int isHit(String userInput,GameManager gameManager):

1. **Input validation:** Verifies whether the length of the user's input (two to three characters) and returns -1 for invalid input.
2. **Coordinate Conversion:** Converts user input into the game board's row and column indices.
3. **Boundary checking:** Verifies that the column index (1–10) and row index ('A' to 'J') fall inside acceptable ranges.
4. **Handling Special scenarios:** Modifies column index for inputs of size three; verifies third character for scenarios like 'A10'.
5. **Board Access:** Utilizing computed indices, retrieves cell values from the game board.
6. **GameManager Interaction:** To update the status of the game, call calculatePointsForFortList.
7. **Hit or Miss Handling:** Adjusts the fort list and board according to the hit result.
8. **Return Values:** Indicates hit, miss, or incorrect input using integer returns (1, 2, 0, -1).
9. **Classes Involved:** Fort and GameManager

boolean exploreCells(int count,Cell startCell,char fortName):

1. **Initialization:** Retrieves the row and column indices from the location of the starting cell. Makes use of the `Cell} class to handle properties and get positions.
2. **Exploration Loop:** From the beginning, iterates across adjacent cells. Increases the count by adding accessible nearby cells to the list of used cells. If the count hits 5, which denotes completion, returns true. Uses the Cell class to store location and fort data.

3. **Recursive Exploration :** Calls the exploreCells method on neighbouring cells that haven't been investigated recursively.
4. **Termination:** If no additional cells were investigated recursively, returns false. Returns true to indicate completion if not. Uses the `Cell} class to manage forts and set termination parameters.