GROUP 18
Griño, Mary Eunice
Maristela, Kyle Gabriel

Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
initializeGame	1	Game starts correctly when choosing option 1	1	“Shuffling pieces…” …	“Shuffling pieces…” …	P
	2	Game exits when choosing option 2	2	“Exiting game…”	“Exiting game…”	P
	3	Will continuously prompt the user until valid input	3	Invalid choice! Please enter 1 or 2.	Invalid choice! Please enter 1 or 2.	P
shuffleAndAssignPieces	1	Correct player goes first	Player 1 picks hidden number 1 (Dog), Player 2 gets hidden number 2 (Cat)	Player 1 goes first	Player 1 goes first	P
	2	Pieces are shuffled and assigned correctly	Player 1 picks hidden number 1 (Dog)	System assigns remaining piece to other player	Player 2 is assigned hidden number 2	P
	3	Will continuously prompt the user until valid input	Player 1, pick a number between 1-2: 3	Invalid choice. Please pick a valid number. Player 1, pick a number between 1-2:	Invalid choice. Please pick a valid number. Player 1, pick a number between 1-2:	P
playGame	1	When a player has no pieces left	currentPlayer.getActivePieces().isEmpty() = true	"Player 1 has no pieces left. Turn forfeited." message	"Player 1 has no pieces left. Turn forfeited." message	P

	2	When both players have no pieces	player1.getActivePieces().isEmpty() && player2.getActivePieces().isEmpty() = true	"Both players have no pieces left. Game ends in a draw!"	"Both players have no pieces left. Game ends in a draw!"	P
	3	When a player captures opponents home base	Player 1 (Blue) captures Player 2 (Green)'s home base	“Player 1 (Blue) wins the game!”	“Player 1 (Blue) wins the game!”	P
selectPiece	1	Valid and existing piece selected	dog	Piece moves	Piece moves	P
	2	Invalid piece selected	doge	“Piece not found or has been captured! It cannot be moved.”	“Piece not found or has been captured! It cannot be moved.”	P
	3	Valid but captured piece selected	dog	“Piece not found or has been captured! It cannot be moved.”	“Piece not found or has been captured! It cannot be moved.”	P
getDirectionInput	1	Valid direction input	w	Piece moves up	Piece moves up	P
	2	Invalid direction input	e	"Invalid direction. Use W, A, S, or D."	"Invalid direction. Use W, A, S, or D."	P
checkWinCondition	1	Check win condition when a piece has reached home base	Player 1's Dog reaches Green's home base	“Player 1 (Blue) wins the game! Game over!”	“Player 1 (Blue) wins the game! Game over!”	P

switchTurn	1	Turn switches correctly between players	Player 1 finishes move	Player 2's turn starts	Player 2's turn starts	P
addPiece	1	Dog piece is added to the player's list	dog	Piece dog is added in pieces list	Piece dog is added in pieces list	P
getPiece	1	Determines if dog piece can be used while it is captured	dog	"Piece not found or has been captured! It cannot be moved." null	"Piece not found or has been captured! It cannot be moved." null	P
	2	Determines if cat piece can be used	cat	cat	cat	P
getAllPieces	1	Retrieves all of player’s pieces	pieces	pieces	pieces	P
getActivePieces	1	Retrieves all of player’s pieces	pieces	pieces	pieces	P
movePiece	1	Wrong direction input	C	"Invalid direction. Use W, A, S, or D."	"Invalid direction. Use W, A, S, or D."	P
	2	Determines if piece can be moved over another piece from the same owner/current player	Blue Dog, Blue Cat	“Invalid move.”	“Invalid move.”	P
	3	Determines if piece can be moved over opponent piece with greater strength	Blue cat, Green Dog	“Capture not allowed”	“Capture not allowed”	P
hasLostAllPieces	1	Detects if all pieces has been lost	All pieces are captured	true	true	P
	2	Player still has active pieces	Pieces are not all captured	false	false	P
move	1	Piece moves correctly	Dog moves from (3,3) to (4,3)	Dog is now at (4,3)	Dog is now at (4,3)	P

	2	Piece's position updates after multiple moves	Dog moves from (2,2) → (2,3) → (3,3)	Dog is now at (3,3)	Dog is now at (3,3)	P
	3	Piece did not move to new position	Dog tries to move from (2,2) → (2,3)	Dog is still at (2,2)	Dog is still at (2,2)	P
canMove	1	Test if a valid move is allowed	Dog moves from (3,3) to (4,3)	true	true	P
	2	Test if an invalid move is blocked	Blue Dog moves from (3,3) to (4,3), Blue Cat is occupying tile	false	false	P
canCapture	1	Determines if player can capture own pieces	Blue Dog moves from (3,3) to (4,3), Blue Cat is occupying tile	false	false	P
canCapture	2	Determines if a piece of lower strength can capture an opponent piece with higher strength but in a trap	Blue cat captures Green dog	true	true	P
canCapture	3	Determines if a piece can capture an opponent with lower strength	Blue dog captures green cat	true	true	P
canCapture	4	Determines if a piece can capture an opponent with same strength	Blue dog captures green dog	true	true	P
getOwner	1	Return piece owner	p1Dog	Player 1 (Blue)	Player 1 (Blue)	P
getStrength	1	Return piece strength	p1Dog	3	3	P

getSymbol	1	Return piece symbol	p1Dog	BD	BD	P
getName	1	Return piece name	p1Dog	Dog	Dog	P
getRow	1	Return piece row	p1Dog	5	5	P
getCol	1	Return piece cal	p1Dog	1	1	P
isTrapped	1	Determines if piece is trapped in opponent trap	p1Cat	true	true	P
	2	Determines if piece is trapped when not in opponent trap	p1Cat	false	false	P
isInLake	1	Determines if piece is in lake	p2Cat	false	false	P
	2	Determines if piece is not in lake	p2Cat	true	true	P
isCaptured	1	Determines if piece is captured	p2Dog	true	true	P
	2	Determines if piece is not captured	p2Dog	false	false	P
setTrapped	1	Piece isTrapped status updated to trapped correctly	p1Dog moves into a trap	p1.Dog.isTrapped = true	p1.Dog.isTrapped = true	P
	2	Piece isTrapped status updates when no longer trapped	p1Dog moves out of trap	p1.Dog.isTrapped = false	p1.Dog.isTrapped = false	P
setInLake	1	Piece inLake status updated to in lake correctly	p1Dog moves into lake	p1.Dog.isInLake= false	p1.Dog.isInLake= false	P
	2	Piece inLake status updates when no longer in lake	p1Dog moves into lake	p1.Dog.isInLake= false	p1.Dog.isInLake= false	P
setCaptured	1	Piece isCaptured status updated when captured correctly	p1Dog is captured	p1.Dog.isCaptured = true	p1.Dog.isCaptured = true	P
	2	Piece isCaptured status updates when still active	p1Dog is not	p1.Dog.isCaptured	p1.Dog.isCaptured =	P

			captured	= false	false	
initializeBoard	1	Test if the board initializes correctly	New Board	Empty grid with lakes (~), traps (?), and home bases (#/$)	Empty grid with lakes (~), traps (?), and home bases (#/$)	P
placePiece	1	Test if a piece is correctly placed	Place p1Dog at (5,1)	p1Dog appears at (5,1)	p1Dog appears at (5,1)	P
	2	Test if multiple pieces are correctly placed	Place p1Dog at (5,1) and p1Cat at (1,1)	p1Dog (BD) at (5,1), p1Cat (BC) at (1,1)	p1Dog (BD) at (5,1), p1Cat (BC) at (1,1)	P
getPieceAt	1	Retrieve a piece given a row and column	Get piece at (5,1)	p1Dog (BD)	p1Dog (BD)	P
	2	Retrieve a non-existent piece returns null	Get piece at (6,6) (empty tile)	null	null	P
updateBoard	1	Refresh blue trap when piece moves out of it	Piece moves out of trap blue trap	?	?	P
	2	Refresh green trap when piece moves out of it	Piece moves out of trap green trap	!	!	P
	3	Piece moves to another position	Piece moves from (2,2) to (2,1)	Piece symbol moves from (2,2) to (2,1)	Piece symbol moves from (2,2) to (2,1)	P
isLake	1	Determines if position is a lake	(1,3)	true	true	P
	2	Determines if position that’s not a lake correctly	(0,0)	false	false	P
isTrap	1	Determines if position is a trap	(2,0)	true	true	P

	2	Determines if position that's not a trap correctly	(5,1)	false	false	P
getTrapOwner	1	There is a trap owner	(2,0)	player1	player1	P
	2	There is no trap owner	(5,1)	null	null	P
isEmptyTile	1	Determines if tile is empty	(3,2)	true	true	P
	2	Determines if occupied with trap tile is empty	(2,0)	false	false	P
	3	Determines if occupied with lake tile is empty	(1,3)	false	false	P
isOpponentPiece	1	Determines if owned piece is opponent piece	p1Dog, (5,1)	false	false	P
	2	Determines if opponent piece is opponent piece	p1Dog, (1,7)	true	true	P
removePiece	1	Player 1's dog is removed correctly	p1Dog	p1Dog disappears from the board, isCaptured = true	p1Dog disappears from the board, isCaptured = true	P
isHomeBase	1	Determines if tile is blue home base of current player (blue)	(3,0)	true	true	P
	2	Determines if tile is green home base of current player (green)	(3,8)	true	true	P
	3	Determines if tile is green home base of current player (blue)	(3,2)	false	false	P
displayBoard	1	Prints board grid				P
getRows()		Returns grid's amount of rows		7	7	P
getCols()		Returns grid's amount of columns		9	9	P