

Testing

Versione	Data	Descrizione
1.0	19 Gennaio 2026	Casi di test e copertura codice

Casi di test di unità

I test di unità sono stati effettuati con il tool JUnit 5 e grazie all'aiuto della struttura del progetto Maven: abbiamo infatti una cartella apposta `src/test/java` dove poter creare i package e le classiTest corrispondenti alle rispettive classi che andremo a testare. Il testing avviene grazie alla dichiarazione di variabili globali e di un metodo `@BeforeEach` `setUpAll()` che viene eseguito prima dell'esecuzione di ogni caso di test.

Package Card

In questo package è stata testato il metodo `checkRank()` fornito dalla classe `Hand` per calcolare il suo Rank, ovvero il punteggio della mano. Sono stati effettuati diversi casi di test, uno per ogni possibile punteggio del gioco: carta alta, coppia, doppia coppia, tris, scala, colore, full, poker, scala colore.

Package Game

In questo package è stata testata la classe `Game`:

- metodo `checkFirstTurn()` che calcola l'id del primo turno sull'insieme dei soli giocatori attivi: è stato testato in modo che gli id fossero non incrementali e non consecutivi, e uno dei giocatori ha foldato.
- metodo `nextTurn()` che calcola l'id del turno successivo sull'insieme dei soli giocatori attivi. E' stato testato in modo che gli id fossero non incrementali e non consecutivi e che:
 - si passasse dall'ultimo del giro al secondo, poichè il primo ha foldato
 - si passasse da un giocatore che ha foldato a quello successivo
- metodo `splitPot()` che distribuisce il piatto tra i soli giocatori attivi

Package Server


























L'architettura client/server ci obbliga a scambiare messaggi, quindi è stata dichiarata una classe `FakeSocketSafe` per testare il codice senza effettivamente farlo.

In questo package è stato testato una possibile combinazioni di scelte dei giocatori accentuando alcuni elementi:

- azioni fuori turno
- azioni non valide
- all-in

Copertura

Grazie all'analisi di copertura dei test JUnit abbiamo ottenuto il report che descrive il grado di copertura delle istruzioni nel nostro codice: nel nostro progetto è stato coperto circa il 70% delle istruzioni, un valore ben più che accettabile.

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
▼ poker	 69,3 %	3.944	1.744	5.688
▼ src/main/java	 56,1 %	2.225	1.741	3.966
▼ THRProject.server	 48,0 %	925	1.001	1.926
> Server.java	 51,0 %	880	844	1.724
> ClientHandler.java	 23,2 %	45	149	194
> ServerMain.java	 0,0 %	0	8	8
> THRProject.client	 0,0 %	0	450	450
▼ THRProject.game	 82,5 %	542	115	657
> Game.java	 79,2 %	438	115	553
> GamePhase.java	 100,0 %	74	0	74
> Pot.java	 100,0 %	30	0	30
▼ THRProject.message	 60,6 %	123	80	203
> ActionType.java	 0,0 %	0	74	74
> Message.java	 60,0 %	9	6	15
> ControlType.java	 100,0 %	114	0	114
▼ THRProject.player	 60,0 %	99	66	165
> Player.java	 31,2 %	25	55	80
> PlayerStatus.java	 87,1 %	74	11	85
▼ THRProject.card	 94,9 %	536	29	565
> Card.java	 37,5 %	15	25	40
> Deck.java	 94,4 %	67	4	71
> Hand.java	 100,0 %	393	0	393
> Rank.java	 100,0 %	17	0	17
> Suit.java	 100,0 %	44	0	44
> src/test/java	 99,8 %	1.719	3	1.722