



Recibe una cálida:

# ¡Bienvenida!

---

Te estábamos esperando 😊 

# ➤ Pruebas Unitarias en Java

---

**Plan formativo:** Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

# HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



# REPASO CLASE ANTERIOR

En la clase anterior trabajamos :

- ✓ Características de los Test Unitarios
- ✓ Ventajas y desventajas de los Test Unitarios

# LEARNING PATHWAY

4.7

Start! 🏁

Pruebas Unitarias en  
Java

Introducción a JUnit

JUnit

Aplicando la teoría

# OBJETIVOS DE APRENDIZAJE

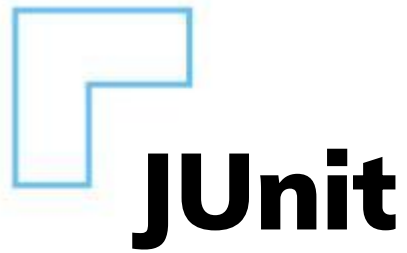
¿Qué aprenderemos?



**Comprender el concepto e implementación de la herramienta JUnit para Test Unitarios en Java**



# ➤ Introducción a JUnit



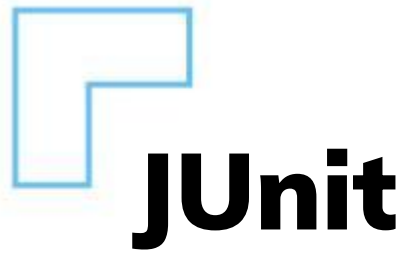
## ¿Qué es?:

JUnit es un marco de prueba unitaria para Java que proporciona estructura y herramientas para escribir y ejecutar pruebas unitarias de manera automatizada. JUnit simplifica la creación y ejecución de pruebas, facilitando la identificación temprana de errores y la mejora de la calidad del código.

JUnit es ampliamente utilizado en la comunidad de desarrollo de Java y se ha convertido en una herramienta estándar para la escritura de pruebas unitarias. Permite a los desarrolladores definir casos de prueba, ejecutarlos automáticamente y generar informes detallados sobre el estado de las pruebas.







### Ciclo de Vida de una Prueba:

JUnit sigue un ciclo de vida para las pruebas: **configuración** (setup), **ejecución** de la prueba y **limpieza** (teardown).

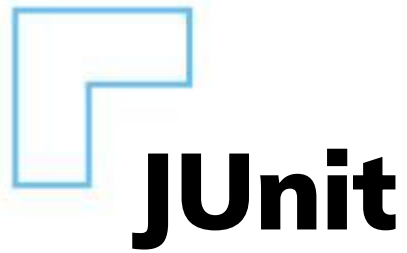
La configuración y la limpieza se realizan antes y después de cada prueba para garantizar que estén en un estado conocido.

### Mensajes de Error Significativos:

Cuando una prueba falla, JUnit proporciona mensajes de error significativos que ayudan a identificar la causa del fallo.

Esto facilita la depuración y corrección de problemas.





### **Herramientas de IDE:**

Muchas herramientas de desarrollo integrado (IDE) como Eclipse, IntelliJ IDEA y NetBeans ofrecen soporte integrado para JUnit. Esto facilita la creación, ejecución y análisis de pruebas desde el entorno de desarrollo.

### **Cobertura de Código:**

Algunas herramientas y complementos de JUnit pueden calcular la cobertura de código, lo que te permite saber cuántas líneas de código están siendo probadas por tus pruebas.

### **Pruebas Continuas (CI/CD):**

Las pruebas unitarias con JUnit son esenciales en las prácticas de integración continua (CI) y entrega continua (CD), ya que ayudan a garantizar que el código nuevo no rompa funcionalidades existentes.

# JUnit



## ¿Qué es necesario conocer antes de escribir Test con JUnit?

**Programación en Java:** Es fundamental tener un buen entendimiento de los conceptos básicos de programación en Java, como variables, tipos de datos, estructuras de control (if, bucles, etc.), clases, métodos y excepciones.

**POO:** Dado que JUnit se utiliza para probar unidades de código a nivel de clase o método, es importante tener conocimientos de programación orientada a objetos.

**Método de prueba:** Antes de utilizar JUnit, es beneficioso entender qué es un método de prueba y cómo funciona. Un método de prueba es una función que verifica el comportamiento esperado de una unidad de código específica. Debes comprender cómo escribir métodos de prueba y cómo verificar resultados utilizando aserciones (assertions).



# JUnit



## ¿Qué es necesario conocer antes de escribir Test con JUnit?

**Anotaciones:** JUnit utiliza anotaciones Java para marcar los métodos de prueba y definir su comportamiento. Es importante comprender cómo funcionan las anotaciones en Java y cómo se aplican en el contexto de JUnit. Algunas anotaciones clave en JUnit son **@Test**, **@BeforeEach**, **@AfterEach**, **@BeforeAll** y **@AfterAll**, conceptos que más adelante explicaremos.

**Aserciones (assertions):** Las aserciones son declaraciones que se utilizan para verificar si una condición es verdadera o falsa. JUnit5 proporciona un conjunto de métodos de aserción que se utilizan para evaluar y validar resultados en las pruebas unitarias. Es importante conocer los diferentes métodos de aserción disponibles en JUnit5 y cómo utilizarlos correctamente.



# Evaluación Integradora ✨

¿Listos para un nuevo desafío? En esta clase comenzamos a construir nuestro...

## Trabajo Integrador del Módulo 💪

Iremos completándolo progresivamente clase a clase.



# LIVE CODING


Ejemplo en vivo



*Vamos a crear las primeras clases Test para el proyecto de BilleteraVirtual:*

- 1. Crear una clase de pruebas CuentaTest para probar la clase Cuenta. ✦*
- 2. Crear una clase de pruebas TransaccionTest para probar la clase Transaccion.*
- 3. Crear una clase de pruebas MonedaTest para probar la interfaz Moneda.*
- 4. Explicar el uso de **assertEquals** para verificar el resultado.*



 **Tiempo: 30 minutos**





# **Ejercicio N° 1**

# **Aplicando la teoría**





# Aplicando la teoría



**A responder levantando la mano o en el chat (justifica la respuesta): 🙌**

Vamos a responder con V o F las siguientes afirmaciones:

1. JUnit sirve para crear pruebas unitarias automatizadas. (V)
2. Las clases de prueba deben llamarse IgualQueLaClaseOriginalTest. (F)
3. JUnit solo sirve para probar código Java. (F)
4. Las pruebas se crean en métodos anotados con @Test. (V)
5. JUnit permite aislar el código a probar de dependencias externas. (V)
6. assertEquals() compara resultados esperados vs actuales. (V)



**Tiempo** 🕒: 10 minutos





# Aplicando la teoría

## A investigar: 🙌

En los próximos 15 minutos investiguen sobre las siguientes anotaciones y su función principal en los test. Tendremos una puesta en común al finalizar el ejercicio!

1. @Test
2. @Before
3. @After
4. @BeforeClass
5. @AfterClass
6. @Ignore

**Tiempo** 🕒: 15 minutos



○

# ¿Alguna consulta?

+



# RESUMEN

¿Qué logramos en esta clase?

- ✓ **Comprender el concepto e implementación de la herramienta JUnit para Test Unitarios en Java**



# #WorkingTime

Continuemos ejercitando

¡**Antes de cerrar la clase!** Te invitamos a: 📌📌📌

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
  - a. *Lectura Módulo 4, Lección 7: páginas 5 - 6*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

# ¡Muchas Gracias!

Nos vemos en la próxima clase 🙌



Momento: ✚

# Time-out!

🕒 5 min.

