cmpt370

**playerstats:**

Pre-condition:

Post-condition: all the attributes are proper set

* + - damagedealt
    - damageTaken
    - TanksDestoryed
    - TilesMoved

getter/setter ();

The playerstats class is using for storing the stats during the game, which includes damage dealt to enemy, damage taken from enemy, enemy tanks destoryed, and total tiles moved. Playerstats class is connected to the server class, and has an extend class called AIstats.

The attributs include

* + - * + int damagedealt

An integer represents the total damage dealt to the enemy

* + - * + int damageTaken

An integer represents the total damage taken from the enemy

* + - * + int TanksDestoryed

An integer represents the total tank destoryed

* + - * + int TilesMoved

An integer represents the total tiles moved

Methods include

* + - * + int getdamagedealt ();

get the current damage dealt to the enemy

* + - * + int getdamageTaken ();

get the current damage taken from enemy

* + - * + int getTanksDestoryed ();

get the current number of tanks destoryed

* + - * + int getTilesMoved ();

get the current number of tiles moved

* + - * + int setdamagedealt ();

set the damage dealt to the enemy

* + - * + int setdamageTaken ();

set the damage taken from enemy

* + - * + int setTanksDestoryed ();

set the number of tanks destoryed

* + - * + int setTilesMoved ();

set the total number of tiles moved

**AIstats:**

Pre-condition: playerstats class is initialized

Post-condition: The attributes are properly set

* + - turnsSinceLastMove
    - TurnsSinceLastFire

getter/setter ();

The AIstats is using to store the stats of AI player. The attributes include turnsSinceLastMove, and TurnsSinceLastFire. These attributes will prevent the AI player stop moving and don’t fire for long. The AIstats class is an extend class to the playerstats class.

The attributes include:

* Int turnsSinceLastMove
  + store the number of turns the player don’t move
* int TurnsSinceLastFire
  + Store the number of turns the player don’t fire

The methods include:

* + - * getturnsSinceLastMove ();
        + get the number of turns the player don’t move
      * setturnsSinceLastMove ();
        + set the value of turnsSinceLastMove
      * getTurnsSinceLastFire ();
        + get the number of turns the player don’t fire
      * setTurnsSinceLastFire ();
        + set the value of TurnsSinceLastFire

**playerInfo:**

Pre-condition:

Post-condition:

* + - <list>robots
    - playerID/threadID

construct (playerID)

getRobot (type);

getRobotAlive ();

getPalyerID ();

The playerInfo class is using to get and store the robots of a player. The attributes include a list of robots and an integer represent the player ID. This class will use the player ID to get the corresponding robots from the robot librarian.

The attributes include:

* + - * + <list> robots

A list that store the robots of corresponding player

* + - * + Int playerID/threadID

An integer represents the player, since planning to write the player as a thread, playerID is also refer to thread ID.

The methods include:

* + - * + constructor (playerID)

construct the playerInfo class, and store the robots belong to player with ‘playerID’ to the <list>robots

* + - * + getRobot (type)

get the robot from the <list>robots with correct type

* + - * + getRobotAlive ();

get the robots of corresponding player that are still alive

* + - * + getPlayerID ();

get the playerID which is the attribute of this class

**robots:**

Pre-condition:

Post-condition:

* + - bool alive
    - health
    - movement
    - type
    - loc[][]
    - direction
    - playerID

construct(playerID);

getter/setter ();

The robots class stores all the information related to the robots. The attributes include a Boolean variable to represent whether the robot is alive, the robots health, total movements of robots, robot type, a two dimension array which represent the location of robots on the board, the direction robot is facing, playerID of the player who own the robot.

The attributes include:

* + - * Bool alive
        + A Boolean variable represents whether the robot is alive
      * Int health
        + An integer represents the robot’s health
      * Int movement
        + An Integer represents the total number of tiles the robot moved
      * Int type
        + An integer represents the type of robot
      * <array> loc [] []
        + A two-dimensional array represents the current location of robots on the board
      * Int direction
        + An integer represents the direction robot is currently facing
      * Int playerID
        + An integer represents the player that own the robot

The methods include:

* + - * Constructor(playerID)
        + Construct the robot class
      * getAlive ()
        + get the attribute alive
      * getHealth ()
        + get the current health of robot
      * getMovement ()
        + get the total number of movements of robots
      * getType ()
        + get the type of robot
      * getLoc ()
        + get the current location of robot
      * getDir ();
        + get the direction that the robot is currently facing
      * getPlayerID ();
        + get the player ID of the player that own the robot
      * setAlive ()
        + set the attribute alive
      * setHealth ()
        + set the current health of robot
      * setMovement ()
        + set the total movement of robot
      * setType ()
        + set the type of robot
      * setLoc ()
        + set the current location of robot
      * setDir ();
        + set the current direction the robot is facing
      * setPlayerID ();
        + set the player ID