



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 :

- ✓ Implementación de mocks en programación orientada a objetos (Mockito)

# LEARNING PATHWAY

Nº de la unidad . ¿Sobre qué temas trabajaremos?

4.7

Start! 🏁

**Pruebas Unitarias en  
Java**

Suites de pruebas

En Suite

Suites de pruebas

# OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?



**Comprender el concepto e implementación de una Suite de Pruebas (Test Suite)**



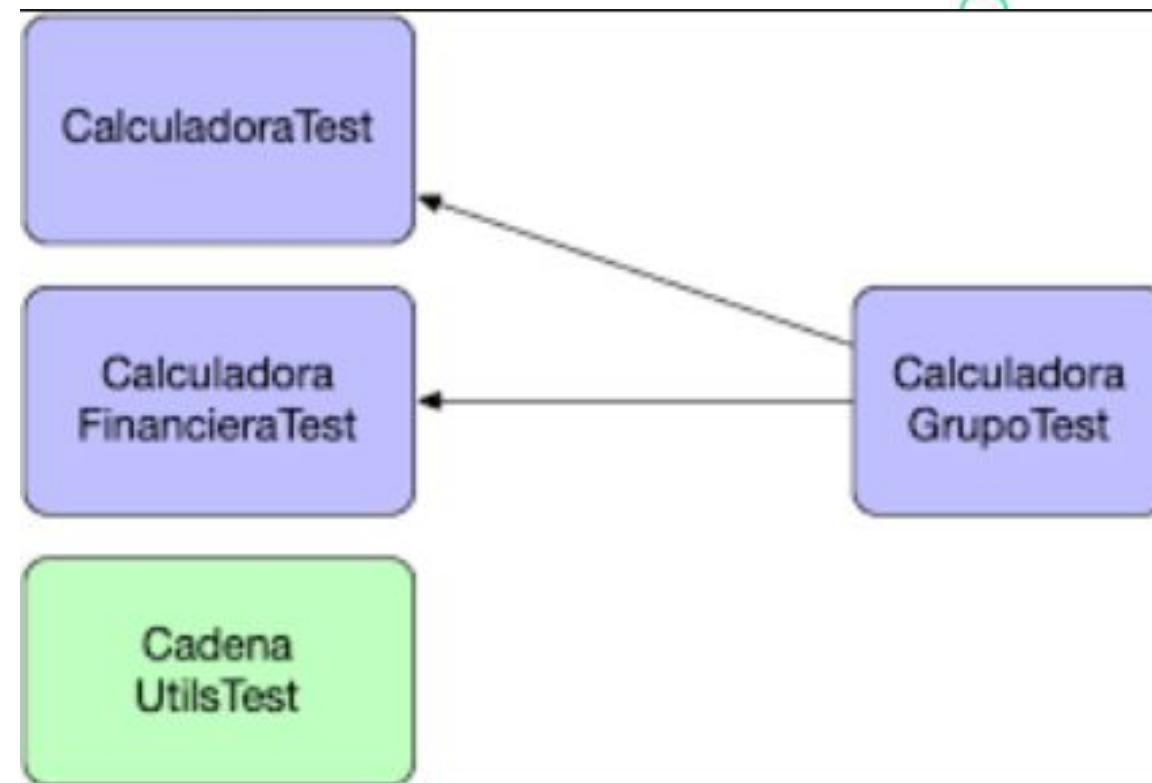


# Rompehielo 🧊



**Respondan levantando la mano:** 🙋

- ¿**Qué es mejor**: ejecutar cada caso de prueba individualmente o agrupar casos relacionados y ejecutarlos juntos? ¿Por qué?
- ¿Cómo podríamos agrupar casos de prueba en JUnit?



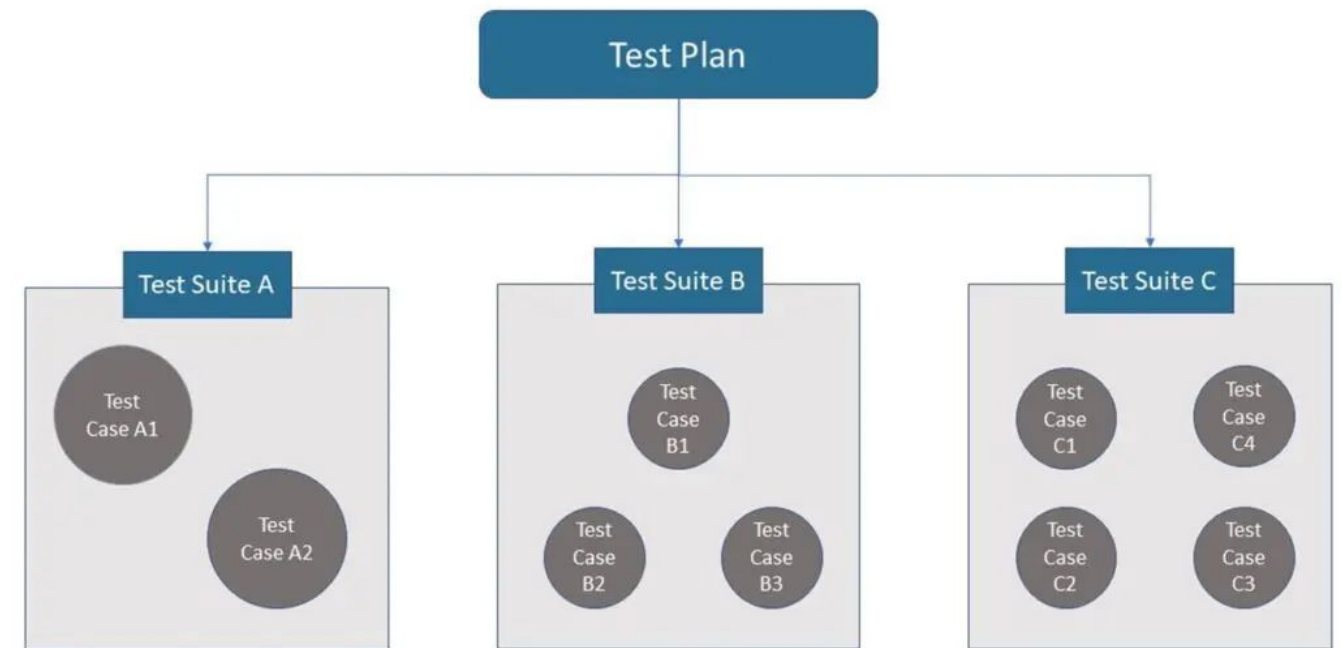
# › Suites de pruebas (Test Suite)





# Suites de pruebas

Las Test Suites en JUnit **son colecciones de casos de prueba que se agrupan para ejecutarse como una sola entidad**. Una suite de prueba permite organizar y ejecutar múltiples casos de prueba en secuencia o paralelamente, lo que facilita la ejecución y el análisis de pruebas en conjunto. En resumen, una suite de prueba **es una clase que actúa como contenedor para casos de prueba** JUnit.





# Suites de pruebas

## Características principales:



- **Agrupación de Casos de Prueba:** Las suites de prueba permiten agrupar casos de prueba relacionados o relevantes en una sola entidad.
- **Orden de Ejecución:** Puedes definir el orden de ejecución de los casos de prueba en una suite, lo que es útil cuando las pruebas dependen del estado resultante de pruebas anteriores.
- **Reusabilidad:** Las suites de prueba se pueden reutilizar en diferentes contextos y proyectos, lo que ahorra tiempo y esfuerzo en la configuración de las pruebas.
- **Separación de Pruebas:** Puedes crear múltiples suites de prueba para diferentes conjuntos de pruebas o escenarios de pruebas, lo que facilita la gestión de pruebas complejas.





# Suites de pruebas

## ¿Cómo se plantean las Test Suites?



La forma en que se plantean las Test Suites **puede variar según la herramienta de prueba utilizada**. Veamos una guía general sobre cómo plantear una Test Suite:




- **Identificar casos de prueba relacionados:** Agrupa los casos de prueba que están relacionados y se centran en probar una funcionalidad o un componente específico.
- **Crear una clase de suite** que sirva como punto de entrada para la ejecución de la suite. Esta clase debe estar anotada adecuadamente según la herramienta de prueba que estés utilizando (por ejemplo `@ExtendWith` en JUnit 5).





# Suites de pruebas

- **Especificar los casos de prueba incluidos:** En la clase de suite, especifica los casos de prueba individuales que desees incluir en la suite. Puedes hacer esto utilizando anotaciones o métodos específicos proporcionados por la herramienta de prueba.
- **Configurar y ejecutar la suite:** Configurar cualquier configuración adicional necesaria para la suite y ejecutarla. Esto puede implicar la configuración de parámetros de ejecución, establecimiento de datos iniciales, etc. 

Una Test Suite es una colección de casos de prueba relacionados que se ejecutan juntos para probar una funcionalidad o un componente del software. Las suites de prueba son útiles para organizar, reutilizar y ejecutar eficientemente casos de prueba, y proporcionan informes agregados para una visión general de los resultados de las pruebas.





# Suites de pruebas



## Creación de Suites de Prueba en JUnit:

1. Crear una nueva clase que actúe como la suite de prueba. Esta clase no contiene lógica de prueba, pero se utiliza para agrupar casos de prueba relacionados.
2. Usar la anotación **@RunWith(Suite.class)** en la clase de la suite para indicar que es una suite de prueba.
3. Definir una lista de clases de casos de prueba a ejecutar dentro de la suite usando la anotación **@SuiteClasses({})**. Enumera las clases de casos de prueba separadas por comas.
4. Ejecutar la suite de prueba, y JUnit ejecutará todos los casos de prueba enumerados en la suite.



# 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

## Test Suites:

*Vamos a crear una suite de prueba que contenga los casos de prueba codificados en las clases anteriores.*

- 1. Crear la suite y escribir las anotaciones necesarias para agrupar los casos de prueba que realizamos en las clases anteriores.*

 **Tiempo: 30 minutos**



# **Ejercicio N° 1**

# **En Suite**





# En Suite



**Consigna:** 🖋️ **Vamos a generar una Test Suite**

Ahora vamos a escribir la anotaciones y clases necesarias para implementar las Suites de Pruebas de los casos de prueba codificados en los ejercicios anteriores (de las clases de Polimorfismo)



**Tiempo** 🕒: 30 minutos



○

# ¿Alguna consulta?

+



# RESUMEN

¿Qué logramos en esta clase?

- ✓ **Comprender el concepto de Test Suite y sus características**



# #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 15 - 16*
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.

