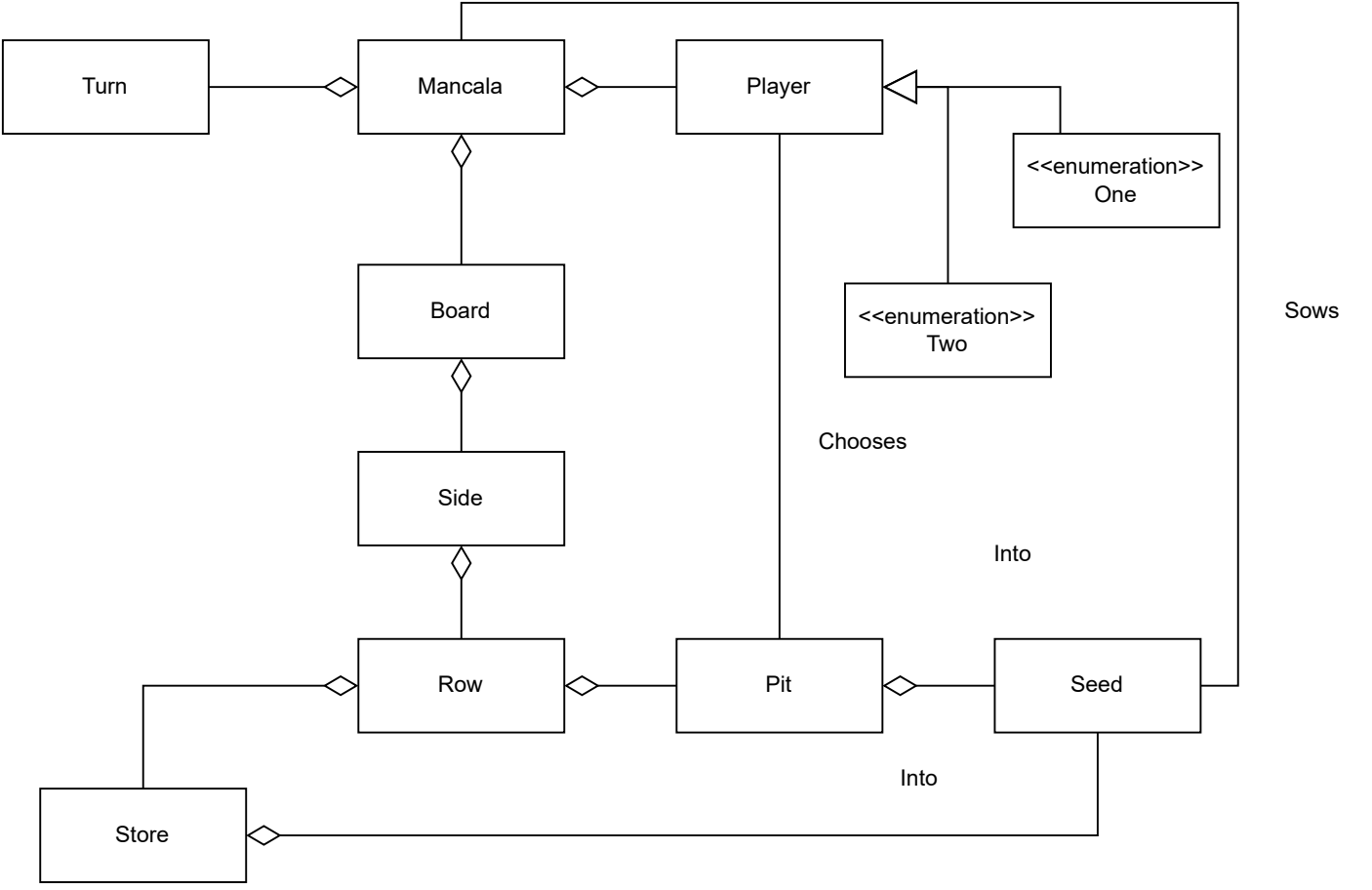
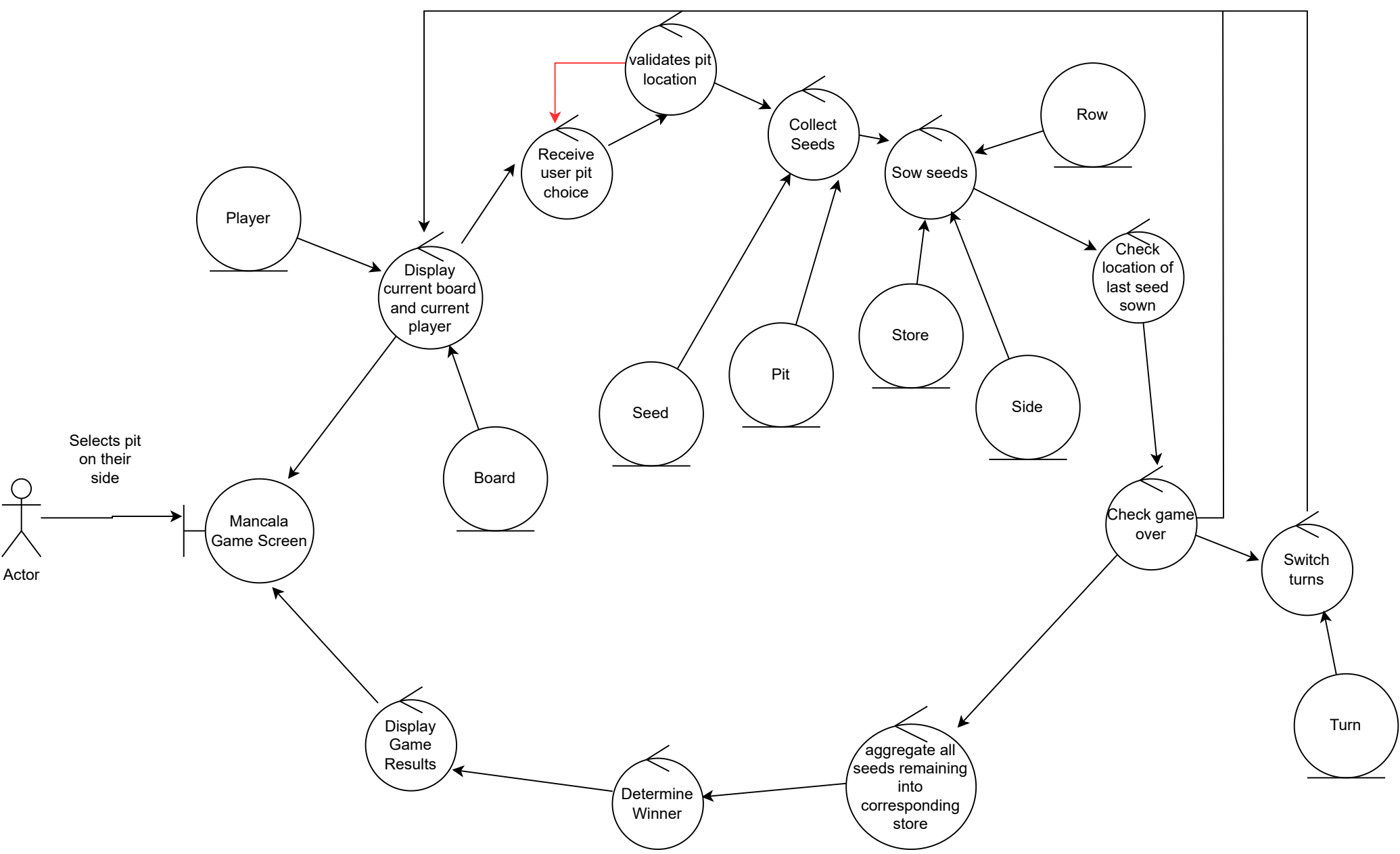
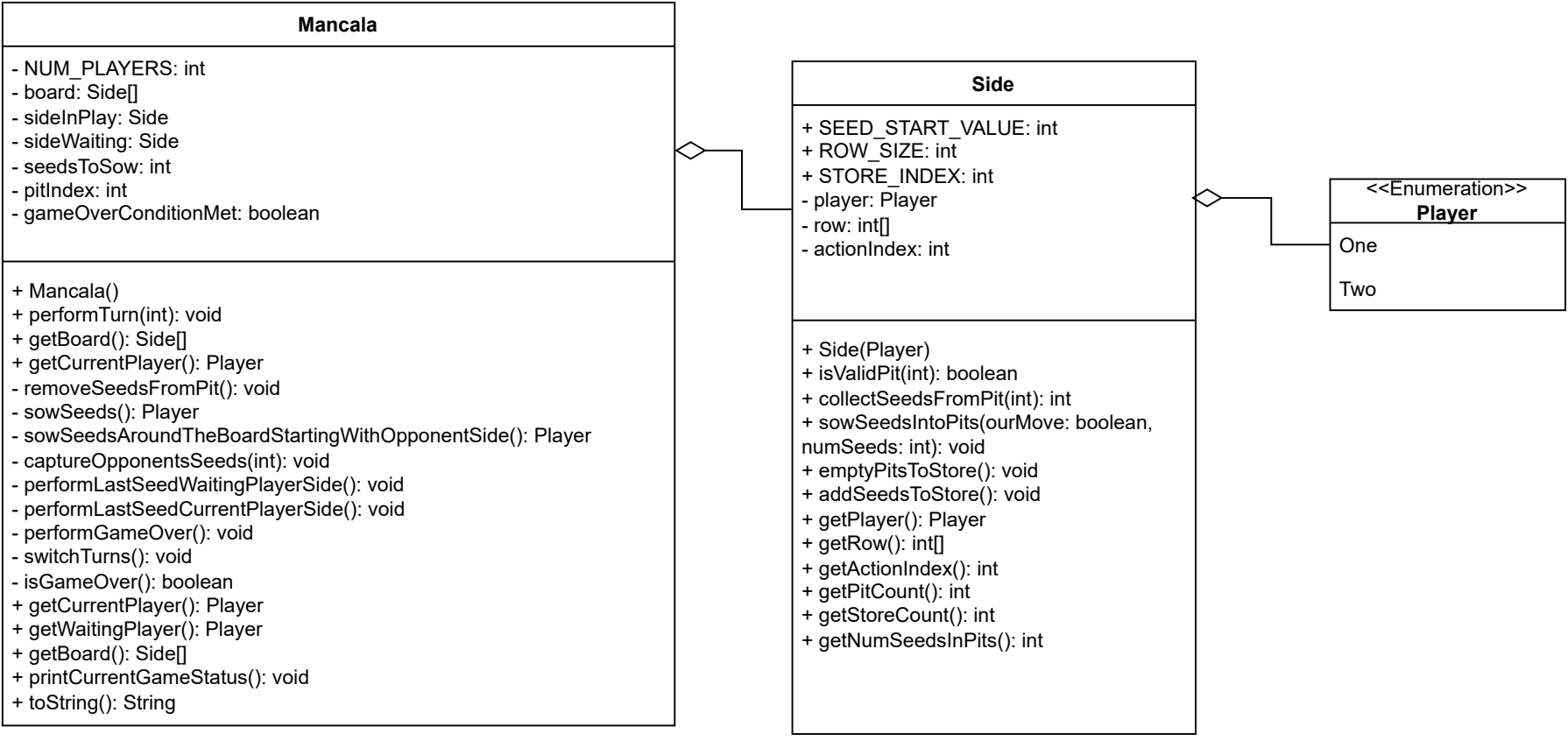
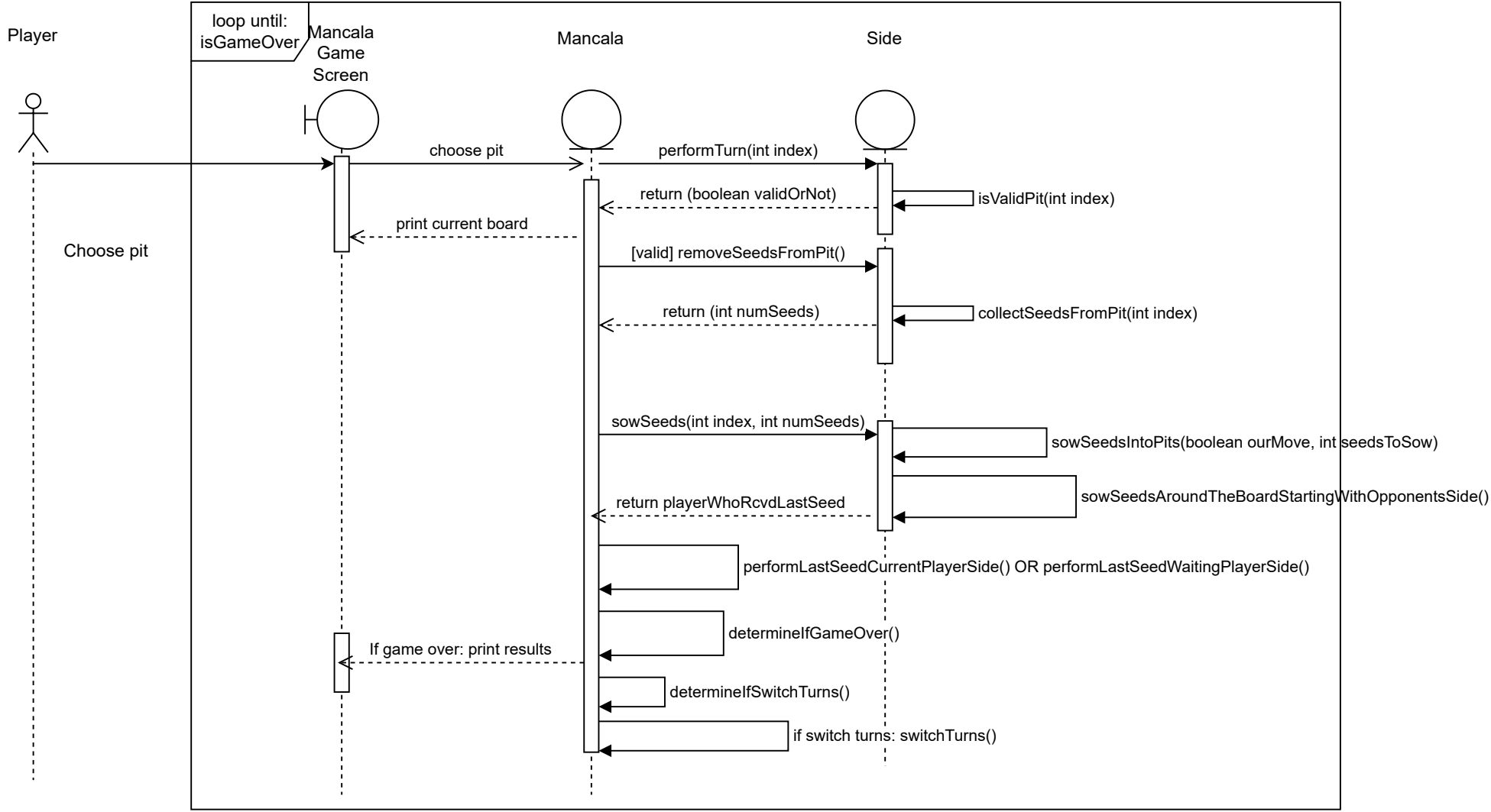

Turn
Mancala
Player
<<enumeration>>
One
Board
<<enumeration>>
Two
Sows
Chooses
Side
Into
Row
Pit
Seed
Into
Store

Actor	System
	1. Displays current board and current player.
2. Current player chooses a pit from their side of the board. [ALT-0]	
	3. Collects seeds from chosen pit and sows one seed per pit in current player's row starting with the pit to the right of the chosen pit. System sows one seed into current player's store if reached, and continues into opposite player's row and pits. It skips opposite player's store. When no more seeds to sow, system checks where final seed sown. ● If last seed landed in opponent's pit, check [Rule 1] then continue. ● Else [ALT-1].
	4. Switch turns. Continue step 2.
[ALT-0]	● Pit choice is invalid (pit count is zero or not 1-6 inclusive) ● Notify invalid choice, Continue step 2.
[ALT-1]	● If last seed landed in current player's store, check [Rule 1] then continue step 1. ● If last seed landed in current player's pit and that pit was empty, put that seed into current player's store and collect seeds from opposite player's pit and place them into current player's store. Check [Rule 1] then continue step 4.
[Rule 1] *Gameover*	Game over condition: *Current player's pits are empty or waiting player's pits are empty, if True:* ● Collect the seeds from pits that contain them and deposit them into the corresponding store with that side of the board ● Print the winner and the final board.


validates pit location
Collect Seeds
Row
Receive user pit choice
Player
Sow seeds
Display current board and current player
Check location of last seed sown
Store
Pit
Seed
Side
Selects pit on their side
Board
Mancala Game Screen
Check game over
Switch turns
Actor
Turn
Display Game Results
aggregate all seeds remaining into corresponding store
Determine Winner


Mancala
- NUM_PLAYERS: int
- board: Side[]
- sideInPlay: Side
- sideWaiting: Side
- seedsToSow: int
- pitIndex: int
- gameOverConditionMet: boolean
+ Mancala()
+ performTurn(int): void
+ getBoard(): Side[]
+ getCurrentPlayer(): Player
- removeSeedsFromPit(): void
- sowSeeds(): Player
- sowSeedsAroundTheBoardStartingWithOpponentSide(): Player
- captureOpponentsSeeds(int): void
- performLastSeedWaitingPlayerSide(): void
- performLastSeedCurrentPlayerSide(): void
- performGameOver(): void
- switchTurns(): void
- isGameOver(): boolean
+ getCurrentPlayer(): Player
+ getWaitingPlayer(): Player
+ getBoard(): Side[]
+ printCurrentGameStatus(): void
+ toString(): String
Side
+ SEED_START_VALUE: int
+ ROW_SIZE: int
+ STORE_INDEX: int
- player: Player
- row: int[]
- actionIndex: int
+ Side(Player)
+ isValidPit(int): boolean
+ collectSeedsFromPit(int): int
+ sowSeedsIntoPits(ourMove: boolean, numSeeds: int): void
+ emptyPitsToStore(): void
+ addSeedsToStore(): void
+ getPlayer(): Player
+ getRow(): int[]
+ getActionIndex(): int
+ getPitCount(): int
+ getStoreCount(): int
+ getNumSeedsInPits(): int
<<Enumeration>>
Player
One
Two


Player
loop until: isGameOver
Mancala Game Screen
Mancala
Side
Choose pit
choose pit
performTurn(int index)
return (boolean validOrNot)
isValidPit(int index)
print current board
[valid] removeSeedsFromPit()
return (int numSeeds)
collectSeedsFromPit(int index)
sowSeeds(int index, int numSeeds)
sowSeedsIntoPits(boolean ourMove, int seedsToSow)
sowSeedsAroundTheBoardStartingWithOpponentsSide()
return playerWhoRcvdLastSeed
performLastSeedCurrentPlayerSide() OR performLastSeedWaitingPlayerSide()
determineIfGameOver()
If game over: print results
determineIfSwitchTurns()
if switch turns: switchTurns()