Behaviour Contract

Operation: `Player.moveWorker(workerId: int, destinationX: int, destinationY: int)`

Cross References: Use Case: Player Moves Worker

Preconditions:

It is the turn of the player invoking the move operation.

The **`workerId`** corresponds to a worker owned by the current player.

The **`destinationX`** and **`destinationY`** correspond to a square on the game board that is adjacent to the worker's current location (one of the 8 squares surrounding the worker's current position).
The destination square is unoccupied or occupied only by a block that is no more than one level higher than the worker's current level.
The game stage is set to "MOVE", indicating that a move action is expected.

Postconditions:

The worker has been moved to the new location with **`destinationX`** and **`destinationY`** coordinates on the game board.
The previous location of the worker on the board is now empty or has been updated to reflect the worker's departure.
The worker's **`z`** coordinate (level) has been updated to match the height of the block on the new square, if any, or remains the same if moving to an empty or lower-level square.
A check has been performed to determine if the move results in a win condition for the current player (e.g., the worker has moved to a third-level block). If a win is detected:

- The game stage is set to "END".
- The winner is recorded as the current player.
- Further game interactions are disabled or ignored.

If the move does not result in a win:

- The game stage transitions to "BUILD", allowing the player to choose where to build next.

Assumptions:

The game board is correctly initialized and all game pieces are placed according to the rules.

The player's workers have been correctly assigned and can be uniquely identified by **`workerId`**.

Guarantees:

The move operation will not disrupt the integrity of the game state.

The rules of the game regarding worker movement are enforced.