Java For Testers

Capítulo 2
Introducción a Java

Fecha: 01/09/2025

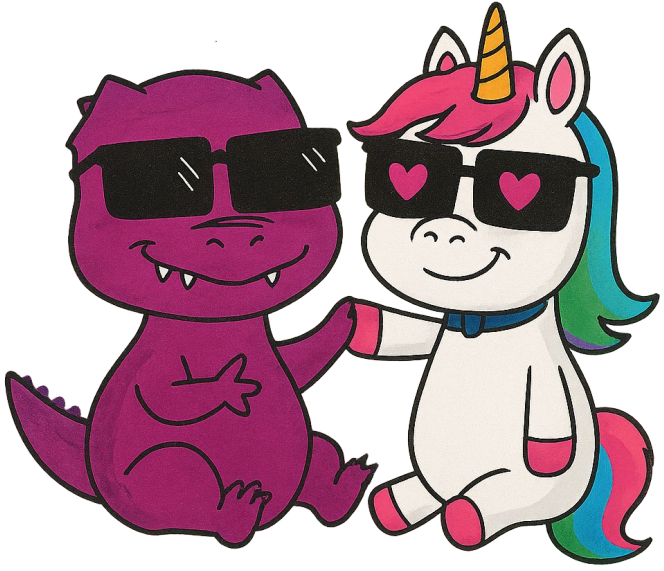
2 Introducción a Java

1. ¿Qué es Java?
2. JDK, JRE, JVM
3. POO
    - Encapsulamiento
    - Herencia
    - Abstracción
    - Poliformismo

2. Introducción

Java

★ **Lenguaje de programación:** poderoso y sirve para construir software, aplicaciones Web, Mobile, API



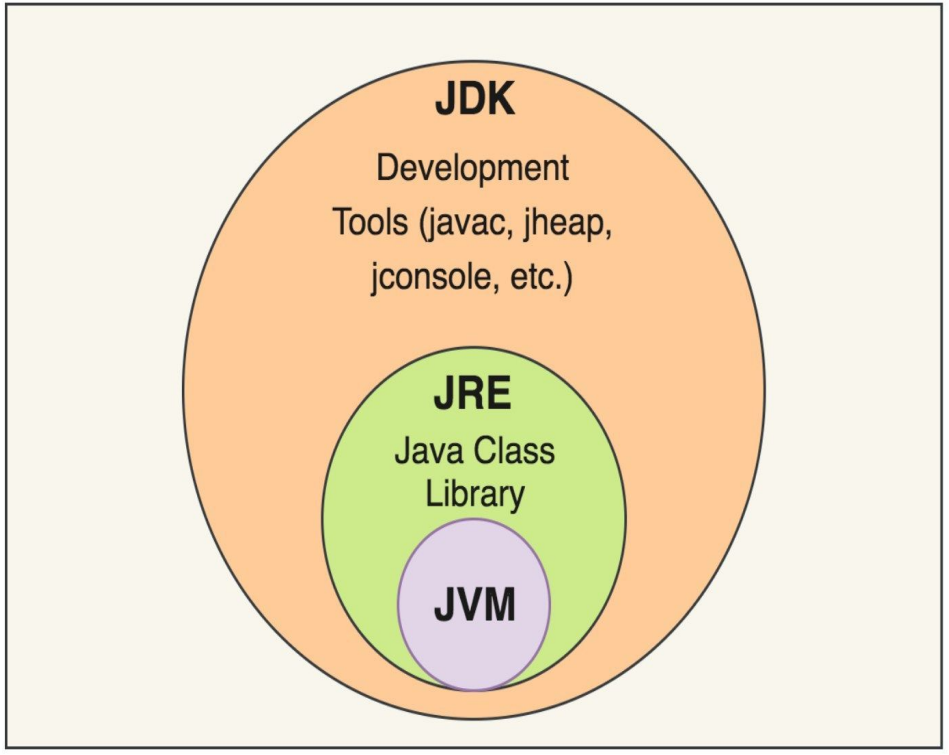
Por qué un Tester debería saber Java

★ **Errores**: Es probable que encuentres mensajes de error o fallos relacionados con Java. En un futuro ser un code reviewer

★ **Automatización**: Algunas herramientas de pruebas automatizadas (Selenium, Appium) se crean con Java

★ **Comunicación**: Si entiendes lo básico de Java, podrás hablar mejor con los desarrolladores cuando reportes problemas o pidas aclaraciones.

- ★ **JDK**
  Java Development Kit
- ★ **JRE**
  Java Runtime Environment
- ★ **JVM**
  Java Virtual Machine

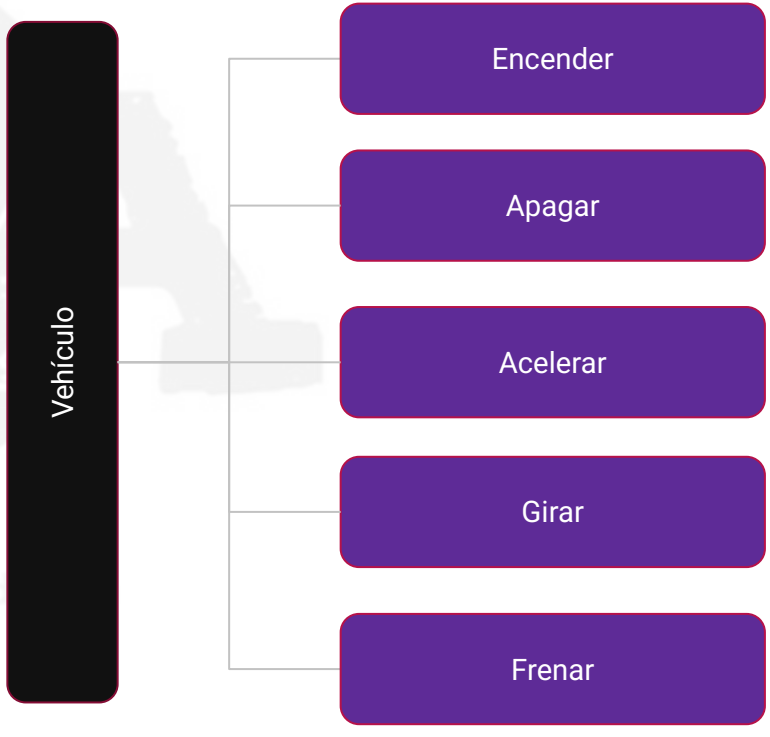


- Abstracción
- Encapsulamiento
- Herencia
- Polimorfismo

Abstracción

- ★ Consiste en simplificar la complejidad al enfocarse en los aspectos esenciales de los objetos, ignorando los detalles no relevantes.
- ★ Es como hacer un modelo o una representación simplificada del objeto real, destacando sólo lo que importa para el propósito específico.

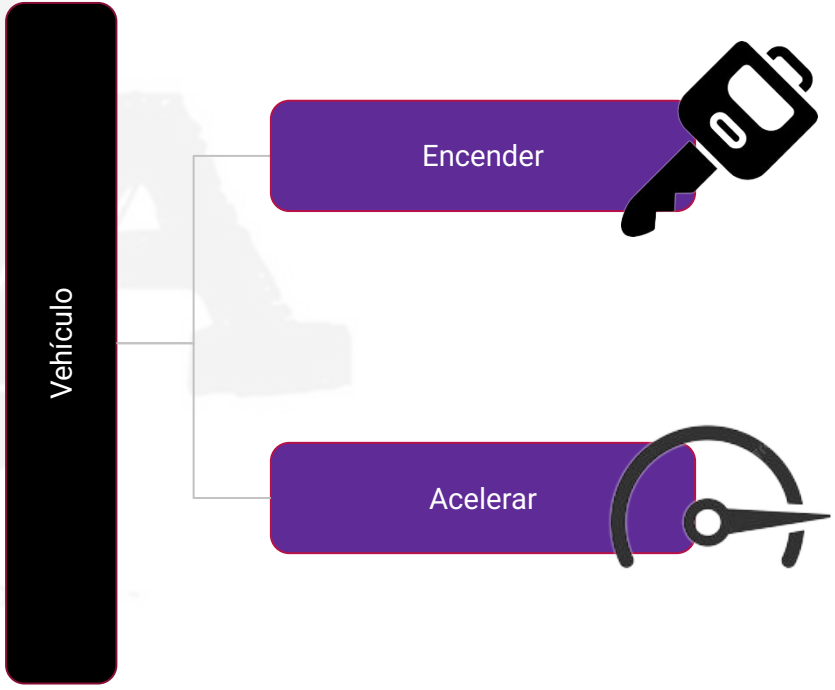
Ejemplo: Si pensamos en un "VEHÍCULO", solo nos importa que tenga funciones como encender, apagar, acelerar, frenar y girar, sin necesidad de conocer cómo funciona cada componente interno del motor.



Encapsulamiento

- ★ Se refiere a ocultar los detalles internos de un objeto y permitir que se acceda o modifique solo a través de métodos controlados.
- ★ Esto ayuda a proteger los datos del objeto y a controlar cómo interactúan con él.

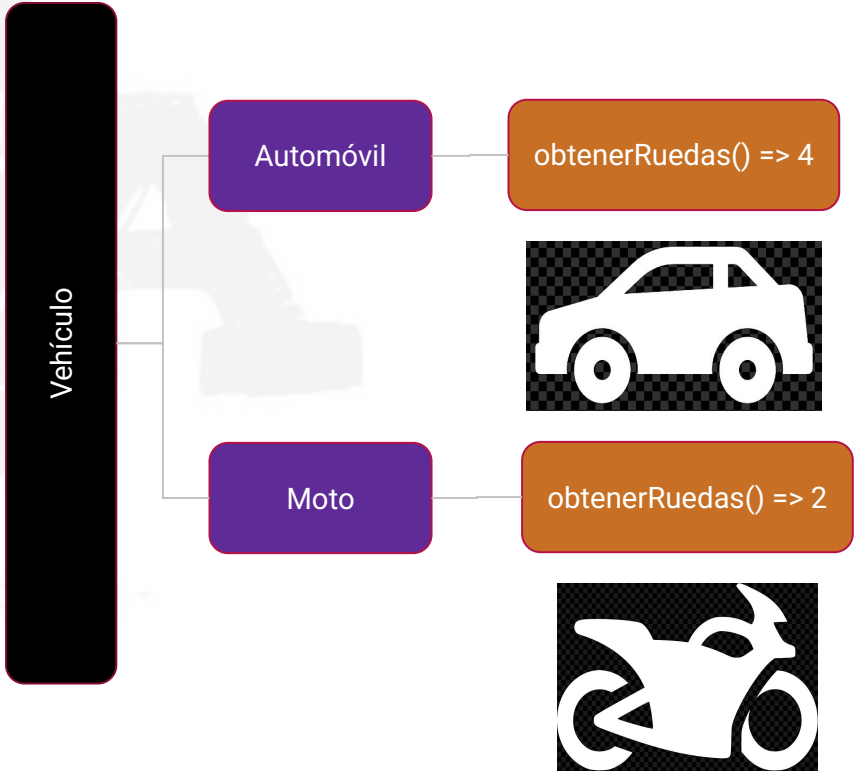
Ejemplo: Un coche tiene un motor, pero como usuario solo tienes acceso a encenderlo y apagarlo mediante un botón o una llave, no directamente al motor. Los detalles internos del motor están ocultos (encapsulados).



Herencia

- ★ Una clase hija hereda las propiedades y métodos de su clase padre, lo que promueve la reutilización del código y la organización jerárquica.

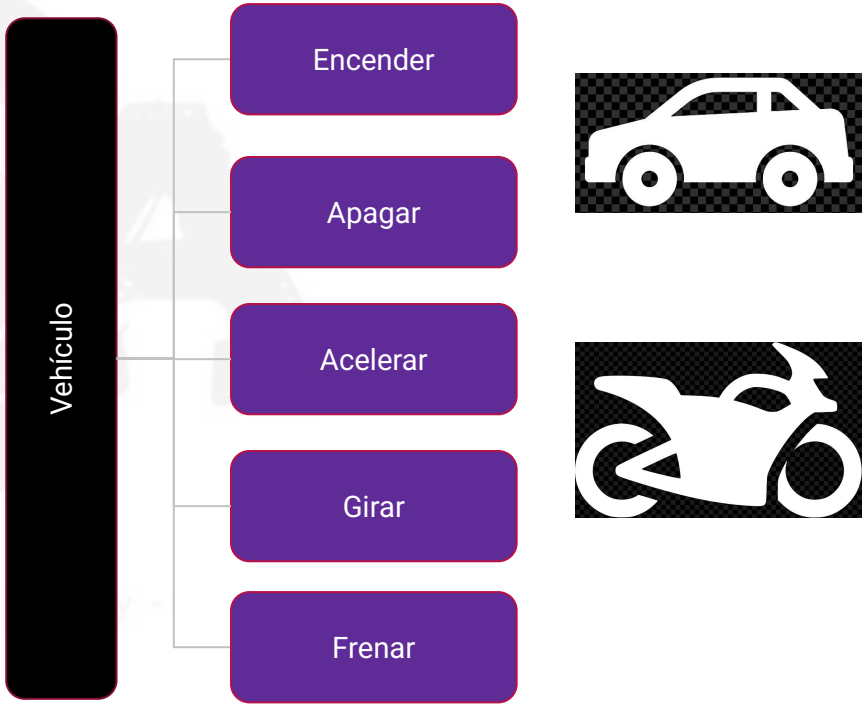
Ejemplo: Si tienes una clase "Vehículo", puedes crear una clase "Automóvil" una clase “Moto”, que estas hereden de "Vehículo", aprovechando las características generales de un Vehículo (como moverse) y añadiendo otras específicas del Automóvil (como tener 4 ruedas) o la Moto (como tener 2 ruedas).



Polimorfismo

- ★ Permite que un mismo método o función pueda tener diferentes comportamientos según el objeto que lo esté utilizando.
- ★ Esto significa que diferentes objetos pueden responder de manera distinta al mismo mensaje o acción.

Ejemplo: Si tienes una función “encender()” en la clase "Vehículo", un automóvil, una moto y un avión (todos heredan de Vehículo) podrían implementar la función "encender()" de manera diferente, aunque para el usuario final la acción sea la misma.



Quizz

Quizz

Realiza los siguientes

- ★ Quiz 01
- ★ Quiz 02

Recuerda:

- ★ Trata de no memorizar orden de las respuestas, entiende, asimila y aprende
- ★ En la página del curso, todos los capítulos tienen su acceso directo a los quizzes

Recursos del curso

Recursos

En esta sección encontrarás recursos más importantes del curso

- ★ Página del curso
- ★ Repositorio de Git
- ★ YouTube videos del curso

Si deseas cursos personalizados u otro servicio no dudes en contactarnos

https://testeracademy.com.mx/contacto

Gracias

🧠 “No aprendemos para aprobar pruebas, aprendemos para transformar realidades.”

👋 ¡Hasta pronto!