

# Progetto del software

## Prova scritta – 14 luglio 2025 – 2h

### PARTE 1 – RISPOSTA SINGOLA - Ogni domanda ha una sola risposta VERA.

- Una risposta esatta fa acquisire il punteggio positivo riportato a fianco della domanda
  - Una risposta errata fa perdere il punteggio negativo riportato a fianco della domanda
  - Una risposta lasciata in bianco viene valutata 0
1. **(3, -.5)** I requisiti non funzionali di un software:
    - a) Sono quelli che danno valore al software stesso, e pertanto non sono concordati col cliente
    - b) Sono quelli che danno valore al software stesso, e pertanto sono direttamente concordati col cliente
    - c) Sono quelli relativi al processo di produzione del software, e generalmente non sono concordati col cliente
    - d) Nessuna delle precedenti
  2. **(3, -.5)** Un'architettura CLEAN:
    - a) Prevede che si identifichi un nucleo "core" di Entities che non dipendono da nulla se non da loro stesse
    - b) Prevede che si identifichi un nucleo "core" di Entities che dipendono solo dalle tecnologie legacy utilizzate (es. un DB Mongo)
    - c) Prevede che si identifichi un nucleo "core" di Entities che dipendono solo dai casi d'uso concordati con i clienti
    - d) Prevede che si identifichi un nucleo "core" di Entities che dipendono solo dal sistema operativo utilizzato
  3. **(3, -.5)** Il principio *Liskov Substitution* della programmazione SOLID:
    - a) Si applica solo alla programmazione funzionale
    - b) Si applica solo alle classi che implementano un'interfaccia (secondo *Interface Segregation*)
    - c) Prevede che una classe non possa estendere le caratteristiche di una classe padre
    - d) Prevede che una classe possa estendere una classe padre, a patto che possano venir sempre sostituite l'una con l'altra

## PARTE 2 – (POSSIBILI) RISPOSTE MULTIPLE -

Ogni domanda può avere da zero a quattro risposte CORRETTE.

- Ogni risposta esatta viene calcolata: +1
  - Ogni risposta errata viene calcolata: -0.5
  - Una risposta lasciata in bianco viene calcolata: 0
4. Il principio *divide-et-impera*:
- a) Può venire applicato solo a progetti software che seguano le metodologie waterfall
  - b) Può venire applicato anche a livello di architettura del software
  - c) Può venire applicato solo a progetti software che usano Git come tool di versioning
  - d) Può venire applicato anche a livello del singolo modulo o sottomodulo del software
5. Un paradigma di comunicazione asincrono:
- a) Non è applicabile nei client-server, che per definizione sono sempre sincroni
  - b) Prevede che il chiamante possa ignorare il risultato della chiamata, che pertanto può fallire in maniera silente, ed in tal caso si parla di modalità *fire-and-forget*
  - c) Prevede che il chiamante possa specificare una funzione di *callback*, o delegato, che si occuperà di gestire il risultato dell'operazione richiesta
  - d) Non è applicabile nel web
6. I principali problemi che sorgono nel progettare un software moderno sono principalmente causati dai seguenti fattori:
- a) La complessità crescente dei sistemi software moderni
  - b) La scarsa complessità dei sistemi software moderni, dovuta all'adozione di architetture a microservizi
  - c) Il privilegiare metodologie agili invece che waterfall
  - d) Il *time-to-market*, che causa pressione sul team di sviluppatori causando errori ed approssimazioni
7. Il concetto di commit di un sistema di versioning come Git:
- a) Non è stata visto nel corso
  - b) Non consente di tornare indietro ad una versione precedente del codice
  - c) Rappresenta la "fotografia" di una certa versione del codice
  - d) E' indispensabile per realizzare i principi SOLID
8. I diagrammi UML di scenario:
- a) Definiscono il comportamento del sistema in determinati scenari operativi, talvolta anche detti *Operational Design Domain*
  - b) Sono incompatibili con l'uso di librerie pre-esistenti
  - c) Possono richiedere di specificare pre-condizioni e post-condizioni per ogni scenario
  - d) Sono accompagnati da disegni e rappresentazioni tabellari, chiare, univoche e complete, che specificano il comportamento del sistema

### PARTE 3 – DOMANDE APERTE

- Una risposta esatta fa acquisire il punteggio positivo riportato a fianco della domanda
  - Una risposta errata può eventualmente causare una penalità che dipende dalla gravità dell'errore
  - Una risposta lasciata in bianco viene calcolata: 0
  - L'eventuale sfioramento del limite di righe o parole (laddove imposto), porterà a una decurtazione di un punto per ogni riga. Eventuali schematici e listati di codice non verranno presi in considerazione nel calcolo delle righe
  - **SI RICORDA CHE L'UNICO FOGLIO DA CONSEGNARE E' IN CALCE AL COMPITO. QUESTO FOGLIO, PUO' SERVIRE ESCLUSIVAMENTE COME "BRUTTA COPIA". EVENTUALI RISPOSTE SCRITTE IN QUESTO FOGLIO NON VERRANNO PRESE IN CONSIDERAZIONE**
9. **(5 pt)** Si descrivano cosa sono gli integration test, e come il *mocking* possa essere d'aiuto nel realizzarli

Nome e Cognome \_\_\_\_\_ Matricola: \_\_\_\_\_

10. **(6 pt)** Si descriva la differenza fra un toolkit ed un framework.

Nome e Cognome \_\_\_\_\_ Matricola: \_\_\_\_\_

## Progetto del software

### Prova scritta – 14 luglio 2025 – 2h

Indicare le risposte corrette apponendo una croce nella casella corrispondente. Per superare la prova bisogna aver raggiunto almeno 9 punti nelle domande a risposta singola/multipla, ed almeno 15 complessivamente. Questa è l'unica pagina che dovete consegnare.

	Risposte				Punti/ Penalità	
	A	B	C	D		
1					3	-0.5
2					3	-0.5
3					3	-0.5
4						
5						
6						
7						
8						

Risposta alla domanda 9 (5 pt):

Nome e Cognome \_\_\_\_\_ Matricola: \_\_\_\_\_

**Risposta alla domanda 10 (6 pt):**