

# Orientación a Objetos

## Diseño

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

3 de diciembre de 2020



# Programación orientada a objetos

- 1 Programación orientada a objetos
- 2 Diseño orientado a objetos
- 3 UML

# Temas

- 1 Programación orientada a objetos
  - Definiciones
  - El modelo
  - Programación orientada a objetos con clases

# Programación orientada a objetos

## Definición

El paradigma<sup>a</sup> de programación orientada a objetos (POO) es un modelo de organización de programas y una colección de patrones y prácticas para diseñar programas; no se centra en el desarrollo de algoritmos Eliëns 1994.

---

<sup>a</sup>Para una discusión más detallada de lo que significa la palabra *paradigma* a secas, consultar Porto y Merino Publicado: 2008. Actualizado: 2012.

# Ventajas

Las ventajas que la programación orientada a objetos ofrece a la ingeniería de software son:

- Permite reutilizar el código, reduciendo el tiempo requerido para desarrollar software.
- Permite estructurar mejor los programas, de modo que sea más fácil entender qué hace cada rutina.
- Permite probar los programas más fácilmente. Un programa está integrado por pequeños componentes que pueden ser probados aisladamente.
- Los programas orientados a objetos son fáciles de extender cuando se requiere.

Hyman y Arnson 1998

# Características

Hay dos características que distinguen a la POO de otros paradigmas:

- El *encapsulamiento* está vinculado al concepto mismo de objeto y promueve la manipulación **local** de un grupo **pequeño** de datos **altamente correlacionados** entre sí.
- La *herencia* es un mecanismo de reutilización de código, basado en la definición de categorías de objetos, partiendo de las más generales, hacia las más especializadas.

# Temas

- 1 Programación orientada a objetos
  - Definiciones
  - El modelo
  - Programación orientada a objetos con clases

# El modelo

- En el modelo orientado a objetos el proceso de cómputo se lleva a cabo enviando *mensajes* entre *objetos*.
- Los mensajes se le envían a un objeto a través de sus métodos y los argumentos que reciben dichos métodos.
- Se le denomina *protocolo* a la colección de mensajes a los cuales puede responder un objeto.



# ¿Qué es un objeto?

## Definición (Objeto)

Un *objeto* es una *abstracción* que contiene **datos** y las **rutinas** que procesan esos datos.

Simultáneamente un *tipo de dato abstracto (TDA)* está consituido por:

- Un conjunto de datos y
- la definición de las operaciones que se pueden realizar con ellos.

# Datos

- Los objetos guardan sus datos en variables, conocidas como *atributos* o *variables de instancia*.
- Los valores guardados en estas variables pueden cambiar a lo largo de la ejecución de un programa, por lo que se hace necesario hablar del *estado del objeto*.

## Definición (Estado del objeto)

El *estado del objeto* son los valores que tienen sus atributos en un momento dado.

# Rutinas, métodos y mensajes

- En orientación a objetos se dice que los objetos se comunican enviándose *mensajes*.
- Cada objeto tiene un conjunto de mensajes que entiende y son los **únicos** que puede recibir.
- Al recibir un mensaje conocido ejecutará el *método* correspondiente.
- Un método contiene la rutina o secuencia de instrucciones que ejecutará el objeto.

# Tipos de métodos

Distinguiremos cinco tipos de rutinas asociadas a los objetos, a las cuales llamaremos *métodos* Viso y Peláez V. 2012:

- *Constructores*: sirven para establecer el estado inicial de un objeto.
- *De acceso*: permiten conocer el estado del objeto.
- *De modificación*: permiten modificar el estado del objeto.
- *De implementación*: son los que ofrecen los **servicios** especiales para los que fue diseñado el objeto.
- *Auxiliares*: Permiten al objeto organizar mejor sus tareas internamente y no proveen servicios al exterior.

# Temas

- 1 Programación orientada a objetos
  - Definiciones
  - El modelo
  - Programación orientada a objetos con clases

# Clases

## Definición (Clase)

Las *clases* son descripciones genéricas de grupos de objetos que comparten el mismo esquema: almacenan los **mismos tipos de datos** y pueden ejecutar las **mismas acciones**. Las clases especifican dos elementos:

- Atributos** Son un conjunto de variables, con sus respectivos tipos, que definen el conjunto de datos que procesará esta clase. Generalmente, la clase impedirá que código fuera de ella acceda directamente a estos datos.
- Métodos** Son funciones que operan sobre los datos de la clase y establecen la comunicación con código fuera de la clase.

- Objetos individuales son creados siguiendo la descripción proporcionada por la clase, pero diferentes objetos podrán encontrarse en diferentes estados.

# Interfaces

## Definición (Interfaz)

Las *interfaces* son **contratos** por medio de los cuales las clases se compromenten a implementar **protocolos** específicos.

- Es decir, las clases que implementen una interfaz deben contener todos los métodos especificados por dicha interfaz.
- En este sentido, las interfaces también definen **tipos abstractos de datos**.

# Diseño orientado a objetos

- 1 Programación orientada a objetos
- 2 Diseño orientado a objetos
- 3 UML



# Temas

- 2 Diseño orientado a objetos
  - Método
  - Tarjetas de responsabilidades

# Método

Dada la descripción de un problema, para diseñar una solución orientada a objetos, realizar los siguientes pasos:

- ➊ **Identificar los sustantivos.** Estos serán los candidatos a definir clases y objetos.
- ➋ **Agruparlos en clases.**
- ➌ **Determinar responsabilidades (verbos).**
- ➍ **Asigna responsabilidades de las clases.**

# Temas

- 2 Diseño orientado a objetos
  - Método
  - Tarjetas de responsabilidades

# Tarjetas de responsabilidades

Clase: <Nombre>

Atributos:

Campo	Acceso	Modificador(es)	tipo	Identificador	Descripción
		static, final			

Métodos:

Nombre	Responsabilidad

# UML

- 1 Programación orientada a objetos
- 2 Diseño orientado a objetos
- 3 UML

# Lenguaje de modelado unificado

- Utilizaremos algunos símbolos del *lenguaje de modelado unificado* (*Unified Modeling Language (UML)*) para representar las relaciones entre clases e interfaces.
- A los diagramas que utilizaremos se les llama *diagramas de clases*.

# Clases

Las clases se representan con cajas donde se pueden escribir:

- El nombre de la clase.
- Los atributos de la clase.
- Los métodos de la clase.

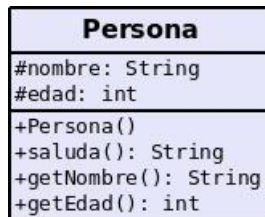


Figura: Ejemplo de diagrama de clase UML

# Accesos

- privado. Solamente los métodos de la clase pueden acceder a estos atributos o métodos.
- # protegido. La clase y clases que la extiendan, directa o indirectamente, pueden acceder a estos atributos.
- + público. Todos pueden llamar a estos métodos, leer y escribir directamente estos atributos. Se recomienda sólo utilizarlos cuando los atributos son `final`, es decir, su valor nunca cambia.



# Generalización/Herencia

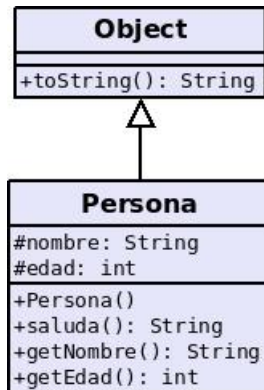


Figura: Ejemplo de generalización UML

# Interfaz



Figura: Ejemplo de una interfaz en UML

# Realización/Implementación

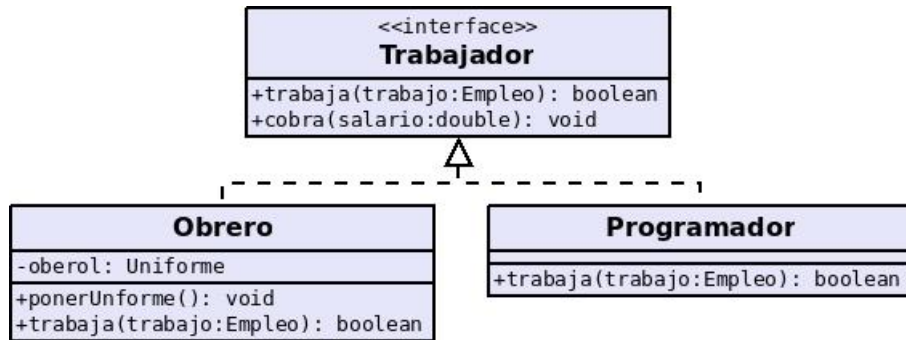
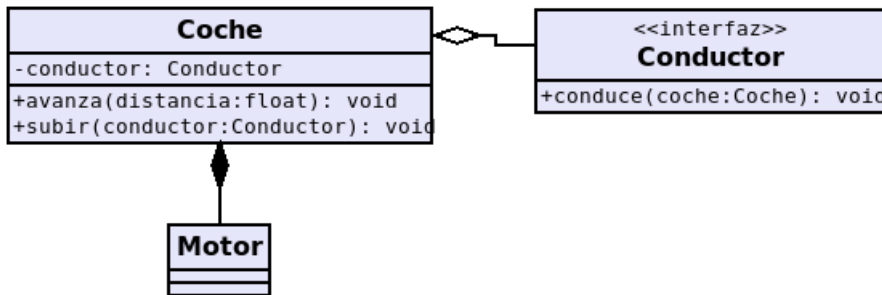


Figura: Ejemplo de realización UML

# Contención

**Agregación** El objeto contenido puede existir, aunque el contenedor deje de existir.

**Composición** El objeto contenedor está hecho de los objetos que contiene.



**Figura:** Ejemplos de contención UML. Agregación entre Coche y Conductor. Composición entre Coche y Motor.

# Dependencia y asociación

**Dependencia** indica que una clase depende del trabajo realizado por otra. Agregación y contención son casos particulares.



Figura: Dependencia.

**Asociación** cuando dos clases están relacionadas de cualquier forma, se dice que están asociadas.

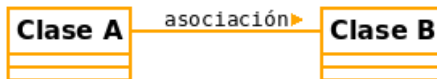






Figura: Asociación.

## Bibliografía I

-  Eliëns, Anton (1994). *Object-Oriented Software Development*. Addison Wesley.
-  Hyman, Michael y Bob Arnsen (1998). *Visual C++ 6 for Dummies*. IDG Books Worldwide, Inc.
-  Porto, Julián Pérez y María Merino (Publicado: 2008. Actualizado: 2012.). *Definición de Paradigma*. URL: <http://definicion.de/paradigma/>.
-  Viso, Elisa y Canek Peláez V. (2012). *Introducción a las ciencias de la computación con Java*. 2a. Temas de computación. Las prensas de ciencias. 571 págs. ISBN: 978-607-02-3345-6.

# Licencia

Creative Commons  
Atribución-No Comercial-Compartir Igual

