DEVS4J

Tendencias: Español · English

Programación orientada a objetos

RAIDENTRANCE HACE 2 DÍAS DEJA UN COMENTARIO

1 Vote

¿Qué es un paradigma de programación?

Define un **estilo de programación**, esto no se refiere a ningún lenguaje de programación en específico, existen múltiples estrategias o formas de resolver un problema, a estas se les conoce como paradigmas, a continuación algunos de los más populares:

- Programación orientada a eventos
- Programación orientada a objetos
- Programación orientada a aspectos
- Programación funcional
- Programación imperativa
- Programación funcional

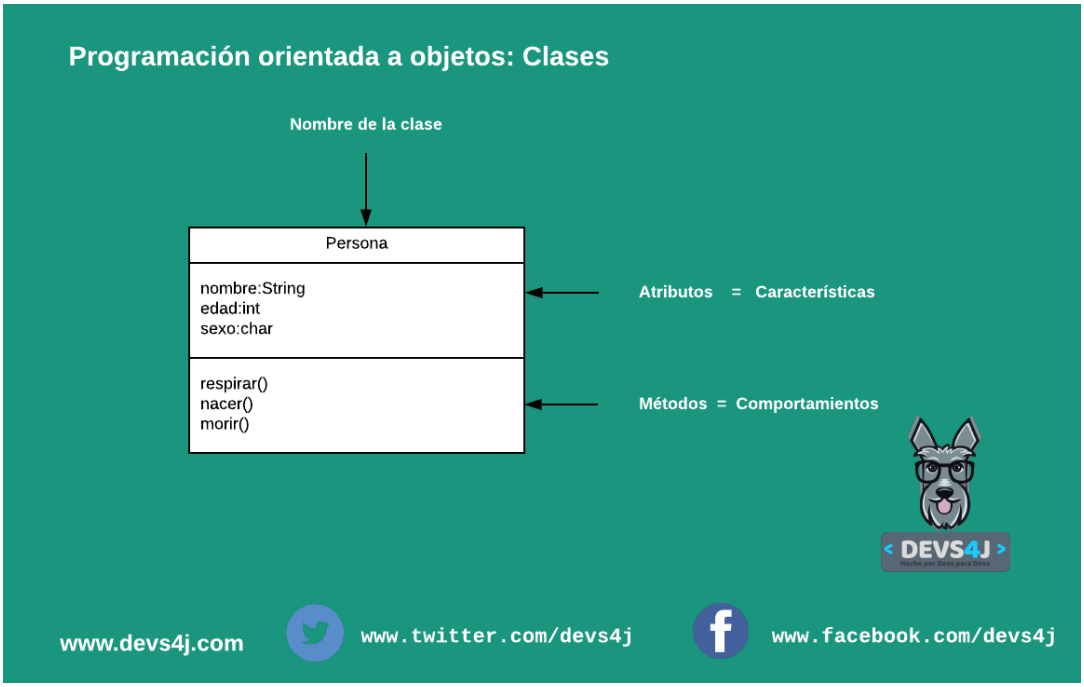
Cada lenguaje de programación tiene soporte para uno o más de estos paradigmas, por esto que es importante que conozcas los conozcas y aprendas sus conceptos principios y reglas.

¿Qué es la programación orientada a objetos?

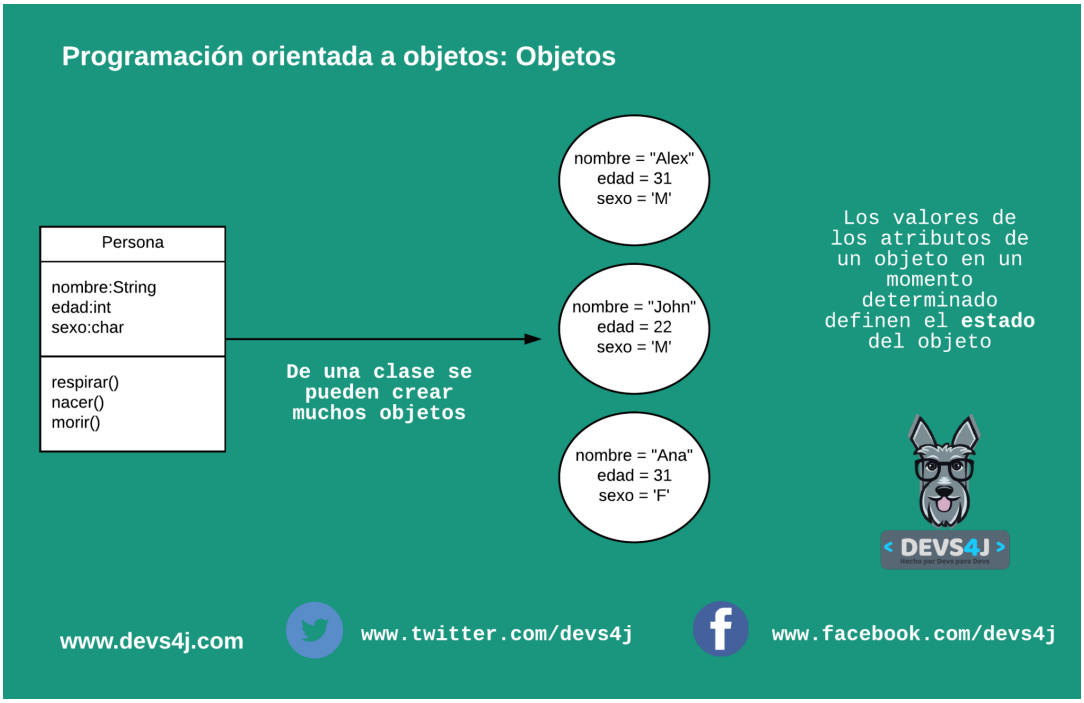
Es el paradigma principal soportado por Java basado principalmente en interacción entre objetos, a continuación se listan sus principales conceptos, principios y reglas.

Conceptos

- **Clases**: Cuando se escribe código en Java se escribe sobre clases o interfaces, en las clases es posible definir :
  - Atributos : Las características se definen a través de atributos, pueden ser definidos a través de tipos de datos primitivos o a través de otros objetos.
  - Métodos : Los comportamientos son definidos a través de métodos.

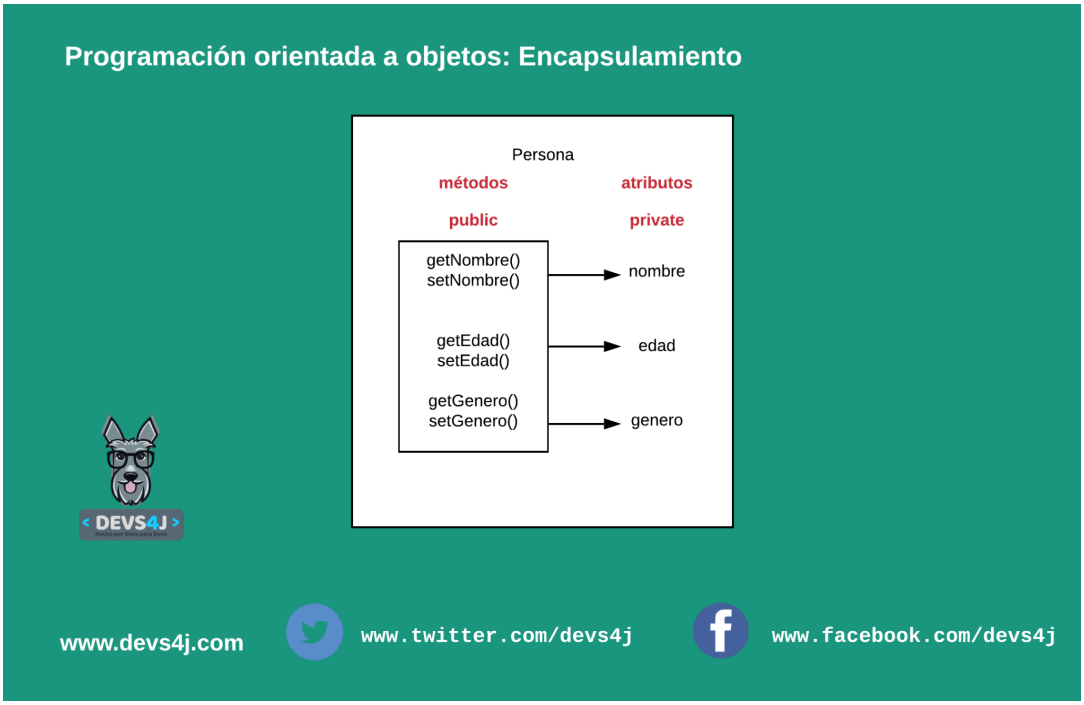


- **Objetos**: Los objetos se crean a través de clases y mantiene dos cosas principalmente estado y comportamiento. Una clase se puede utilizar para crear muchos objetos.



Principios

- **Encapsulamiento**: Define la interacción entre los objetos, se consigue cuando los objetos mantienen su **estado** privado y exponen métodos que permiten modificarlo.



- **Abstracción**: La abstracción es una extensión del encapsulamiento y define que un objeto solo debe exponer los mecanismos necesarios para utilizarlo y **ocultar los detalles de implementación**. Si pensamos en un objeto de tipo "Auto" que será utilizado por un objeto de tipo "Persona", el auto solo deberá exponer comportamientos como "encender", "acelerar", "frenar", "cambiar velocidad", etc. y no los comportamientos específicos como "pasar corriente", "iniciar marcha", "verificar niveles", etc.

- **Herencia**: Algunas veces los objetos son muy similares entre ellos pero no completamente iguales, la herencia permite compartir algunas funcionalidades entre los objetos a través de clases padres e hijas, existe una regla que nos ayuda a determinar

cuando se puede aplicar herencia llamada “IS A (ES UN)” a continuación algunos ejemplos de su uso:

- Coche **ES UN** Vehículo – SI – Coche puede heredar de Vehículo
- Avión **ES UN** Vehículo – SI – Avión puede heredar de Vehículo
- Coche **ES UN** Avión – NO – Coche NO puede heredar de Avión
- Animal **ES UN** Perro – NO – Animal NO puede heredar de Perro
- Perro **ES UN** Animal – SI – Perro puede heredar de Animal

- **Polimorfismo**: Polimorfismo significa muchas formas y está muy relacionado con el concepto de herencia, pensemos en el siguiente ejemplo, se tiene la clase **Animal** y múltiples subclases **Perro**, **Ave**, **Gato**, **Águila**, etc. El polimorfismo nos permite lo siguiente:
  - Animal **ES UN** Animal – Entonces es posible hacer Animal a=new Animal();
  - Perro **ES UN** Animal – Entonces es posible hacer Animal a=new Perro();
  - Ave **ES UN** Animal – Entonces es posible hacer Animal a=new Ave();
  - Águila **ES UN** Animal – Entonces es posible hacer Animal a=new Águila();
  - Águila **ES UN** Ave – Entonces es posible hacer Ave a=new Aguila();
- De igual modo es posible hacer uso de arreglos y colecciones utilizando este concepto, a continuación un ejemplo:
  - Animal array[]={new Perro(), new Ave(), new Aguila(), new Gato()};
- La limitante que se tiene es que solo se podrán invocar los métodos definidos en la referencia, en el ejemplo los métodos definidos en la clase animal.

Si te gusta el contenido y quieres enterarte cuando realicemos un post nuevo síguenos en nuestras redes sociales.

Autor: Alejandro Agapito Bautista

Twitter: @raidentrance (https://geeksjavamexico.wordpress.com/mentions/raidentrance/)

Publicado por raidentrance

Soy @raidentrance en Twitter y en Github, soy egresado de la Facultad de Ingeniería de la UNAM, cuento con 8 certificaciones en diferentes áreas del desarrollo de software, me gustan las cervezas y soy Geek.

Ver todas las entradas de raidentrance (https://devs4j.com/author/raidentrance/)

Español

java