DESCRIPTION/SUMMARY?

### 2.4.1 Piece Class

The Piece class is mainly a container for holding data regarding the statistics and capabilities of a given piece. It is not responsible for tracking information such as the position of the piece. It serves as mostly an end point of queries.

The Piece class contains the following fields:

* + Integer attack: The attack field holds the Piece’s attack value, representing how much damage it does when it shoots an enemy.
  + Integer range: The range field holds the Piece’s range value, representing both how far the Piece can see and how far it can shoot.
  + Integer movement: The movement field holds the piece’s movement value, representing how many spaces the Piece can move per turn.
  + Integer currentMovement: The currentMovement field holds the Piece’s current movement value, representing how many more spaces the Piece can move on this turn.
  + Integer health: The health field holds the Piece’s maximum health value, representing how much total damage the Piece can absorb before it dies.
  + Integer currentHealth: The currentHealth field holds the Piece’s current health value, representing how much more damage the Piece can absorb before it dies.
  + bool alive: The alive field represents whether the Piece is currently alive. If it is not alive, it may not move, shoot or otherwise act.
  + Integer damageDealt: The damageDealt field is a running total of all of the damage this Piece has dealt to other Pieces (including itself) during this match. It has no impact on the game itself but is stored for updating statistics when the match is over.
  + Integer damageTaken: The damageTaken field is a running total of how much damage this Piece has absorbed during this match. It has no impact on the game itself but is stored for updating statistics when the match is over.
  + Integer spacesMoved: The spacesMoved field is a running total of how many spaces this Piece has moved during this match. It has no impact on the game itself but is stored for updating statistics when the match is over.
  + Integer enemiesDefeated: The enemiesDefeated field is a running total of how many opposing Pieces this Piece has defeated during this match. It has no impact on the game itself but is stored for updating statistics when the match is over.
  + Integer turnsTaken: The turnsTaken field is a running total of how many turns this Piece has taken during this match. It has no impact on the game itself but is stored for updating statistics when the match is over.
  + Integer absoluteRotation: This field is a representation of the absolute rotation of this piece, given from 0 to 5, on the board.
  + bool hasShot: This field determines whether the Piece has shot during its turn. If it has shot once, it may not shoot again.

Most of the methods associated with the Piece class are accessor methods for its field. The Board may query the piece’s fields using the methods getAttack(), getRange(), getMovement(), getCurrentMovement(), getHealth(), getCurrentHealth(), getDamageDealt(), getDamageTaken(), getSpacesMoved(), getEnemiesDefeated(), getTurnsTaken(), getAbsoluteRotation, getHasShot() and isAlive().

Some of the Piece’s fields are modified during its turn. The reset() method allows these values (hasShot and currentMovement) to be reset, so that the Piece’s next turn may be taken correctly.

During a turn, the Board component handles the spatial logic of a Piece moving and shooting, but the Piece must still handle these actions internally in order to update its statistics correctly. The shoot() method takes in Integers representing the amount of damage dealt and enemies defeated on a shot, and updates these statistics, also setting hasShot to true for this turn. The move() method takes two integers, representing the number of spaces moved and the relative direction of movement, and it decrements the value of currentMovement, updating statistics as appropriate. The takeDamage() method is called when the Piece is shot, and takes in an integer representing the amount of damage received. It calculates the Piece’s remaining health and returns the value of the alive field. All of these methods will also send a message to the GameScreen class (see Section 2.1.3), informing the Display that the Piece has performed some action in order to animate it properly.

The remaining methods in the Piece class concern the rotation of the Piece, which has no impact on the logic of the Board. The getRelativeRotation() method takes an integer representing some absolute rotation value. It compares the value given with this Piece’s absoluteRotation value, and returns the relative rotation from the Piece’s rotation to the passed-in rotation, which will be a value between 0 and 5 (that is, how many times the Piece must turn to reach the target rotatation). Conversely, the getAbsoluteRotation() method takes an integer representing some relative rotation value between -5 and 5. It checks this value against the Piece’s absoluteRotation, and returns the absolute rotation of the passed-in value. The rotate() method actually updates the absoluteRotation value of the piece, taking in a relative rotation value between -5 and 5, calling getAbsoluteRotation(), and setting the piece’s absoluteRotation to the return value. It will also send a message to the GameScreen class, informing the display that the Piece has updated its rotation.

### 2.4.2 Team Class

The Pieces are all stored in a Team class, which represents a group of Pieces working in tandem to win the match. It holds all the Pieces which are working together, the colour of the Team that is being represented, and the Interpreter associated with the Team, as well as values that help it take its turns properly.

The Team class contains the following fields:

* + Piece[] pieces: The pieces field is an array of Pieces, indexed by the PieceEnum. It contains the Pieces for this team.
  + bool eliminated: This field is a boolean value representing whether this Team has been eliminated or not. A Team is eliminated when all of its Pieces are dead.
  + TeamEnum colour: The colour field is a TeamEnum value representing the colour of the team, as described below.
  + Interpreter interpreter: The interpreter field holds the Interpreter for this Team (see Section 2.5.1). If this Team is controlled by a human player, the value of this field will be NULL.
  + Integer activePiece: The activePiece field is an integer representation of which piece is currently active and taking a turn. It can be used to index the pieces array. This value is set to -1 if there is no active piece.

Some values in the Team class depend on the PieceEnum. The PieceEnum is an enumeration containing a fixed set of values which map to the three types of piece in the game. It will contain the values SCOUT, SNIPER and TANK.

The Team’s colour is given by the value of the TeamEnum. The TeamEnum is an enumeration containing a fixed set of values which map to the six colours of teams in the game, in the order that they may take their turns. It will contain the values RED, ORANGE, YELLOW, GREEN, BLUE, and PURPLE.

The Team class contains a method for getting the next active piece. getNextPiece() will examine each Piece in the pieces array after the last value of activePiece, and calls setActivePiece() to assign the value of activePiece to that index if it is able to take a turn. If no Piece is able to take a turn, it calls the setEliminated() method, which sets its eliminated field to true. The isEliminated() method will return the value of the eliminated field.

The Team class can return its own Interpreter with the getInterpreter() method, and can order the Interpreter to run a piece’s play command with the playAI() method, which takes an integer index as a parameter and runs that Interpreter’s play() method on the correct piece.