Clase grabada

Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

English Always | alkemy

Clase grabada

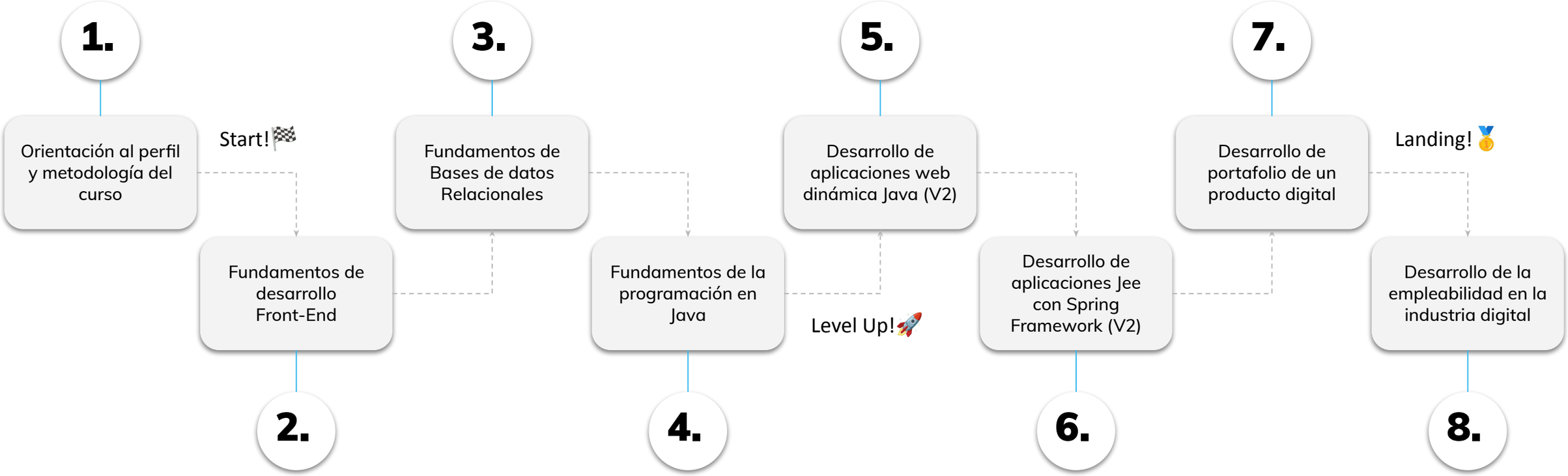
Pruebas Unitarias en Java

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

English Always | alkemy

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



English Always | alkemy

REPASO CLASE ANTERIOR

En la clase anterior trabajamos 📚:

- ✓ *Introducción a JUnit*

English Always | alkemy

LEARNING PATHWAY

4.7 Start!🏁

Pruebas Unitarias en Java → Creando un Caso de Prueba → Poniendo a Prueba

Casos de Prueba

English Always | alkemy

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?

- ***Comprender el concepto e implementación de los Casos de Prueba***

Rompehielo🥶

¿Qué probarías?: Respondan en el chat o levantando la mano 🙌

1. ¿Cómo podrían probar que los métodos sumar y restar están implementados correctamente?
2. ¿Qué posibilidades deberíamos cubrir como mínimo?

```java
public class Matematica {

    public int sumar(int a, int b) {
        return a+b;
    }

    public int restar(int a, int b) {
        return a-b;
    }
```

Casos de Prueba

Caso de Prueba

¿Qué son?:

Son unidades de trabajo que se utilizan para **verificar el comportamiento de una parte específica de un programa**. Representan situaciones de **entrada**, **ejecución** y **resultados esperados** de una función o método en el código.

Los casos de prueba son una parte fundamental del proceso de prueba unitaria y se utilizan para asegurarse de que las unidades de código, como clases o métodos, funcionen correctamente.

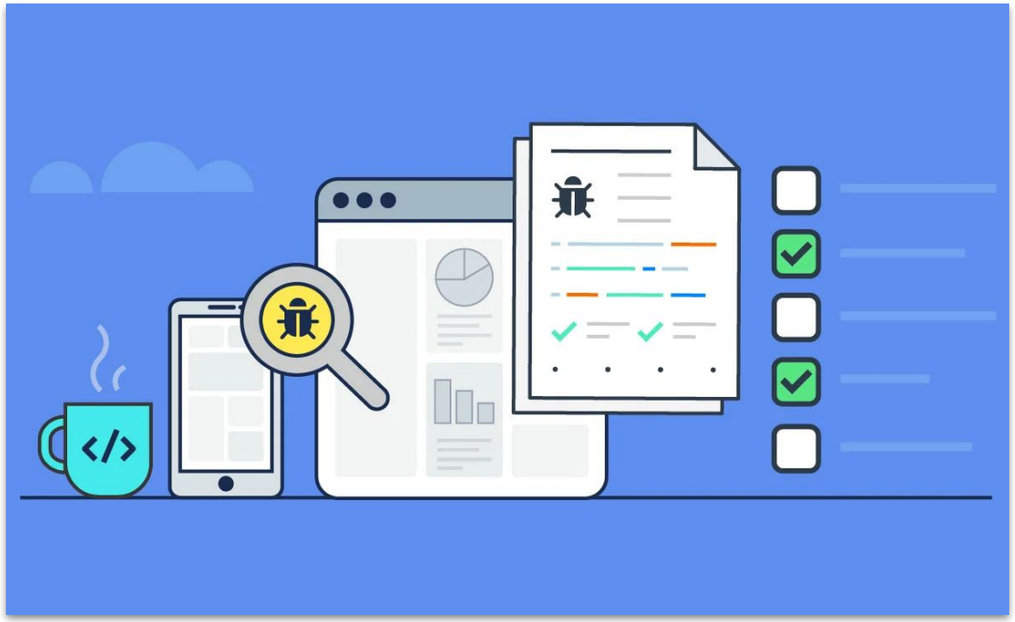
Casos de Prueba

Cobertura (coverage):

A la hora de crear los casos de prueba deberemos tener en cuenta que porcentaje de **coverage** o cobertura estableceremos:

- más del 70% es el valor mínimo aceptable
- entre el 85% y 95% es muy bueno
- más del 95% es excelente.

Una vez definido este valor, deberemos **considerar todas las clases a cubrir con test** y que casos posibles vamos a contemplar en los test.

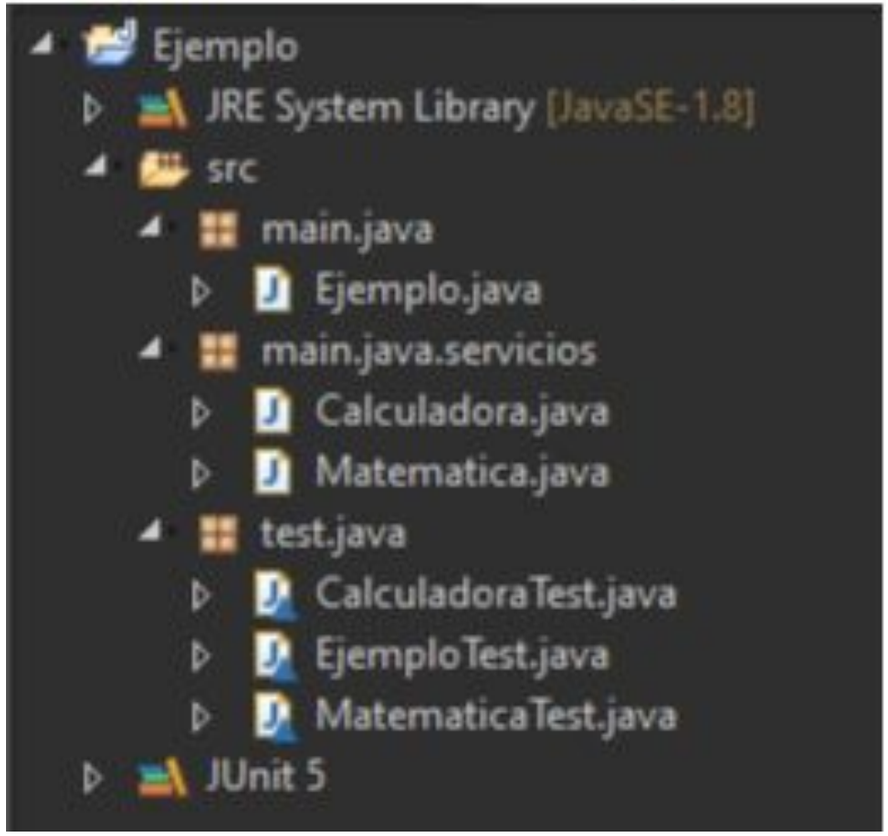


Casos de Prueba

Algunos criterios para decidir cómo organizar nuestros tests.

1. **Tantas clases Test como Clases a testear**: Se debe generar **una ClaseTest por cada Clase** que se desea testear.

   Un error común al iniciarse en la creación de los test unitarios es que dentro de una ClaseTest, busquemos testear tanto una Clase1 como una Clase2. Si deseamos testear ambas clases deberemos generar dos Clases Test distintas para cubrirlas.



Casos de Prueba

2. **Funciones a testear:** Si la Clase a Testear tiene cinco funciones, la ClaseTest deberá tener un mínimo de cinco @Test. Nunca se debe integrar en un @Test más de una función a testear.
Algo que sí es posible dentro de un @Test es intentar testear dos posibles salidas, un ejemplo de eso sería la siguiente función, parImpar, que puede devolver dos posibles valores (el @Test posee dos assertEquals).

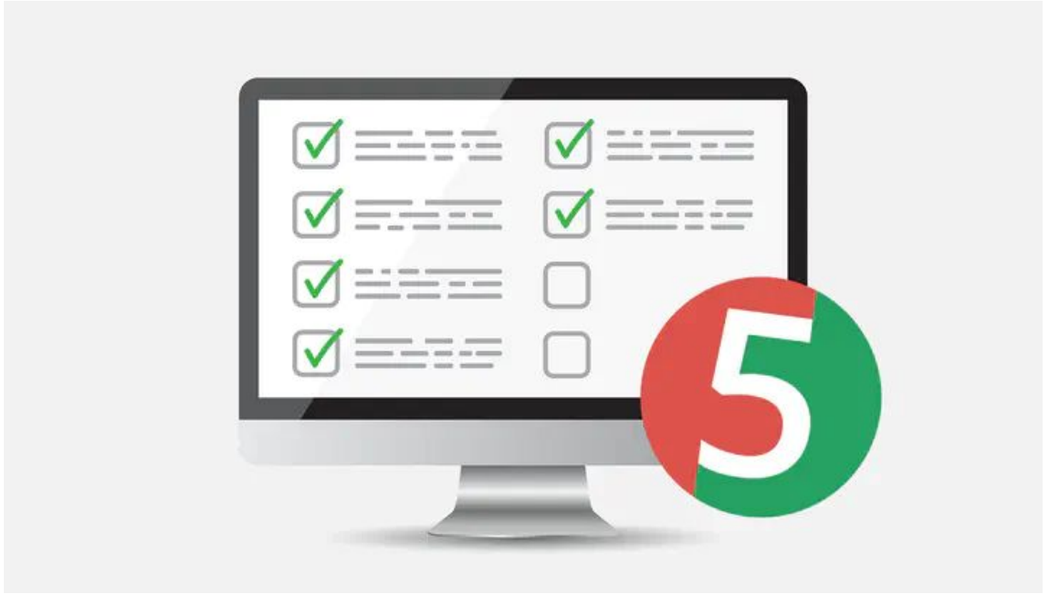
```
public String parImpar(int num) {
    if (num % 2 == 0) {
        return "Par";
    } else {
        return "Impar";
    }
}
```

```
@Test
void parImpar() {
    String expected = "Par";
    String actual = this.matematica.parImpar(2);
    assertEquals(expected, actual);

     expected = "Impar";
     actual = this.matematica.parImpar(1);
    assertEquals(expected, actual);
}
```

Casos de Prueba

3. **Casos de prueba:** Una vez definida la clase que queremos cubrir con los test, se deberá entender que funciones es necesario cubrir, qué parámetros de entrada necesitan las mismas, que parámetros de salida tendrán y qué posibles errores pueden ocurrir al usarlas.



Evaluación Integradora ✨

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

Trabajo Integrador del Módulo 💪🏽

Iremos completándolo progresivamente clase a clase.

English Always | alkemy

LIVE DEMO

Ejemplo en vivo

Probando: Vamos a realizar algunas pruebas para poner en práctica lo aprendido.

1. *Crear casos de prueba para depositar(), retirar() y getSaldo().*
2. *Explicar cómo se parametrizan las pruebas con diferentes casos y datos de entrada*

Tiempo: 20 minutos

Ejercicio N° 1

Poniendo a Prueba

Poniendo a Prueba

Practiquemos lo aprendido: En salas reducidas analizar lo siguiente 🙌

Imaginen que tenemos una clase de prueba CalculadoraTest y debemos pensar posibles casos de prueba para el método dividir().

Luego, en la sala principal compartir conclusiones según cada experiencia:

- ¿Qué tipo de pruebas crearían? ¿Por qué?

Tiempo🕐: *20 minutos*

¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ *Reconocer las implementaciones de los casos de prueba y sus características*

English Always | alkemy

#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 6 - 7*
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 👋🏻

English Always | alkemy

Momento:

Time-out!

⏰5 min.