

# Sesión 6

## **Principio de los diagramas de UML**

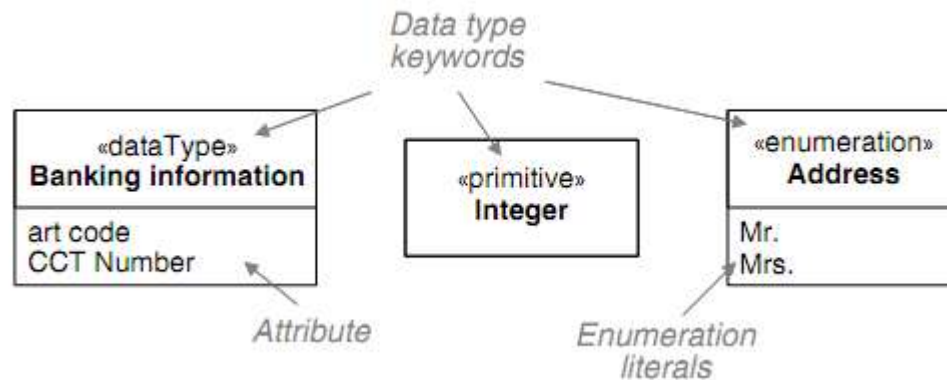
Unidad 2

**Introducción al Unified Model  
Language (UML)**

Mg. Gustavo G. Delgado Ugarte

# Tipos de datos en UML

- UML distingue los siguientes tipos de datos:
  - Tipos de datos simples (DataType)
  - Tipos de datos primitivos (PrimitiveType)
  - Tipos Enumeración



# Tipos de datos en UML

- Tipos de datos simples (DataType)
  - Tipo con valores que no tienen una identidad
    - Ej. dinero, información bancaria
    - Dos instancias de un tipo de datos con los mismos valores de los atributos son indistinguibles

# Tipos de datos en UML

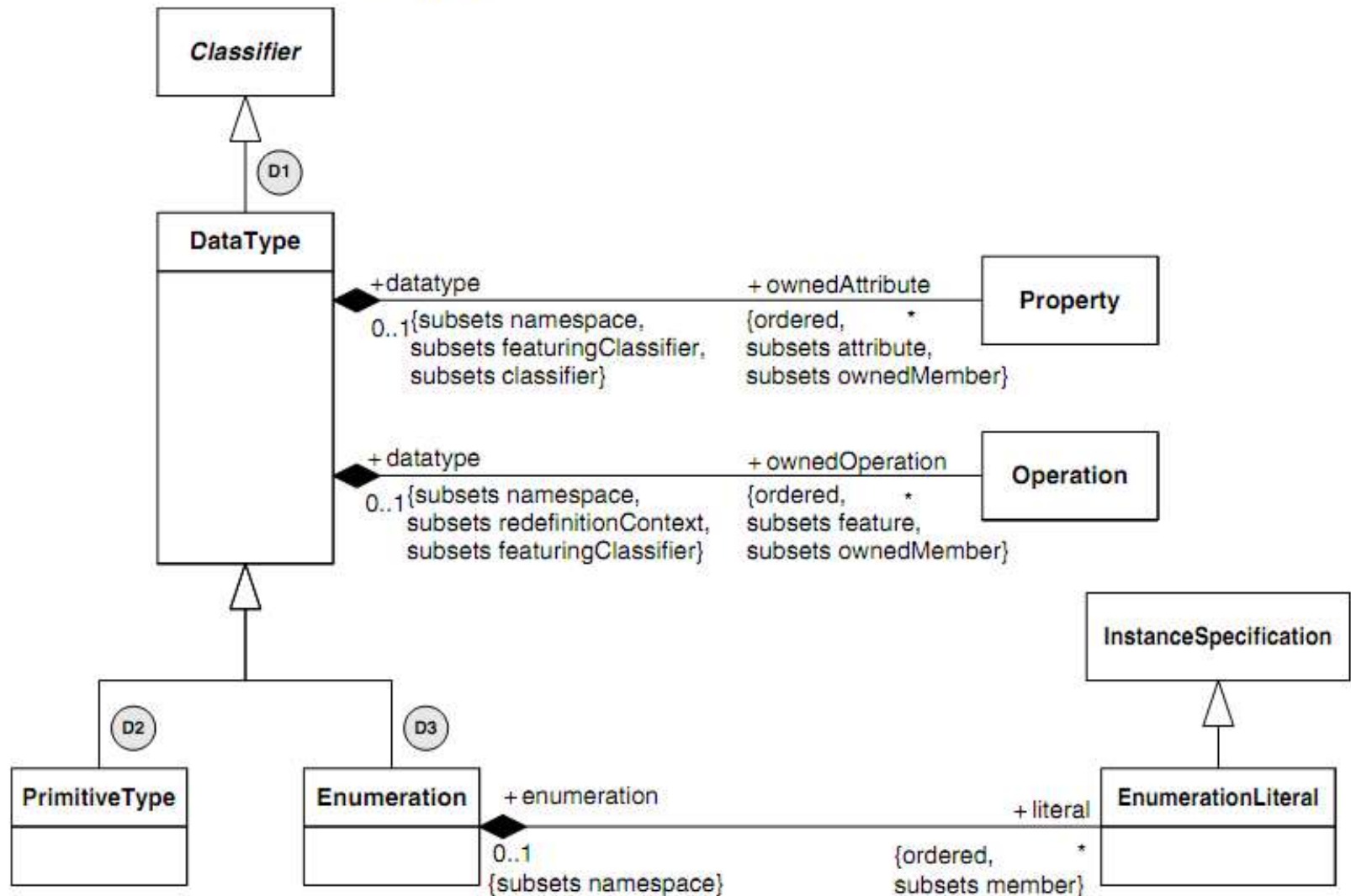
- Tipos de datos primitivos (PrimitiveType)
  - Tipo de datos simple sin estructuras
  - UML define los siguientes tipos de datos primitivos:
    - Entero: (infinito)
      - Conjunto de enteros: (... , -2, -1, 0, 1, 2, ...)
    - Boolean: verdadero, falso
    - Naturales Ilimitados (Infinito)
      - Conjunto de números naturales (0, 1, 2,...)
      - El símbolo de infinito es el asterisco (\*)
      - Este tipo de datos se utiliza, por ejemplo, en el metamodelo para definir multiplicidades

# Tipos de datos en UML

- Tipos Enumeración
  - Tipos de datos simples con valores que se originan de un conjunto limitado de literales enumerados
  - Ejemplo: Estado civil

# Tipos de datos en UML

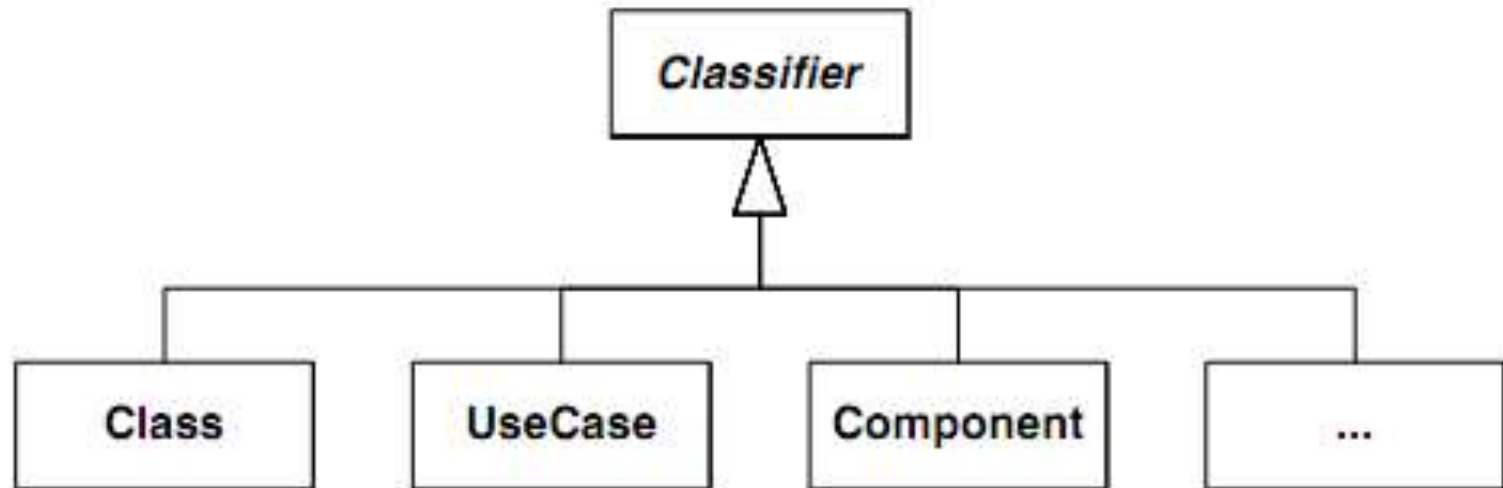
Metamodel



# Clasificadores

- Aunque el metamodelo de la clase Clasificador es abstracto, lo que significa no se ve como un elemento de modelado, una notación se define en la especificación
- Esta notación es heredada por las subclases y utilizados por ellas
  - Algunas de la notaciones eventualmente se sobrescribirán

# Clasificadores



---

**FIGURE 2.28** A classifier and a set of concrete subclasses (simplified).



# Clasificadores

- Es una clase base abstracta que clasifica las instancias en lo que respecta a sus características
- Es un espacio de nombres (namespace) y un tipo, y puede ser generalizada
- Clasificadores concretos en el metamodelo UML incluyen, por ejemplo, clase, componente, y caso de uso
- Un clasificador asocia un conjunto de características
  - Características concretas son operaciones y atributos

# Clasificadores

## Metamodel

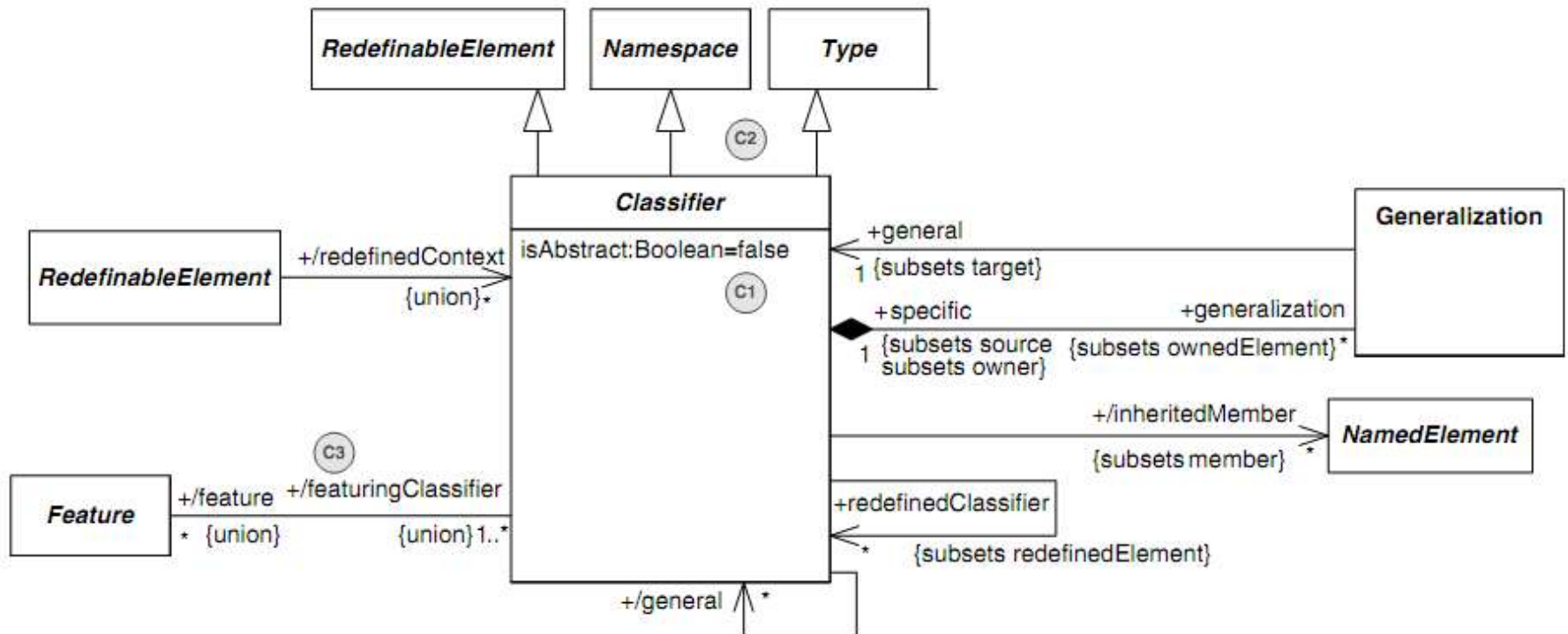
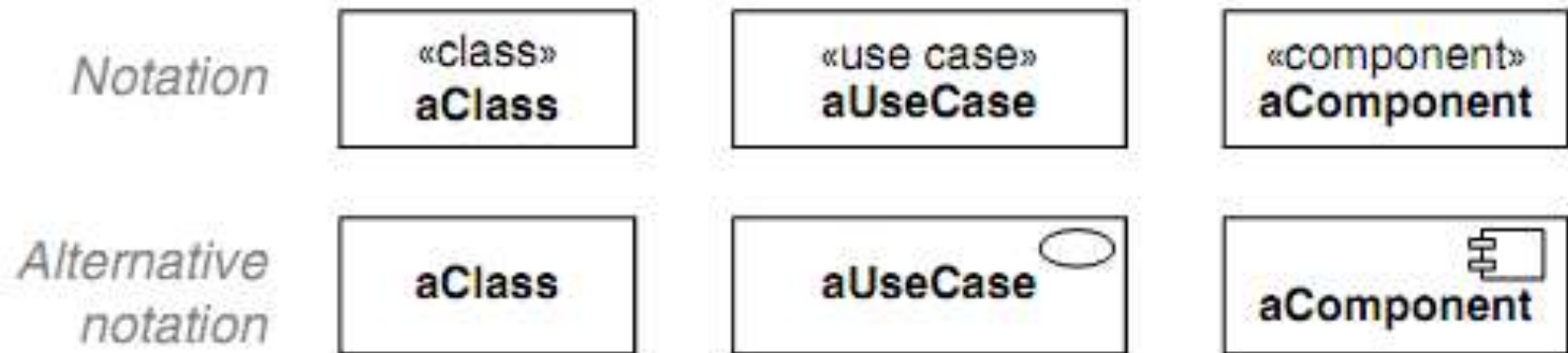


FIGURE 2.32 The classifier metamodel.

# Clasificadores





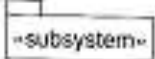

- La notación estándar para el clasificador es un rectángulo que contiene el nombre con el nombre de la subclase entre guillemets (<<, >>) por encima de ella
- Cada subclase introduce variantes de notación
  - El nombre de la clase concreto del metamodelo («clase») suele omitirse
  - Un caso de uso es generalmente representado en la notación por una elipse

# Clasificadores



**FIGURE 2.29** Examples of the notation of Classifier subclasses.

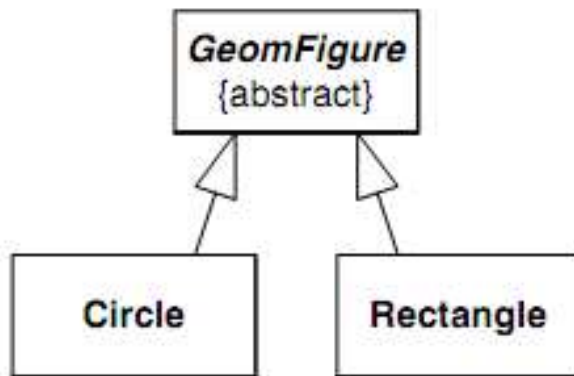
**Tabla 4.1** Tipos de Clasificadores

<i>Clasificador</i>	<i>Función</i>	<i>Notación</i>
actor	Un usuario externo al sistema	
clase	Un concepto del sistema modelado	<b>Nombre</b>
clase en un estado	Una clase restringida a estar en el estado dado	<b>Nombre[S]</b>
rol	Clasificador restringido a un uso particular en una colaboración	<b>rol:Nombre</b>
componente	Una pieza física de un sistema	
tipo de dato	Un descriptor de un conjunto de valores primitivos que carecen de identidad	<b>Nombre</b>
interfaz	Un conjunto de operaciones con nombre que caracteriza un comportamiento	 <b>lnombre</b>
nodo	Un recurso computacional	
señal	Una comunicación asíncrona entre objetos	<b>~signal~</b>
subsistema	Un paquete que es tratado como una unidad con una especificación, implementación, e identidad	 <b>~subsystem~</b>
caso de uso	Una especificación del comportamiento de una identidad y su interacción con los agentes externos	

# Clasificadores

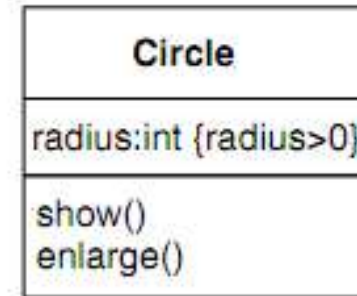
- Un clasificador es abstracto si su descripción es incompleta
  - Esto significa que no se pueden crear instancias
- La Abstracción es una característica de los clasificadores
  - Los nombres de los clasificadores abstractos se escriben en cursiva.
  - Si lo desea, puede agregar la cadena de propiedad {abstracta}.
  - Además de la zona de notación para el nombre (compartimentos en UML), puede haber otros, por ejemplo, para los atributos y las operaciones

# Clasificadores



---

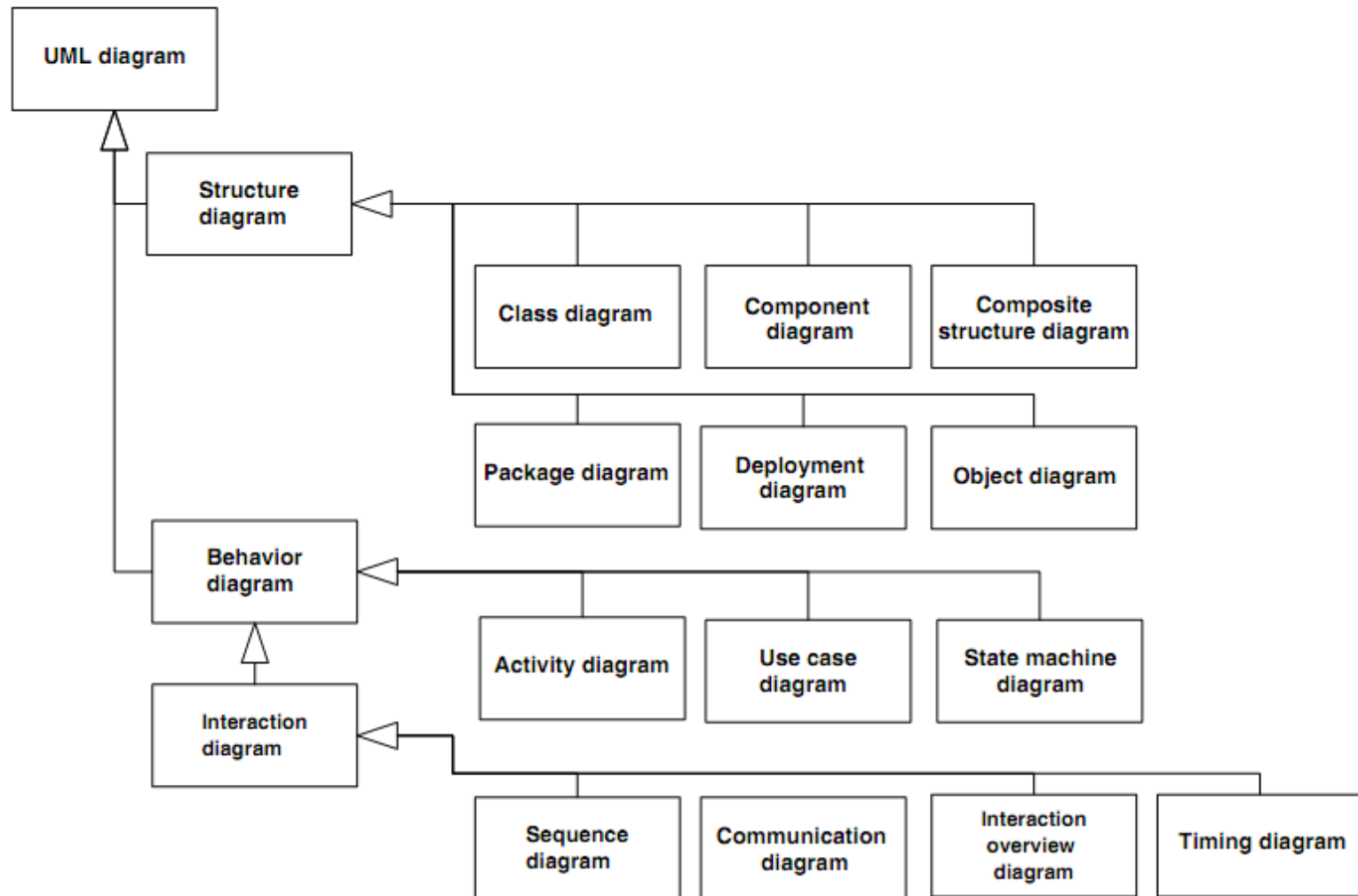
**FIGURE 2.30** An abstract class called GeomFigure.



---

**FIGURE 2.31** A class with compartments for name, attributes, and operations.

# Introducción a los diagramas en UML



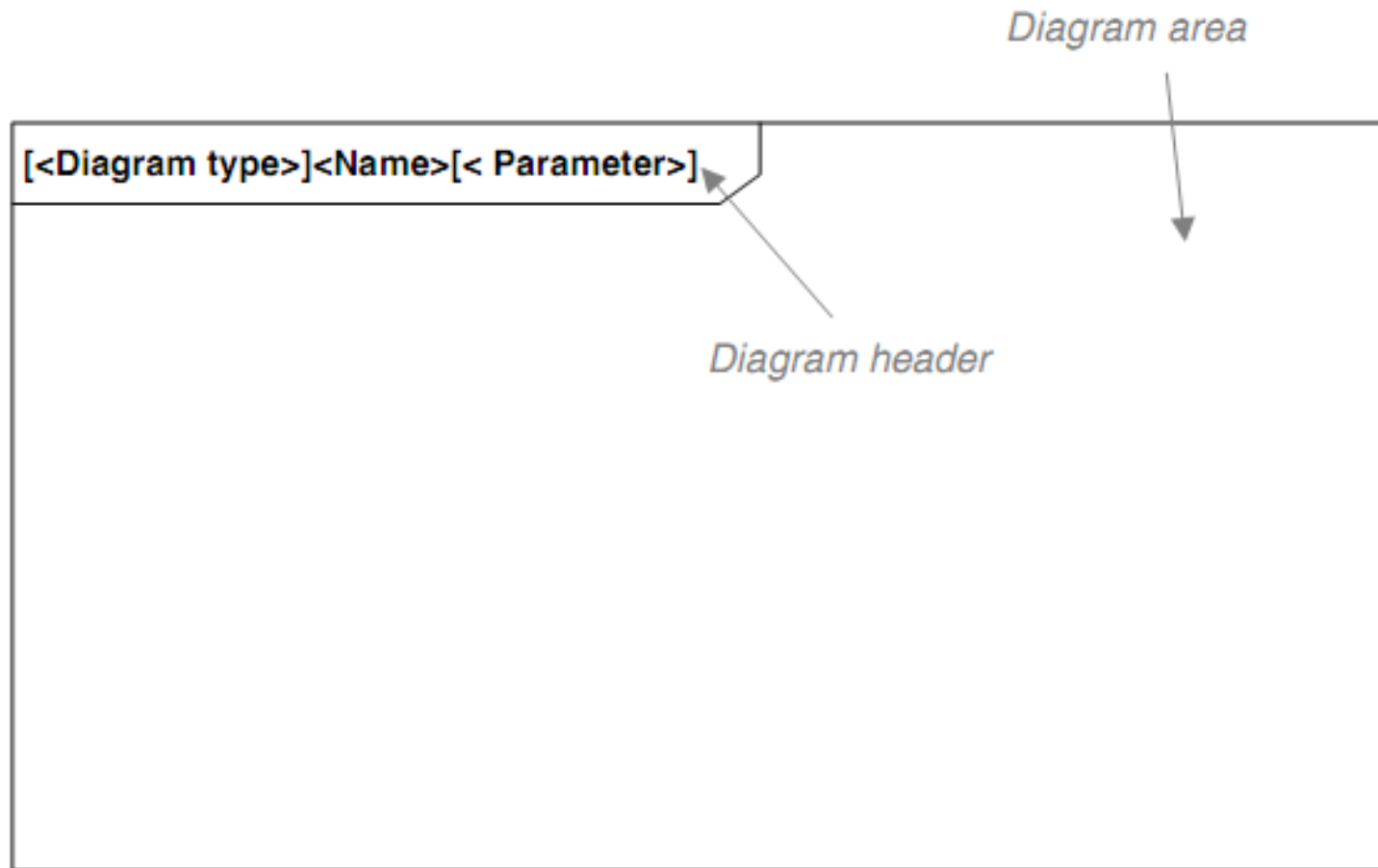
**FIGURE 2.3** Overview of the UML diagrams.



# Introducción a los diagramas en UML

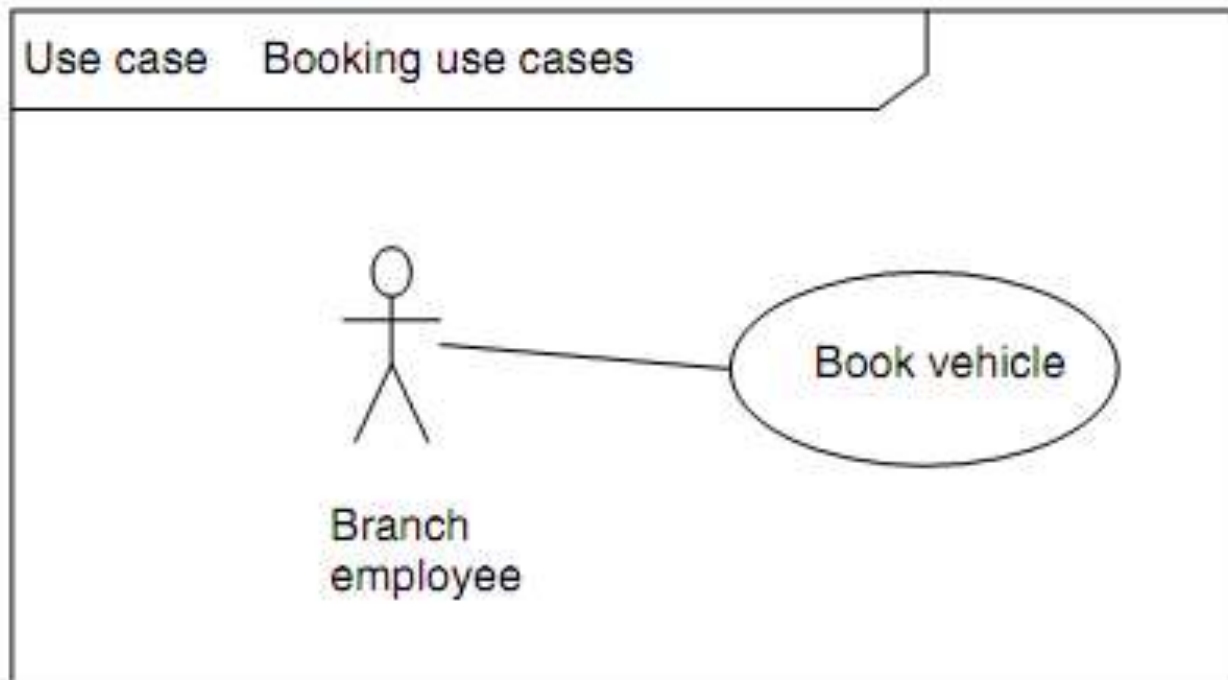
- Un diagrama consta de un área dentro de un rectángulo y un encabezado en la esquina superior izquierda
- El encabezado diagrama muestra el tipo de diagrama (opcional), el nombre del diagrama (obligatorio), y los parámetros (opcional).

# Introducción a los diagramas en UML



**FIGURE 2.4** Basic notation for diagrams.

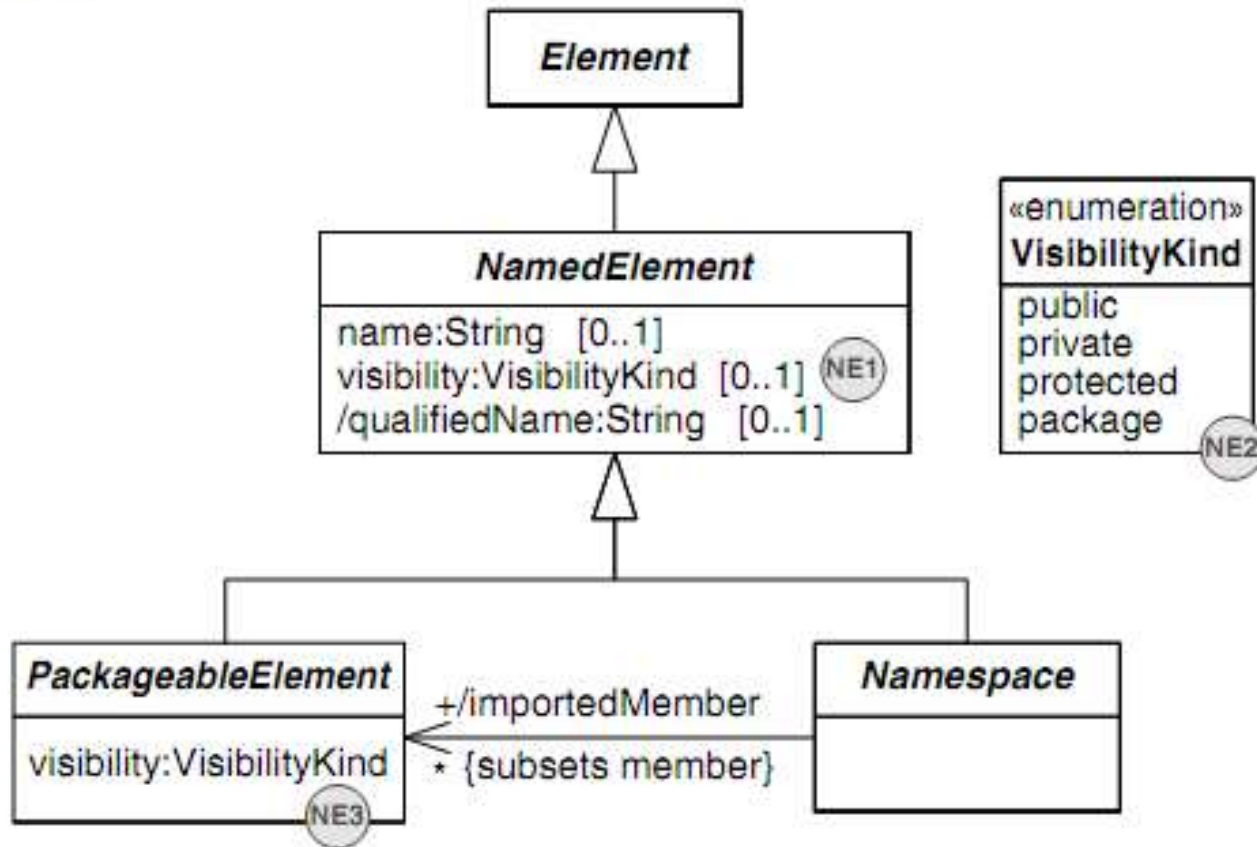
# Introducción a los diagramas en UML



**FIGURE 2.5** Example of a use case diagram.

# Tipos de elementos del UML

## Metamodel



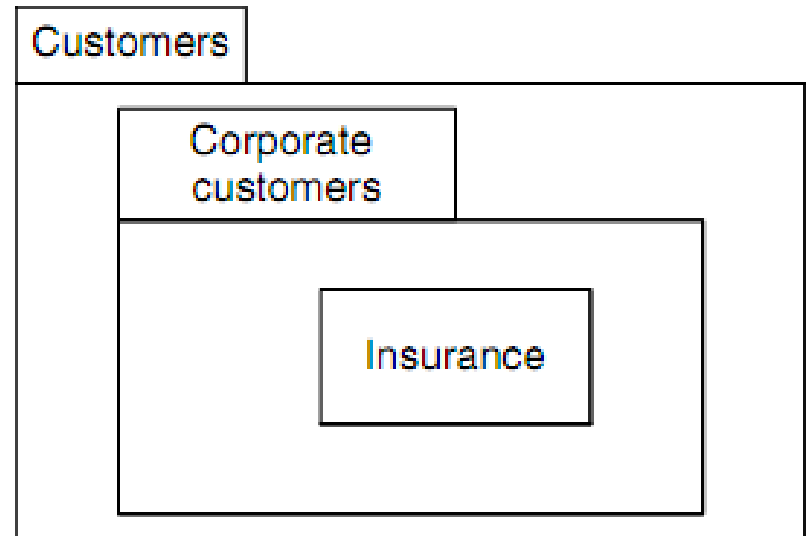
**FIGURE 2.15** The metamodel for NamedElement.

# Tipos de elementos del UML

- **Elemento nombrado** es un elemento que puede tener un nombre y una visibilidad definida (público, privado, protegido , empaquetado)

# Tipos de elementos del UML

- **Espacio de nombres (namespace)** es un elemento nombrado que puede contener elementos nombrados
  - Dentro de un espacio de nombres, elementos nombrados se identifican por sus nombres
    - Tienen un nombre cualificado, resultando en espacios de nombres anidados, por ejemplo, la clase Customers::CorporateCustomers::Insurance



# Tipos de elementos del UML

- **Elemento empaquetable** es un elemento nombrado que puede pertenecer directamente a un paquete.
  - La declaración de la visibilidad es obligatoria para un elemento empaquetable

+ = public

- = private

# = protected

~ = package

# Estereotipos

- Los estereotipos son extensiones formales de los elementos del modelo existente en el metamodelo UML, es decir, extensiones metamodelo
- El elemento de modelado está directamente influenciada por la semántica definida por la extensión
- En lugar de introducir un elemento nuevo del modelo al metamodelo, los estereotipos agregan semántica a un elemento actual del modelo



# Estereotipos

- Los Estereotipos clasifican los posibles usos de un elemento del modelo. Esta clasificación no tiene nada que ver con el modelado de metaclases.
- Ciertas características comunes son atribuidos a uno o varios elementos del modelo

# Estereotipos múltiples

- Diversos estereotipos se pueden utilizar para clasificar a un solo elemento de modelado
- La representación visual de un elemento puede estar influenciada por la asignación de estereotipos
- Los estereotipos se pueden añadir a los atributos, operaciones y relaciones
- Los estereotipos pueden tener atributos para almacenar información adicional

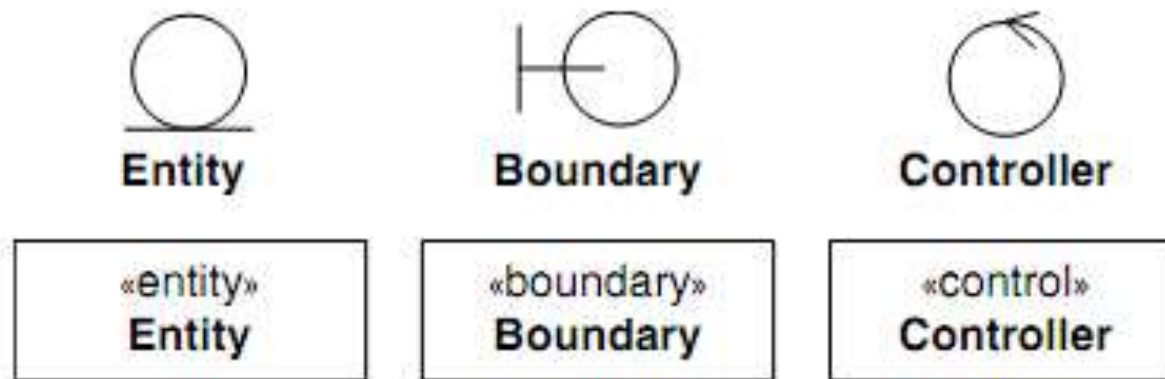
# Estereotipos - Notación

- Un estereotipo se coloca antes o por encima del nombre del elemento (por ejemplo, nombre de la clase) y encerrado en guillemets («,»)
- Los caracteres «y», a veces conocidos como comillas en ángulo, no debe confundirse con los caracteres doble mayor que, <<, o doble menor que, >>
- No todas las ocurrencias de esta notación significa que este viendo un estereotipo. Palabras claves predefinidas en UML también se encierran en guillemets

# Estereotipos – Símbolos Gráficos

- Como alternativa a esta notación puramente textual, símbolos especiales pueden ser utilizados
- Los estereotipos «entity» («entidad»), «boundary» («límite») y «control» son buenos ejemplos.
- Las herramientas son libres de uso de algún color especial u otro elemento de énfasis visual

# Estereotipos – Símbolos Gráficos



---

**FIGURE 2.6** Visual and textual stereotypes.

## UML Standard Stereotypes

Stereotype	UML Element	Description
«call»	Dependency (usage)	Call dependency between operations or classes.
«create»	Dependency (usage)	The source element creates instances of the target element.
«instantiate»	Dependency (usage)	The source element creates instances of the target element. <i>Note: This description is identical to the one of «create».</i>
«responsibility»	Dependency (usage)	The source element is responsible for the target element.
«send»	Dependency (usage)	The source element is an operation and the target element is a signal sent by that operation.
«derive»	Abstraction	The source element can, for instance, be derived from the target element by a calculation
«refine»	Abstraction	A refinement relationship (e.g., between a design element and a pertaining analysis element).
«trace»	Abstraction	Serves to trace of requirements.
«script»	Artifact	A script file (can be executed on a computer).
«auxiliary»	Class	Classes that support other classes («focus»).
«focus»	Class	Classes contain the primary logic. See «auxiliary».
«implementationClass»	Class	An implementation class specially designed for a programming language, where an object may belong to one class only.
«metaclass»	Class	A class with instances that are, in turn, classes.
«type»	Class	Types define a set of operations and attributes, and they are generally abstract.
«utility»	Class	Utility classes are collections of global variables and functions, which are grouped into a class, where they are defined as class attributes/operations.
«buildComponent»	Component	An organizationally motivated component.
«implement»	Component	A component that contains only implementation, no specification.

«buildComponent»	Component	An organizationally motivated component.
«implement»	Component	A component that contains only implementation, no specification.
«framework»	Package	A package that contains Framework elements.
«modelLibrary»	Package	A package that contains model elements, which are reused in other packages.
«create»	Behavioral feature	A property that creates instances of the class to which it belongs (e.g., constructor).
«destroy»	Behavioral feature	A property that destroys instances of the class to which it belongs (e.g., destructor).

**FIGURE 2.7** UML standard stereotypes (selection).