DEFINIZIONE DEI TEST CASES

Test Case per Klotski								
Progetto	Klotski							
Modulo	Movimento blocco							
Scenario ID	Descrizione Scenario	Test Case ID	Pre Condizioni	Steps	Risultati attesi	Risultato ottenuto	Status	Commenti
1	Movimento valido del blocco di dimensioni maggiori nella posizione di vittoria	KLOTSKI TC1_a	La configurazione corrente del gioco permette di spostare il blocco di dimensioni maggiori nella posizione vincente	1.Selezionare il blocco di dimensioni maggiori 2.Trascinare il blocco nella posizione vincente	Viene aggiornata la configurazione con la nuova mossa eseguita e viene visualizzato il messaggio che segnala il completamento del puzzle	I risultati ottenuti sono in linea con quelli attesi	Superato	Questa test case verifica i risultati di uno spostamento valido del blocco di dimensioni maggiori nella posizione di risoluzione del puzzle (per valido si intende che non ci sono blocchi che interpongono nella traiettoria di spostamento e le caselle su cui viene spostato il blocco sono libere)
2	Movimento valido di un blocco	KLOTSKI TC1_b	Non ci sono blocchi che si interpongono nella traiettoria di spostamento e le caselle su cui viene spostato il blocco sono libere	1.Selezionare il blocco da spostare 2.Trascinare il blocco nella posizione valida	Viene aggiornata la configurazione con la nuova mossa eseguita e viene incrementato il contatore delle mosse effettuate	I risultati ottenuti sono in linea con quelli attesi	Superato	Questa test case verifica che la configurazione del puzzle venga aggiornata al seguito di uno spostamento valido di un blocco, inoltre si verifica che il contatore delle mosse effettuate venga incrementato al termine dell' operazione di spostamento
3	Movimento non valido di un blocco	KLOTSKI TC1_c	Ci sono dei blocchi che si frappongono allo spostamento del blocco selezionato	1.Selezionare il blocco da spostare 2.Trascinare il blocco nella posizione non valida	La configurazione non viene aggiornata,non viene incrementato il contatore ed in generale non si ha nessuna modifica nel puzzle	I risultati ottenuti sono in linea con quelli attesi	Superato	Questa test case verifica che uno spostamento non valido non determini nessuna variazione

Test Case per Klotski								
Progetto	Klotski							
Modulo	Scelta della configurazione							
Scenario ID	Descrizione Scenario	Test Case ID	Pre Condizioni	Steps	Risultati attesi	Risultati ottenuti	Status	Commenti
1	Scelta della Configurazione	KLOTSKI TC2	Nessuna	1.Selezionare "Options" 2.Selezionare "Change Configurations" 3.Selezione una delle 4 configurazioni (1-4)	Viene aggiornata la configurazione con una delle 4 selezionate, il contatore delle mosse viene azzerato	I risultati ottenuti sono in linea con quelli attesi	Superato	Questa test case verifica che sia possibile cambiare la configurazione corrente del puzzle con una delle 4 di default, inoltre il contatore delle mosse deve essere azzerato una volta cambiata la configurazione

Test Case per Klotski								
Progetto	Klotski							
Modulo	Reset Puzzle							
Scenario ID	Descrizione Scenario	Test Case ID	Pre Condizioni	Steps	Risultati attesi	Risultati ottenuti	Status	Commenti
1	Reset del puzzle	KLOTSKI TC3	Nessuna	1.Selezionare l'icona ↻	Viene ripristinata la configurazione iniziale ed azzerato il contatore delle mosse	I risultati ottenuti sono in linea con quelli attesi	Superato	Questa test case verifica che sia possibile fare un reset della partita corrente, ripristinando la configurazione iniziale

Test Case per Klotski								
Progetto	Klotski							
Modulo	Undo							
Scenario ID	Descrizione Scenario	Test Case ID	Pre Condizioni	Steps	Risultati attesi	Risultati ottenuti	Status	Commenti
1	Undo eseguito su configurazione non iniziale	KLOTSKI TC4_a	Sono state eseguite precedentemente delle mosse, quindi il contatore non è nullo	1.Selezionare l'icona ←	Viene ripristinata la configurazione precedente all'ultima mossa e decrementato il contatore delle mosse	I risultati ottenuti sono in linea con quelli attesi	Superato	Questa test case verifica che sia possibile fare un undo, annullando l'ultima mossa eseguita
2	Undo eseguito su configurazione iniziale	KLOTSKI TC4_b	Non sono state eseguite precedentemente delle mosse, quindi il contatore è nullo	1.Selezionare l'icona ←	Non essendo stata eseguita nessuna mossa, l'undo non causa nessuna modifica al puzzle	I risultati ottenuti sono in linea con quelli attesi	Superato	Questa test case verifica che un undo eseguito su una configurazione iniziale non causi alcuna modifica

Test Case per Klotski								
Progetto	Klotski							
Modulo	Next Best Move							
Scenario ID	Descrizione Scenario	Test Case ID	Pre Condizioni	Steps	Risultati attesi	Risultati ottenuti	Status	Commenti
1	Esecuzione di molteplici next best moves a partire da una delle quattro configurazioni iniziali	KLOTSKI TC5-a	La configurazione dove viene eseguita la prima next best move deve essere una delle 4 di default, è possibile selezionare delle ulteriori next best move fintanoché l'utente non abbia eseguito nessuna operazione di spostamento blocco	1.Selezionare "Options" 2.Selezionare "Next Best Move"	Il solver aggiorna la configurazione del puzzle compiendo la miglior mossa possibile, viene naturalmente incrementato il contatore delle mosse e l'esecuzione del programma prosegue senza complicazioni	I risultati ottenuti sono in linea con quelli attesi	Superato	1.Perché il solver aggiorni la configurazione corrente con la miglior mossa possibile, è necessario che non non sia stata eseguita dal player umano nessuna operazione di spostamento blocco 2. E' possibile eseguire in serie dei next best moves che determinano la risoluzione del puzzle
2	Esecuzione combinata di next best moves e spostamenti manuali	KLOTSKI TC5_b	Nessuna	1a.Selezionare "Options" 2a.Selezionare "Next Best Move" oppure 1b.Selezionare il blocco da spostare 2b.Trascinare il blocco nella posizione valida	Per qualsiasi configurazione su cui il solver viene invocato, il solver aggiorna la configurazione del puzzle compiendo la miglior mossa possibile, viene naturalmente incrementato il contatore delle mosse e l'esecuzione del programma prosegue senza complicazioni	I risultati ottenuti non sono sempre in linea con quelli attesi. Il player umano attraverso molteplici spostamenti manuali dei blocchi, potrebbe determinare delle configurazioni di puzzle per cui il funzionamento corretto dell'algoritmo di solving non è garantito, determinando degli errori nell'esecuzione del programma	Non Superato	L'algoritmo di solving è progettato per ricondurre la configurazione corrente della partita ad una configurazione per cui è possibile risolvere il puzzle con un numero limitato di next best moves.Se a causa di molteplici spostamenti manuali di blocchi, la configurazione corrente è eccessivamente diversa dalla risoluzione proposta dal Solver, si potrebbe avere un overflow causando un funzionamento non corretto del programma

Test Case per Klotski								
Progetto	Klotski							
Modulo	Salvataggio							
Scenario ID	Descrizione Scenario	Test Case ID	Pre Condizioni	Steps	Risultati attesi	Risultati ottenuti	Status	Commenti
1	Salvataggio	KLOTSKI TC6	Nessuna	1.Selezionare "Options" 2.Selezionare "Save"	La partita corrente viene salvata ed è possibile caricarla in un momento successivo	I risultati ottenuti sono in linea con quelli attesi	Superato	Questa test case verifica che sia possibile creare un salvataggio di una partita corrente

Test Case per Klotski								
Progetto	Klotski							
Modulo	Ripristino							
Scenario ID	Descrizione Scenario	Test Case ID	Pre Condizioni	Steps	Risultati attesi	Risultati ottenuti	Status	Commenti
1	Ripristino	KLOTSKI TC7	Deve essere disponibile un precedente salvataggio di una partita	1.Selezionare "Options" 2.Selezionare "Load"	Viene ripristinata la partita precedentemente salvata	I risultati ottenuti sono in linea con quelli attesi	Superato	Questa test case verifica che sia possibile ripristinare una partita salvata

Test Case per Klotski								
Progetto	Klotski							
Modulo	Quit							
Scenario ID	Descrizione Scenario	Test Case ID	Pre Condizioni	Steps	Risultati attesi	Risultati ottenuti	Status	Commenti
1	Quit	KLOTSKI TC8	Nessuna	1.Selezionare "Options" 2.Selezionare "Quit"	Viene interrotta l'esecuzione del programma	I risultati ottenuti sono in linea con quelli attesi	Superato	Una volta selezionato il quit, il programma termina

SYSTEM TEST REPORT

Sommario

Come osservabile nella definizione dei test case questi sono suddivisi in 8 categorie principali:

1. **I test case che verificano il meccanismo di movimento manuale dei blocchi del puzzle (KLOTSKI TC1)**
   Questi consistono nello spostamento dei blocchi da parte dell'utente, nello specifico esistono 3 possibilità: l'utente ha eseguito lo spostamento che determina la risoluzione del puzzle, l'utente ha eseguito un generico spostamento valido, l'utente ha provato ad eseguire uno spostamento non valido
2. **Il test case che verifica il corretto funzionamento della scelta della configurazione (KLOTSKI TC2)**
3. **Il test case che determina il corretto funzionamento dell'opzione di reset del puzzle (KLOTSKI TC3)**
4. **I test case che verificano l'undo (KLOTSKI TC4)**
   Quindi nel caso sia stato eseguito almeno uno spostamento, si verifichi che sia possibile annullare l'ultima mossa e ripristinare la configurazione precedente
5. **I test case ideati per la next best move (KLOTSKI TC5)**
   Servono per stabilire se il solver riesce a trovare per le varie configurazioni la sequenza risolutiva con il minor numero di spostamenti
6. **Il test case che verifica il salvataggio di una partita in corso (KLOTSKI TC6)**
   Serve per testare che si possa salvare correttamente una partita in corso
7. **Il test case per il ripristino di un salvataggio (KLOTSKI TC7)**
   Serve per verificare che si possa ripristinare correttamente il salvataggio di una partita
8. **Il test case per il Quit (KLOTSKI TC8)**
   Si stabilisce se l'opzione di Quit permette effettivamente di terminare l'esecuzione del programma

Questi test case sono stati verificati tramite 3 classi di test: **BoardTest, SaverLoaderTest e SolverTest** , sviluppate con l'ausilio di JUnit.

BoardTest

- ### testReset()

  SOMMARIO: Questo metodo permette di verificare il test Case del Reset

  TEST CASE DESIGN: Questo metodo crea una Board con una determinata configurazione, fa un'operazione di movimento blocco sulla Board e successivamente un'operazione di Reset, verifica quindi con un assert che la configurazione della Board sia uguale a quella iniziale

  PRE-CONDIZIONI: L'operazione di movimento blocco deve essere valida, altrimenti il test non è sufficiente per stabilire la correttezza del Reset

  POST-CONDIZIONI: La board è configurata alla configurazione iniziale

  RISULTATI ATTESI: L'asserzione conferma l'uguaglianza tra la board nella configurazione iniziale e la board dopo il reset

  RISULTATI OTTENUTI: I risultati ottenuti sono in linea con quelli attesi

- **testSetConfiguration1(), testSetConfiguration2(), testSetConfiguration3(), testSetConfiguration4()**

  SOMMARIO: Ciascuno di questi metodi di test permette di verificare che sia possibile cambiare la configurazione corrente nella rispettiva configurazione di default
  TEST CASE DESIGN: La prima istruzione serve per l'impostazione della configurazione della Board, successivamente si verifica con delle asserzioni che nell'oggetto Board l'attributo identificativo della configurazione sia corretto e che il posizionamento dei blocchi sia coerente con quello atteso.
  PRE-CONDIZIONI: Per il confronto tra oggetti di tipo Board, viene invocato il metodo toString() su ciascun oggetto, il quale restituisce la configurazione della board sottoforma di stringa, è necessario che il corretto funzionamento di questo metodo sia stato precedentemente verificato nella Unit Testing.
  POST-CONDIZIONI: Ogni metodo configura la Board alla rispettiva configurazione (testSetConfiguration1()-> Configurazione 1, ecc…)
  RISULTATI ATTESI: L'asserzione conferma il corretto posizionamento dei blocchi.
  RISULTATI OTTENUTI: I risultati ottenuti sono in linea con quelli attesi

- **testGetBestSolutionSequence()**

  SOMMARIO: Questo metodo permette di verificare che ad ogni best move, la Board venga configurata correttamente.
  TEST CASE DESIGN: Nella prima istruzione viene invocato sull'oggetto board il metodo GetBestSolution(), in modo da ottenere la sequenza delle mosse migliori, successivamente viene invocato il metodo evaluate() per compiere spostamenti di blocchi su Board e per ogni spostamento di blocchi, si verifica attraverso le asserzioni che la Board venga aggiornata correttamente.
  PRE-CONDIZIONI: Nello Unit Testing deve essere verificato che il metodo GetBestSolution() restituisca la sequenza più breve di mosse con cui è possibile risolvere il puzzle
  POST-CONDIZIONI: Alla fine dell'esecuzione del test, la Board rappresenta una configurazione vincente
  RISULTATI ATTESI: La Board viene aggiornata fino alla configurazione vincente, senza che siano presenti delle anomalie
  RISULTATI OTTENUTI: I risultati ottenuti sono in linea con quelli attesi

- **testEvaluate()**

  SOMMARIO: L'esecuzione di questo metodo di test è in relazione con il test case dello spostamento di un blocco, in quanto verifica che il metodo evaluate selezioni il primo spostamento valido nell'insieme delle mosse valide e aggiorni correttamente la configurazione
  TEST CASE DESIGN: La prima istruzione richiede l'insieme dei possibili spostamenti di un blocco, con evaluate() viene eseguito il primo spostamento valido e con un asserzione si controlla che la board venga aggiornata correttamente dopo lo spostamento.
  PRE-CONDIZIONI: Affinché il test convalidi la correttezza di evaluate(), è necessario che il metodo che restituisce le possibili mosse valide, quindi GetPossibleMoves() venga a sua volta testato.
  POST-CONDIZIONI: La board è correttamente configurata all'ultima operazione di spostamento
  RISULTATI ATTESI: L'asserzione dimostra che il metodo evaluate() aggiorna correttamente la configurazione dopo lo spostamento del blocco.
  RISULTATI OTTENUTI: I risultati ottenuti sono in linea con quelli attesi

- **testCheckwin()**

  SOMMARIO: Questo metodo di test verifica che la configurazione vincente determini effettivamente la condizione di vittoria

  TEST CASE DESIGN: Viene inizialmente creato un oggetto Board configurato alla configurazione vincente, a questo punto con un'asserzione si verifica che il sistema segnali la condizione di vittoria

  PRE-CONDIZIONI: L'oggetto Board su cui si esegue il test deve effettivamente rappresentare una configurazione vincente

  POST-CONDIZIONI: Il sistema ha appurato la risoluzione del puzzle

  RISULTATI ATTESI: Viene verificato che una configurazione vincente identifichi la risoluzione del puzzle

  RISULTATI OTTENUTI: I risultati ottenuti sono in linea con quelli attesi

- **testGetPossibleMoves()**

  SOMMARIO: Anche questo metodo di test è in relazione con il test case dello spostamento di un blocco ed insieme al metodo testEvaluate() permette di verificare ulteriormente che il sistema reagisca in maniera corretta allo spostamento di un blocco

  TEST CASE DESIGN: Attraverso delle asserzioni di uguaglianza tra evaluate() e le varie mosse restituite dall'invocazione di GetPossibleMoves(), si verifica che il metodo GetPossibleMoves() restituisca unicamente spostamenti di blocchi validi

  PRE-CONDIZIONI: Deve essere testato il corretto funzionamento del metodo evaluate()

  POST-CONDIZIONI: Viene esaminata la validità per ogni singola mossa restituita da GetPossibleMoves()

  RISULTATI ATTESI: Viene verificato che GetPossibleMoves() restituisca unicamente mosse valide

  RISULTATI OTTENUTI: I risultati ottenuti sono in linea con quelli attesi

SaverLoaderTest

- **testSave()**

  SOMMARIO: Questo metodo di test verifica che sia possibile salvare correttamente una partita

  TEST CASE DESIGN: Sull'oggetto SaverLoader, la cui funzione è quella di contenere le informazioni relative alla configurazione salvata, viene eseguita un'operazione di salvataggio, si verifica che questa venga completata senza anomalie

  PRE-CONDIZIONI: É sufficiente aver inizializzato un oggetto Board prima di eseguire l'operazione di salvataggio

  POST-CONDIZIONI: Il metodo save restituisce la stringa "ok" per segnalare che il salvataggio è avvenuto correttamente, inoltre, l'oggetto SaverLoader contiene ancora le informazioni relative alla configurazione salvata

  RISULTATI ATTESI: L'asserzione usata per verificare il corretto salvataggio, convalida che il metodo save abbia restituito "ok"

  RISULTATI OTTENUTI: I risultati ottenuti sono in linea con quelli attesi

- **testLoad()**

  SOMMARIO: Questo metodo di test verifica che il ripristino di un salvataggio funzioni propriamente

  TEST CASE DESIGN: Sull'oggetto SaverLoader (su cui era precedentemente stato salvato un oggetto Board) viene eseguito il metodo load() che permette di restituire l'ultimo oggetto Board salvato, con un'asserzione si verifica che la Board inizialmente salvata nel SaverLoader coincida con la Board restituita da load()

  PRE-CONDIZIONI: Prima di eseguire testLoad() è necessario aver eseguito il metodo testSave() ed aver appurato il corretto funzionamento del metodo save(), dato che all'istante d'esecuzione di testLoad() il SaverLoader deve contenere l'istanza della Board

  POST-CONDIZIONI: L'oggetto SaverLoader non viene modificato

  RISULTATI ATTESI: L'asserzione d'uguaglianza conferma che la Board salvata viene ripristinata correttamente

  RISULTATI OTTENUTI: I risultati ottenuti sono in linea con quelli attesi

SolverTest

- **testGetNextMove()**

  SOMMARIO: Questo metodo di test estende la Test Coverage per quando riguarda l'algoritmo di Solving e dimostra che una sequenza vincente è individuabile per ciascuna delle quattro configurazioni di default

  TEST CASE DESIGN: Viene eseguito GetNextMove() sul solver, fintantoché questo restituisce una Board per cui l'invocazione di checkwin() restituisce TRUE, questo test viene eseguito per ogni Board di default

  PRE-CONDIZIONI: Per ritenere il test efficace, il testCheckwin() deve essere stato precedentemente eseguito e deve aver dato esiti positivi

  POST-CONDIZIONI: Al termine del test, l'oggetto Board si trova alla configurazione vincente ed il sistema ha segnalato la risoluzione del puzzle

  RISULTATI ATTESI: Per ciascuna Board di default, il solver attraverso un numero limitato di invocazioni di GetNextMove(), deve aggiornare la Board nella configurazione vincente

  RISULTATI OTTENUTI: I risultati ottenuti sono in linea con quelli attesi