Behavioral Contract for Moving a Worker

- Operation: playerMove(int x, int y)

- Cross References: Use case is when player tries to move a worker.

- Preconditions:
  - Game Initialized: The game must be initialized with two players, each having two workers placed on the grid.
  - Valid Turn: It must be the turn of the player attempting the move. Only the current player can move a worker.
  - Worker Selection: A worker belonging to the current player must have been selected based on its coordinates on the grid.
  - Target Cell Validity: The target cell for the worker's move must be within the grid bounds (0 to 4 for both x and y in a 5x5 grid).
  - Unoccupied Cell: The target cell must not be occupied by another worker.
  - Adjacent Move to no Dome Tower: The target cell must be adjacent to the worker's current cell and the move should not involve climbing more than one level, the cell should now have a dome.
- Postconditions:
  - Worker's New Position: If the move is valid, the worker's position is updated to the target cell.
  - Cell Occupancy Update: The target cell is now occupied by the worker, and the worker's previous cell is vacated.
  - Game State Update: Depending on the game's rules, the move might trigger a change in the game state, such as checking for a win condition or changing the turn to the next player if the move ends the current player's turn.

Below is the behavioral contract for every sub-method envoked by playerMove(int, int):
1. Selecting a Worker (selectWorkerForCurrentPlayer(x, y)):
Operation: selectWorkerForCurrentPlayer(x, y)
Cross References: Use case is when user selects a worker to perform this move action.
Preconditions:
The coordinates (x, y) provided must correspond to a cell within the game grid.
A worker belonging to the current player must be present at the specified coordinates.
Postconditions:
The worker at the specified coordinates is set as the current player's selected worker.
If no worker is found at the specified coordinates or the worker does not belong to the current player, an error message is displayed, and no worker is selected.

2. Validating the Move (isValidMove(Cell fromCell, Cell toCell)):
Operation: isValidMove(Cell fromCell, Cell toCell)
Cross References: Use case is when validating the move from Cell to Cell.
Preconditions:
A worker must be selected.
fromCell is the current cell of the selected worker.
toCell is the cell the player intends to move the worker to, derived from new coordinates (x, y).
toCell must be adjacent to fromCell.
toCell must not be occupied by another worker.
The difference in tower levels between fromCell and toCell must not exceed 1.
Postconditions:
If all conditions are met, the move is deemed valid.
If any condition fails, the move is invalid, and an error message is displayed.

3. Performing the Move (moveWorker(Worker worker, Cell toCell)):
Operation: moveWorker(Worker worker, Cell toCell)
Cross References: Use case is when worker performs move action to the cell.
Preconditions:
The move has been validated as per the conditions in step 2.
worker is the selected worker of the current player.
toCell is the validated target cell for the move.
Postconditions:
The worker's position is updated to toCell.
fromCell no longer references the worker.
toCell now references the worker.

4. Checking for Win Condition (checkWinCondition()):
Operation: checkWinCondition()
Cross References: Use case is after worker has moved to a new cell, check the level of worker.
Preconditions:
The worker has been moved to a new cell.
Postconditions:
If the worker reaches a cell with a tower level equal to the win condition (e.g., the third level), the current player is declared the winner, and the game ends.
If the win condition is not met, the game continues with the next player's turn.