



# POLIMORFISMO Y REUTILIZACIÓN

## PARADIGMAS DE PROGRAMACIÓN II

---

Carlos Rojas Sánchez

Licenciatura en Informática

Universidad del Mar

1. Concepto del polimorfismo
2. Sobrecarga
3. Sobreescritura
4. Variables Polimórficas

# Concepto del polimorfismo

---

# Concepto del polimorfismo

El polimorfismo es el modo en que los lenguajes OO implementan el concepto de polisemia del mundo real: Un único nombre para muchos significados, según el contexto.

# Concepto del polimorfismo

- Capacidad de una entidad de referenciar distintos elementos en distintos instantes de tiempo.
- El polimorfismo nos permite programar de manera general en lugar de programar de manera específica.

# Concepto del polimorfismo

Hay cuatro técnicas, cada una de las cuales permite una forma distinta de reutilización de software, que facilita a su vez el desarrollo rápido, la confianza y la facilidad de uso y mantenimiento.

- Sobrecarga
- Sobreescritura
- Variables polimórficas
- Genericidad

**Sobrecarga** (Overloading, Polimorfismo ad-hoc): un solo nombre de método y muchas implementaciones distintas. Las funciones sobrecargadas normalmente se distinguen en tiempo de compilación por tener distintos parámetros de entrada y/o salida.

**Sobreescritura** (Overriding, Polimorfismo de inclusión): Tipo especial de sobrecarga que ocurre dentro de relaciones de herencia. En este caso la signatura es la misma (refinamiento o reemplazo del método del padre) pero los métodos se encuentran en dos clases distintas relacionadas mediante herencia.



**Variables polimórficas** (Polimorfismo de asignación): variable que se declara como de un tipo pero que referencia en realidad un valor de un tipo distinto. Cuando una variable polimórfica se utiliza como argumento, la función resultante se dice que exhibe un polimorfismo puro.

**Genericidad** (plantillas o templates): forma de crear herramientas de propósito general (clases, métodos) y especializarlas para situaciones específicas.

# Tipos de polimorfismo

- Sobrecarga

Factura::imprimir()

Factura::imprimir(int numCopias)

ListaCompra::imprimir()

- Sobreescritura

Cuenta::abonarInteres()

CuentaJoven::abonarInteres()

- Variables polimórficas

Cuenta \*pc=new CuentaJoven();

- Genericidad

Lista<Cliente> Lista<Articulo> Lista<Alumno>

Lista<Habitacion>

# Sobrecarga

---

## Sobrecarga (Overloading, polimorfismo ad-hoc )

- Un mismo nombre de mensaje asociado a varias implementaciones.
- La sobrecarga se realiza en tiempo de compilación (enlace estático) en función de la signature completa del mensaje.

# Sobrecarga (Overloading, polimorfismo ad-hoc )

Dos tipos de sobrecarga:

1. Basada en ámbito: Métodos con diferentes ámbitos de definición, independientemente de sus signatures de tipo. Permitido en todos los lenguajes OO.
  - Un mismo método puede ser utilizado en dos o más clases.
  - P. ej. Sobrecarga de operadores como funciones miembro.

# Sobrecarga (Overloading, polimorfismo ad-hoc )

Dos tipos de sobrecarga:

2. Basada en signatura: Métodos con diferentes signaturas de tipo en el mismo ámbito de definición. No permitido en todos los lenguajes OO.
  - P. ej. Cualquier conjunto de funciones no miembro (en el ámbito de definición global) que comparten nombre.
  - Dos o más métodos en la misma clase pueden tener el mismo nombre siempre que tengan distinta signatura de tipos.

## Sobrecarga basada en ámbito

- Distintos ámbitos implican que el mismo nombre de método puede aparecer en ellos sin ambigüedad ni pérdida de precisión.
- La sobrecarga por ámbito no requiere que las funciones asociadas con un nombre sobrecargado tengan ninguna similitud semántica, ni la misma signatura de tipo.



# Sobrecarga basada en firmas de tipo

- Métodos en el mismo ámbito pueden compartir el mismo nombre siempre que difieran en número, orden y, en lenguajes con tipado estático, el tipo de los argumentos que requieren.
- No todos los lenguajes OO permiten la sobrecarga:
  - Permiten sobrecarga de métodos y operadores: C++
  - Permiten sobrecarga de métodos pero no de operadores: Java, Python, Perl
  - Permiten sobrecarga de operadores pero no de métodos: Eiffel

# Polimorfismo en jerarquías de herencia

## Sobrecarga en jerarquías de herencia

- Métodos con el mismo nombre, la misma signature de tipo y enlace estático: Shadowing (refinamiento/reemplazo): las signatures de tipo son las mismas en clases padre e hijas, pero el método a invocar se decide en tiempo de compilación
- Métodos con el mismo nombre y distinta signature de tipo y enlace estático: Redefinición: clase hija define un método con el mismo nombre que el padre pero con distinta signature de tipos. Modelo MERGE: SOBRECARGA y Modelo JERÁRQUICO: NO HAY SOBRECARGA

# Sobreescritura

---

# Polimorfismo en jerarquías de herencia

## Sobreescritura (Overriding, polimorfismo de inclusión)

- Decimos que un método en una clase derivada sobreescribe un método en la clase base si los dos métodos tienen el mismo nombre, la misma signatura de tipos y enlace dinámico.
- La sobreescritura es importante cuando se combina con el principio de sustitución.
- En algunos lenguajes (Java, Smalltalk) la simple existencia de un método con el mismo nombre y signatura de tipos en clase base y derivada indica sobreescritura.

## Polimorfismo: Sobreescritura, Shadowing y Redefinición

- Sobreescritura: la signatura de tipo para el mensaje es la misma en clase base y derivada, pero el método se enlaza con la llamada en tiempo de ejecución.
- Shadowing: la signatura de tipo para el mensaje es la misma en clase base y derivada, pero el método se enlaza en tiempo de compilación (en función del tipo de la variable receptora).
- Redefinición: La clase derivada define un método con el mismo nombre que en la clase base y con distinta signatura de tipos.

# Variables Polimórficas

---

## Variables Polimórficas (Polimorfismo de asignación)

- Una variable polimórfica es aquélla que puede referenciar más de un tipo de objeto. Puede mantener valores de distintos tipos en distintos momentos de ejecución del programa.
- En un lenguaje débilmente tipado todas las variables son potencialmente polimórficas.
- En un lenguaje fuertemente tipado la variable polimórfica es la materialización del principio de sustitución.

## Ventajas

- El polimorfismo hace posible que un usuario pueda añadir nuevas clases a una jerarquía sin modificar o recompilar el código original.
- Permite programar a nivel de clase base utilizando objetos de clases derivadas (posiblemente no definidas aún): Técnica base de las librerías/frameworks.



- Estableces clase base.
- Defines nuevas variables y funciones.
- Ensamblas con el código objeto que tenías.
- Los métodos de la clase base pueden ser reutilizados con variables y parámetros de la clase derivada.