**Verifications** in Mockito are a crucial aspect that allows you to check if certain methods were called on your mock objects, and if so, how many times and with what arguments. This helps ensure that your code interacts with its collaborators as expected.

**Why Verifications?**

While stubbing (defining what a mock method should return) is important for controlling behavior, verifications are essential for **asserting interactions**. They help answer questions like:

* Was this specific method called?
* How many times was it called?
* Was it called with the correct arguments?
* Was there no interaction with this method?

B**asic Verification Syntax**

The core of Mockito verification is the Mockito.verify() method. You pass the mock object to verify(), and then chain the method call you expect to have occurred.

Here's a concise text summary of Mockito verification modes for your notes:

**Mockito Verification Modes (for Mockito.verify())**

These modes specify how many times a method is expected to be called on a mock.

* **times(int wantedNumberOfInvocations)**:
  + Verifies method was called **exactly** wantedNumberOfInvocations times.
  + *Default mode if no other mode is specified (i.e., times(1)).*
  + **Example:** verify(mock, times(2)).myMethod();
* **never()**:
  + Verifies method was **never** called.
  + Equivalent to times(0).
  + **Example:** verify(mock, never()).myMethod();
* **atLeast(int minNumberOfInvocations)**:
  + Verifies method was called **at least** minNumberOfInvocations times.
  + **Example:** verify(mock, atLeast(1)).myMethod();
* **atLeastOnce()**:
  + Verifies method was called **at least once**.
  + Equivalent to atLeast(1).
  + **Example:** verify(mock, atLeastOnce()).myMethod();
* **atMost(int maxNumberOfInvocations)**:
  + Verifies method was called **at most** maxNumberOfInvocations times.
  + **Example:** verify(mock, atMost(3)).myMethod();
* **only()**:
  + Verifies that this method was the **only method called** on the mock, and it was called **exactly once**.
  + A very strict verification.
  + **Example:** verify(mock, only()).myMethod();

**Verification Matchers:**

**eq(…):**

Checks that the argument is **exactly equal**

🧪 **anyString() Example:**

Used when you **don’t care what the string value is**, only that **some string** was passed.

🔍 *This checks:*

* The first argument is **exactly** "bob@example.com"
* The second argument is **any String**, like "Welcome Bob!", "Hello!", "123", etc.
* 🔢 anyInt() Example
* Used when the method accepts an int argument, but you’re not concerned about its exact value.

verify(mockCalculator).add(anyInt(), eq(10));

🔍 *This checks:*

* The first number can be **any integer**
* The second number must be exactly 10

For example, if add(5, 10) or add(999, 10) was called—✅ it passes. But add(5, 20)—❌ it fails.

**Important Note on Argument Matchers:** When using argument matchers, **all arguments** for a method call in the verify() statement must be matchers. You cannot mix raw values with matchers directly without using eq()

// INCORRECT: Mixing raw value with matcher

// verify(mockDependency).calculate(10, anyInt()); // This will throw an exception

// CORRECT: Use eq() for the specific value

// verify(mockDependency).calculate(eq(10), anyInt());