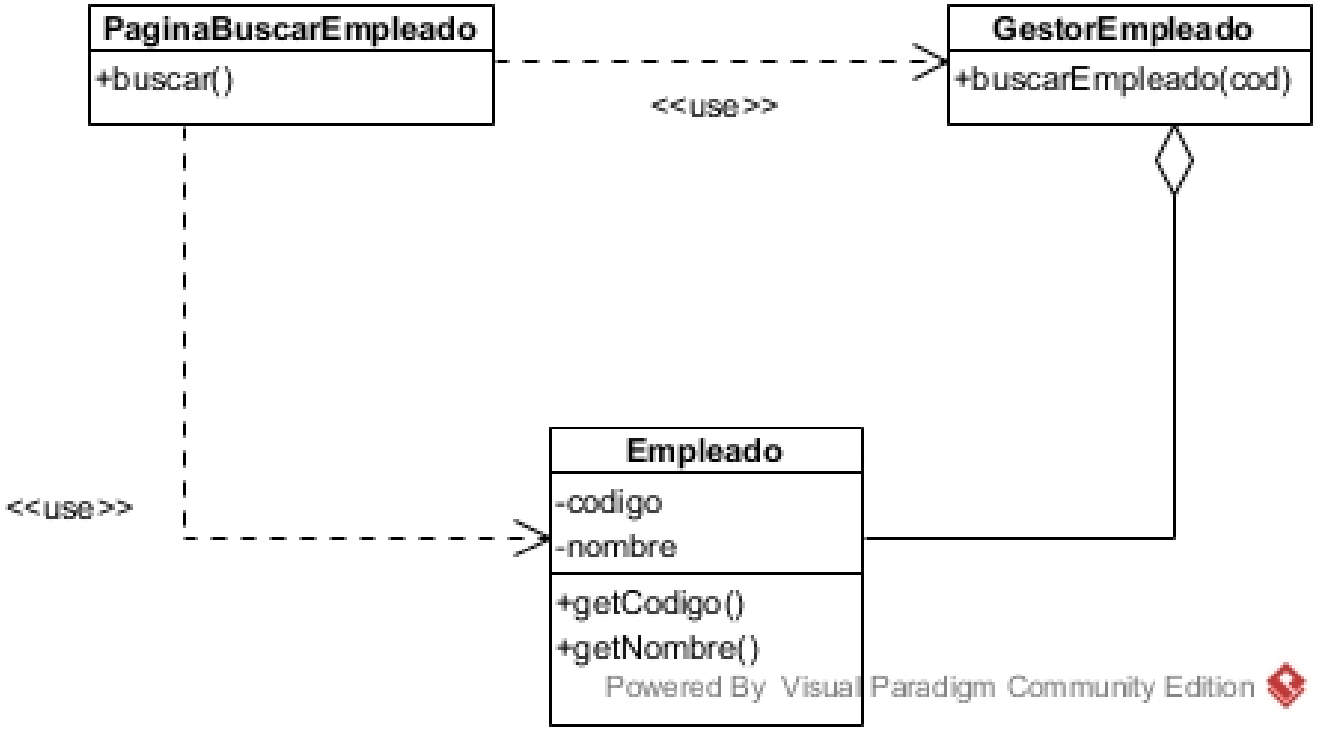


Ejemplo de Diseño Detallado - Diagrama de Secuencia

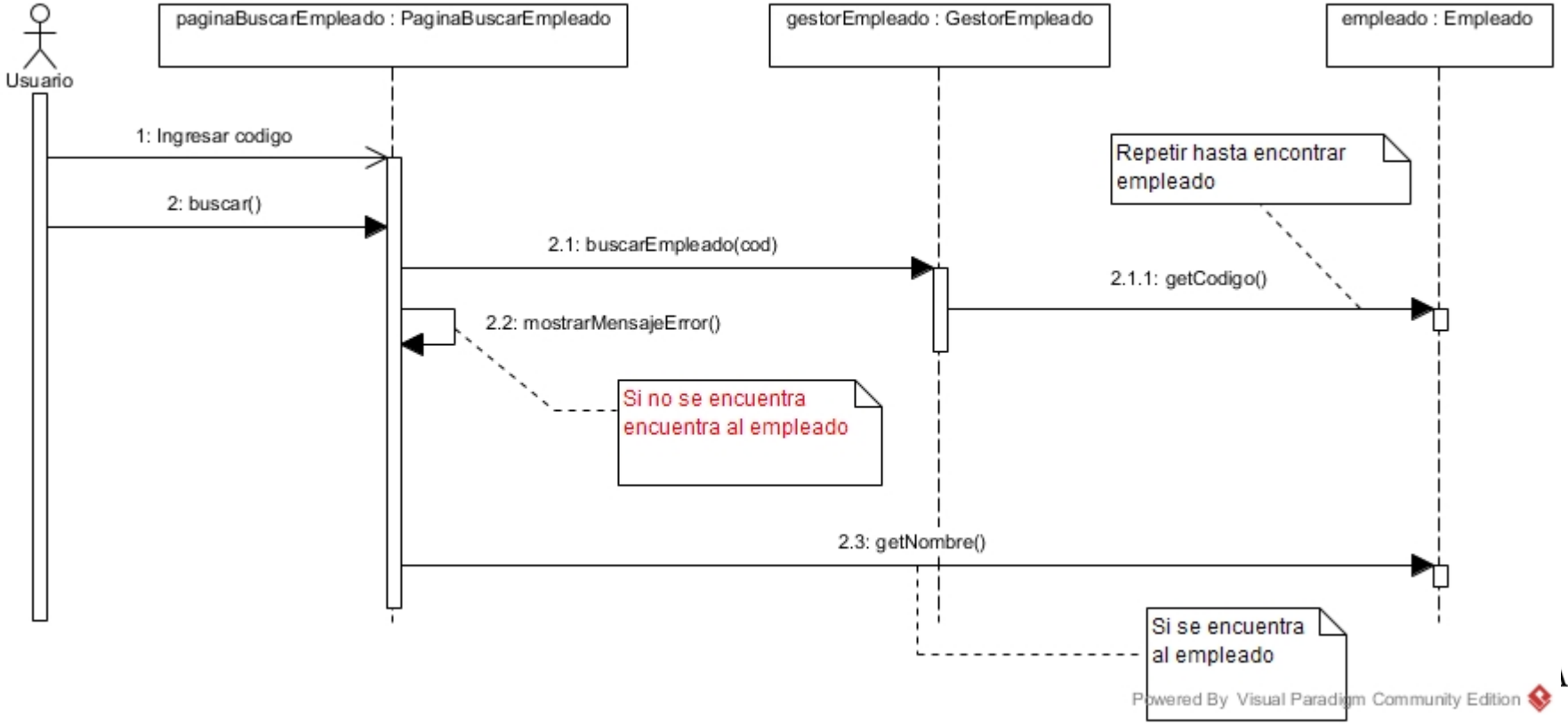


UNIVERSIDAD DE LIMA

Ejemplo de Diseño Detallado - Diagrama de Secuencia



Ejemplo de Diseño Detallado - Diagrama de Secuencia



Codificar el diseño

UNIVERSIDAD DE LIMA | 60 años

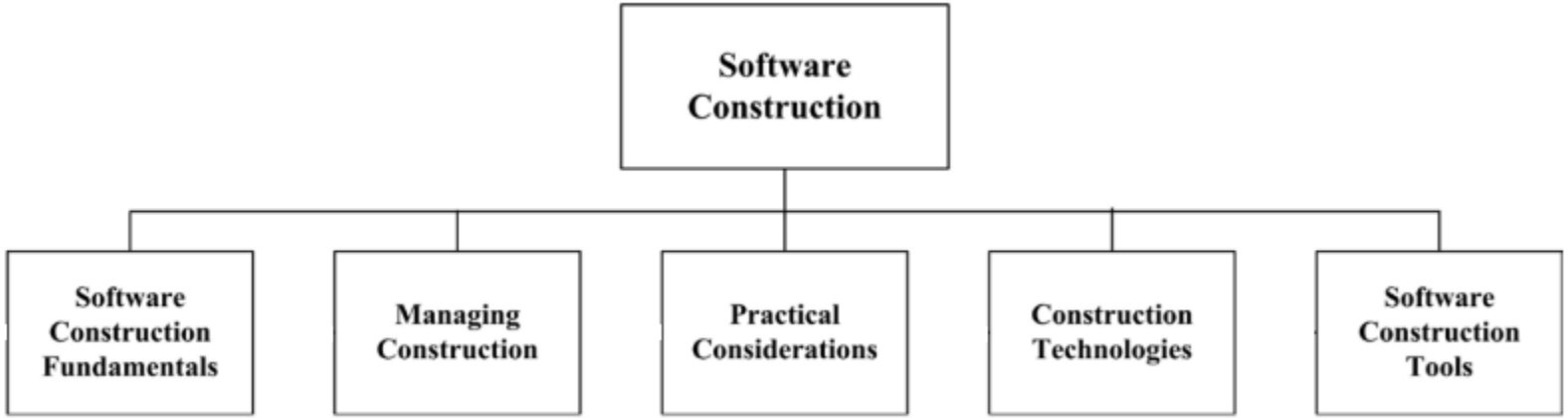
Aspectos de implementación del software

La implementación de software se refiere a la creación detallada de software funcional combinando actividades de codificación, verificación, pruebas unitarias, pruebas de integración y depuración.

Construcción del software

- Está más relacionado a las actividades de **diseño y pruebas del software**
- Produce la mayor cantidad de elementos de configuración que deben administrarse en un proyecto de software (archivos fuente, documentación, casos de prueba, etc.) -> **Gestión de configuración**
- Requiere conocimiento de algoritmos y de prácticas de codificación -> **Fundamentos de computación**.

Temas para la Construcción del Software según SWEBOK 3.0



UNIVERSIDAD DE LIMA | 60 años

Medidas de calidad del diseño OO

Las medidas se pueden utilizar para evaluar o estimar cuantitativamente varios aspectos de un diseño de software; por ejemplo, tamaño, estructura o calidad.

Medidas

La mayoría de las medidas que se han propuesto dependen del enfoque utilizado para producir el diseño. Estas medidas se clasifican en dos grandes categorías:

- **Medidas de diseño basadas en funciones (estructuradas)**: medidas obtenidas mediante el análisis de la descomposición funcional; generalmente representado usando un gráfico de estructura (a veces llamado diagrama jerárquico) en el que se pueden calcular varias medidas.
- **Medidas de diseño orientadas a objetos:** la estructura de diseño normalmente se representa como un diagrama de clases, en el que se pueden calcular varias medidas. También se pueden calcular medidas sobre las propiedades del contenido interno de cada clase.

Referencias Bibliográficas

1. Bennett, S., McRobb, S., & Farmer, R. (2010). Object-oriented systems analysis and design using UML. McGraw-Hill.
2. Weitzenfeld, A. (2004). Ingeniería de software orientada a objetos con UML, Java e Internet. Editorial Thomson.