# Mockito

Cosa la curiosità, nuovamente, ci ha spinto a fare:

Durante lo studio della teoria, nel capitolo relativo ai test double, siamo stati particolarmente “attratti” dal framework mockito.

In seguito abbiamo deciso di implementare un piccolo test per capire al meglio il suo funzionamento.

In allegato:

* Codice
* Descrizioe
* Considerazioni

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

La classe OrderProcessor è responsabile per l'elaborazione degli ordini. Utilizza un'istanza di PaymentService per gestire il pagamento di un ordine.

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

L'interfaccia PaymentService definisce un contratto per i servizi di pagamento. Le classi che implementano questa interfaccia devono fornire un metodo per gestire i pagamenti.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

La classe OrderProcessorTest contiene i test unitari per la classe OrderProcessor. Questi test utilizzano il framework Mockito per simulare il comportamento del servizio di pagamento.

I test verificano il comportamento corretto della classe OrderProcessor in situazioni in cui il pagamento è sia riuscito che fallito. Vengono utilizzati mock di PaymentService per simulare i comportamenti desiderati durante i test.

Considerazioni:

Abbiamo cercato di dimostrare l'utilizzo del framework Mockito per testare la classe OrderProcessor, che è responsabile dell'elaborazione degli ordini e dell'utilizzo di un servizio di pagamento esterno tramite l'interfaccia PaymentService.

**Separazione delle Responsabilità**: La classe OrderProcessor segue il principio di separazione delle responsabilità, concentrandosi sull'elaborazione degli ordini e delegando il pagamento a un servizio separato.

**Utilizzo di Interfacce**: L'interfaccia PaymentService è utilizzata per definire un contratto comune per i servizi di pagamento. Questo favorisce l'incapsulamento e l'estensibilità del codice.

**Mocking con Mockito**: Nei test unitari, Mockito viene utilizzato per creare mock di PaymentService al fine di simulare il comportamento del servizio di pagamento. Ciò permette di isolare il test della classe OrderProcessor e concentrarsi su scenari specifici di successo o fallimento del pagamento.

**Verifica delle Chiamate ai Metodi**: I test verificano che i metodi della classe OrderProcessor interagiscano correttamente con il mock di PaymentService. Questo viene fatto utilizzando il metodo verify di Mockito per assicurarsi che i metodi corretti siano chiamati con gli argomenti previsti.

Made by Salvatore Forte & Antonio Pio Caputo