¡Por supuesto! Aquí tienes un esquema centrado en \*\*JUnit\*\*:

---

### \*\*1. Introducción a JUnit\*\*

- \*\*Definición:\*\* Framework de código abierto para pruebas unitarias en Java.

- \*\*Propósito:\*\* Automatizar pruebas para validar el comportamiento de métodos y clases.

- \*\*Principios clave de las pruebas unitarias:\*\*

- \*\*Independencia:\*\* Cada prueba debe ejecutarse sin depender de otras.

- \*\*Aislamiento:\*\* Probar solo la unidad específica mediante stubs, mocks o fakes.

- \*\*Repetibilidad:\*\* Los resultados deben ser consistentes al repetir pruebas.

- \*\*Automatización:\*\* Las pruebas deben ejecutarse sin intervención manual.

- \*\*Claridad:\*\* Las pruebas deben ser comprensibles.

---

### \*\*2. Creación de una Clase de Prueba\*\*

- \*\*Pasos en Eclipse:\*\*

1. Clic derecho sobre la clase a probar -> Nueva -> Otras -> Java -> JUnit -> Caso de prueba JUnit.

2. Seleccionar la versión de JUnit (por defecto, JUnit5 Jupiter).

3. Elegir métodos de la clase para generar pruebas y finalizar.

- \*\*Estructura generada:\*\*

- Métodos con la anotación `@Test`.

- Métodos públicos, sin parámetros ni salida.

- Inicialmente incluye el método `fail` para señalar pruebas sin implementar.

---

### \*\*3. Métodos de Aserción\*\*

- \*\*Uso:\*\* Verificar resultados esperados en las pruebas.

- \*\*Principales métodos:\*\*

- `assertTrue(boolean expresión)`: Comprueba que la expresión es verdadera.

- `assertFalse(boolean expresión)`: Comprueba que la expresión es falsa.

- `assertEquals(valorEsperado, valorReal)`: Verifica igualdad entre valores.

- `assertNotEquals(valorEsperado, valorReal)`: Verifica que los valores no son iguales.

- `assertNull(Object objeto)`: Comprueba que el objeto es nulo.

- `assertNotNull(Object objeto)`: Comprueba que el objeto no es nulo.

- `assertSame(Object esperado, Object real)`: Verifica si son el mismo objeto.

- `assertNotSame(Object esperado, Object real)`: Verifica si no son el mismo objeto.

- `assertThrows(Class<T> tipoExcepción, Executable ejecutable)`: Verifica que se lanza una excepción esperada.

---

### \*\*4. Ejecución de Pruebas\*\*

- \*\*Pasos:\*\*

1. Clic derecho sobre la clase de pruebas -> Ejecutar Como -> Prueba JUnit.

2. Analizar resultados en la vista JUnit:

- \*\*Éxito:\*\* Check verde.

- \*\*Fallo:\*\* Aspa azul (no se cumple una aserción).

- \*\*Error:\*\* Aspa roja (excepción lanzada).

- \*\*Opciones adicionales en vista JUnit:\*\*

- Re-ejecutar prueba.

- Mostrar solo pruebas fallidas o errores.

- Ver historial de ejecución.

---

### \*\*5. Excepciones y AssertThrows\*\*

- \*\*Propósito:\*\* Verificar que el código lanza excepciones correctamente.

- \*\*Ejemplo:\*\*

```java

assertThrows(ArithmeticException.class, () -> calc.divide());

```

- También permite validar mensajes de excepción:

```java

Exception exc = assertThrows(NumberFormatException.class, () -> metodo());

assertEquals("Mensaje esperado", exc.getMessage());

```

---

### \*\*6. Anotaciones en JUnit\*\*

- \*\*`@BeforeEach:`\*\* Ejecuta un método antes de cada prueba.

- \*\*`@AfterEach:`\*\* Ejecuta un método después de cada prueba.

- \*\*`@BeforeAll:`\*\* Ejecuta un método estático antes de todas las pruebas.

- \*\*`@AfterAll:`\*\* Ejecuta un método estático al finalizar todas las pruebas.

- \*\*`@Disabled:`\*\* Desactiva un método de prueba temporalmente.

---

### \*\*7. Ejercicios Relacionados\*\*

1. \*\*Ejercicio básico:\*\* Completa pruebas para métodos como `testSuma` y `testResta`.

2. \*\*Prueba de concatenación:\*\* Comprueba concatenación de nombres y que el resultado no sea `null`.

3. \*\*Verificación de excepciones:\*\* Implementa métodos que lancen excepciones y úsalos con `assertThrows`.

4. \*\*Batería de pruebas:\*\* Crea casos para una clase con operaciones como suma de enteros, mayor valor y búsqueda de posición.

---

¿Te gustaría que profundice en algún punto o ayude con algún ejercicio en particular? 😊