## 3.1 Board Component Classes

### 3.1.1 Board Class

The Board component is designed to model the game board. The Board class stores an array of the coordinates of the hexagon spaces at which the game pieces are located, an array of the coordinates of the hexagon spaces that are the boundary limits of the game board, an array of the game pieces, the enumeration of the current game piece that is active, the logger manager, the display component, and the interpreter manager. The board component not only stores the necessary coordinates and pieces, but also manipulates the position of each piece according to direct instruction from the event handlers while also checking that each action is valid. The board component notifies the logger manager and the display after each action so that each may update according to how the game board state has changed.

The Board class contains the following fields:

* HexCoord[] pieceCoords: Stores a collection of coordinates that represent the locations of the game pieces on the game board. The piece locations are represented by HexCoord elements (see Section 3.1.2), which will store the x-, y-, and z-coordinates of each piece. The array can be indexed using the enumeration assigned to each piece within the current match. This collection will be updated as the game progresses.
* HexCoord[] boardBounds: Stores a collection of the boundary limits of the board. The boundary coordinates are represented by HexCoord elements (see Section 3.1.2), which will store the x-, y-, and z-coordinates of each boundary limit. The array may be indexed if the index of the needed boundary is known, but it will mostly be iterated through to check that a hexagon coordinate is within the game board boundary limits.
* Team[] teams: Stores the collection of teams. The teams will be represented by the Team class (see Section 2.4). The teams will be manipulated and their state updated as the game is played. The teams will also be used to interact with the interpreters that live in each of the teams. This collection will be manipulated and updated as the game progresses.
* Piece[] pieces: Stores the collection of game pieces on the game board. The pieces will be represented by the Piece class (see Section 2.4). The pieces will be manipulated and their state updated as the game is played. This collection will be manipulated and updated as the game progresses.
* Logger logger: Stores the logger. The logger will be represented by the Logger class (see Section 2.6). The logger manager will be notified when an action or some critical update that needs to be logged is performed.
* Display display: Stores the display. The display will be represented by the Display class (see Section 2.1). The display will be notified when an action or some critical update that needs to be visualized is performed.

The fields in the Board class depend on the definition of the PieceEnum. The PieceEnum is an enumeration, containing a fixed set of values which map to all of the game pieces in the game. It will contain values for each game piece so that each piece has its own unique enumeration. Some values are RED\_SCOUT, RED\_SNIPER, RED\_TANK, BLUE\_SCOUT, BLUE\_SNIPER, BLUE\_TANK, and so forth.

The Board class is responsible for initializing the game, whose state will then be communicated to the display and the logger manager to be displayed and logged, respectively. The Board class will receive instructions from the Flow component (see Section 2.2) in the form of its functions being invoked by the component as events are received and handled.

The Board class has three methods for manipulating the game board. The method movePiece(PieceEnum piece, HexCoord coord) takes in the piece to move and the coordinate to which the piece is to be moved, which must represent a valid coordinate of a hexagon space and be within range of the piece’s remaining mobility points. The method shootSpace(HexCoord coord, PieceEnum piece) takes in the target coordinate of the hexagon space being shot, which must be a valid hexagon space and be within range of the piece’s range points, and the enumeration of the piece doing the shooting. Finally, the method nextPiece() does not take in any parameters but simply determines the next piece that is to be moved.

Also, the Board class has two method for scanning one space and an area. The method scanArea(PieceEnum piece, HexCoord coord) takes in the enumeration that represents the piece that is performing the scan operation and the target coordinate of the hexagon space, which must be a valid coordinate, that is being scanned and then returns a list of HexCoord elements representing the coordinates that are determined to be occupied. The method scanSpace(HexCoord coord) takes in one coordinate of a hexagon space, which must be valid coordinate, and returns a list of pieces determined to be occupying the space.

Next, the Board class has two methods for converting between absolute and relative coordinates. The method abosuleToRelative(HexCoord coord) takes in one coordinate of a hexagon space, which must be a valid coordinate representing a hexagon space on the game board, and returns a HexCoord element representing the coordinate converted into a relative coordinate. The method relativeToAbsolulte(HexCoord coord) takes in one coordinate of a hexagon space, which must be a valid coordinate representing a hexagon space on the game board, and returns a HexCoord element representing the original coordinate converted into an absolute coordinate.

Lastly, the Board class has a method for checking that a given coordinate is within the boundaries of the game board limits. The method isInBounds(HexCoord coord) takes in one coordinate representing a hexagon space on the game board and returns true if the coordinate is within the boundaries of the game board limits and false otherwise.

### 3.1.2 HexCoord Class

The HexCoord class is designed to model the hexagon space elements on the game board. The HexCoord class will be used to refer to the actual hexagon spaces on the game board and for storing the coordinate positions of the game pieces. The HexCoord class will also be used throughout the system when it is necessary to refer to a hexagon space on the game board, such as within the display component to update the visual display of a game piece.

The HexCoord class contains the following fields for representing the position of the element:

* int x: Represents the x-coordinate of the hexagon space. The x-coordinate represents the axis that declines through the hexagon from the left to the right.
* int y: Represents the y-coordinate of the hexagon space. The y-coordinate represents the axis that rises through the hexagon space from the left to the right.
* int z: Represents the z-coordinate of the hexagon space. The z-coordinate represents the axis that runs vertically through the hexagon space.

The HexCoord class’s only methods are accessors and mutators for its three fields. It contains getX(), getY(), getZ(), which all return their respective coordinate values for the space element, setX(int x), setY(int y), setZ(int x), which all take the new value for their respective coordinate value of the space element. The HexCoord class also contains the method reduce(), which simply reduces the space element’s coordinates to their minimal representation.