Description of Test	Test case	Excepted Result
testResetBoardPits: Resets the board to ensure the correct number of stones in each pit initially.	board.resetBoard();	Stones = 4
testResetBoardStores: Resets the board to ensure the correct number of stones in each store initially.	board.resetBoard();	Stones = 0
testRegisterPlayerOne: Registers players to respective stores.	board.registerPlayers (playerOne, playerTwo);	Store 1 owner = playerOne, and playerOne store = store 1
testRegisterPlayerTwo: Registers players to respective stores.	board.registerPlayers (playerOne, playerTwo);	Store 2 owner = playerTwo, and playerTwo store = store 2
testGetNumStones: When a valid pit number is input, get the number of stones in that pit (On the assumption no moves made).	pitNum = 1	Number of stones = 4
testGetNumStonesException: When a invalid pit number is input, throw PitNotFoundException.	pitNum = 13	“13 should be out of bounds”
testGetNumStonesEmpty: When a valid pit number is input, and the pit is empty, get the number of stones in that pit.	After removing pits for pit 2	Stones = 0
testGetNumStonesM	After moving stones	Stones = 5

ove: When a valid pit number is input, distribute stones from that pit (assuming this is the first move made on that pit), and get the number of stones in the next pit.	from pit 3, get number of stones in pit 4	
testDistributeStones: When a valid pit number is input, distribute stones from that pit.	After distributing stones at starting point 4, get the total numbers distributed in stores and pits.	Stones = 4 (on the assumption the game just began)
testDistributeStonesException: When an invalid pit number is input, throw PitNotFoundException.	pitNum = -2	“-2 should be out of bounds”
testDistributeStonesLowBoundary: When 0 (low boundary case) is pitNumber, throw PitNotFoundException.	pitNum = 0	“0 should be out of bounds”
testDistributeStonesHighBoundary: When 13 (high boundary case) is pitNumber throws PitNotFoundException.	pitNum = 13	“13 should be out of bounds”
testCaptureStones: When a valid pit number is input, return the number of stones captured from the opponent side (on the assumption this is the first move).	stoppingPoint = 1	Captured = 4
testCaptureStonesEx	stoppingPoint = 52	“52 should be out of

ception: When an invalid pit, throw PitNotFoundException		bounds”
testCaptureStonesLowBoundary: When 0 (low boundary case) is stoppingPoint, throw PitNotFoundException.	stoppingPoint = 0	“0 should be out of bounds”
testCaptureStonesHighBoundary: When 13 (high boundary case) is stoppingPoint, throw PitNotFoundException.	stoppingPoint = 13	“13 should be out of bounds”
testCaptureStonesAction: When valid input is entered, get the number of captured stones after a variety of moves.	startPit = 3 (p1) startPit = 4 (p1) startPit = 8 (p2) startPit = 5 (p1) startPit = 9 (p2) startPit = 2 (p1) startPit = 8 (p2)	Stones captures = 1
testIsSideOneEmpty: Before any moves are made, check to ensure playerOne’s side is full.	pitNum = 1	isSideEmpty returns false
testIsSideTwoEmpty: Before any moves are made, check to ensure playerTwo’s side is full.	pitNum = 7	isSideEmpty returns false
testIsSideFull: After removing every stone from every pit on playerOne’s side, ensure playerOne’s side is empty.	*remove stones from pits 1-6 pitNum = 1	isSideEmpty returns true
testIsSideEmptyExce	pitNum = 0	“0 should be out of

ption: When invalid input is entered, throw PitNotFoundExceptio n		bounds”