cmpt370

**playerstats:**

The playerstats class is using for storing the stats during the game. It can be divided into two parts. First part is for both human players and AI players. It records and set the damage dealt to enemy, damage taken from enemy, total tank destroyed, and total number of tiles moved. These stats will keep recording during the game. Second part is for AI player only. The class will record the number of turns since the AI player last fire and move, which can prevent the AI players don’t move and fire for long. The methods include the setter and getter for all the attributes. The setter can set the attributes equal to the parameter. The getter can get and return the value of corresponding attribute. The setters and getter don’t have pre-condition and post conditions.

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

* int turnsSinceLastMove
  + An integer store the number of turns the player don’t move
* int TurnsSinceLastFire

An integer store the number of turns the player don’t fire

Methods include

* + - * + int getdamagedealt ();

return the current damage dealt to the enemy

* + - * + int getdamageTaken ();

return the current damage taken from enemy

* + - * + int getTanksDestoryed ();

return the current number of tanks destoryed

* + - * + int getTilesMoved ();

return the current number of tiles moved

* + - * + void setdamagedealt (int x);

set the value of x as damage dealt to the enemy

* + - * + void setdamageTaken (int x);

set the value of x as damage taken from enemy

* + - * + void setTanksDestoryed (int x);

set the value of x as number of tanks destroyed

* + - * + void setTilesMoved (int x);

set the value of x as total number of tiles moved

* + - * int getturnsSinceLastMove ();

return the number of turns the player don’t move

* + - * void setturnsSinceLastMove (int x);

set the value of x to turnsSinceLastMove

* + - * int getTurnsSinceLastFire ();

return the number of turns the player don’t fire

* + - * void setTurnsSinceLastFire (int x);

set the value of x to TurnsSinceLastFire

**playerInfo:**

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 and store them into the <list> robots. The methods include constructor(playerID), getRobot(type), getRobotAlive (), getPlayerID (). The constructor will create a new class of playerInfo, initialize the palyerID equal to the parameter and store the robots belong to this player into the <list>robots. The getRobot(type) method will get the robots of type “type” from <list>robots. Method getRobotAlive () will return the robots in the <list>robots which are still alive. The method getPlayerID () will return the player ID of the player that owns the robots store in the <list> robots.

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)

Pre-condition: none

Post condition: none

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

* + - * + getRobot (type)

Pre-condition: none

Post condition: none

return the robot from the <list>robots with correct type, return null if not found

* + - * + getRobotAlive ();

Pre-condition: none

Post condition: none

return the robots of corresponding player that are still alive, return null if no alive robot

* + - * + int getPlayerID ();

Pre-condition: none

Post condition: none

return the playerID which is the attribute of this class

**robots:**

The robots class stores all the information related to the robots. The attributes include a variable to represent whether the robot is alive, the robot’s 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 methods include the constructor and the setter and getter of all the attributes. The constructor will create the robots class, and initialize the attribute playerID equal to the parameter. All other attributes will be initialized to zero for integer and null for Boolean and array. The setter will set the corresponding attribute equal to the parameter. The getter will get and return the value of corresponding attribute. The getters and setters have no pre-condition and post condition.

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)
        + Pre-condition: none
        + Post condition: none
        + create a new robot class, initialize the player ID equals to parameter and initialize other attributes equal to zero for integer and null for Boolean and array.
      * Bool getAlive ()
        + return the attribute alive that represents whether the robot is alive
      * int getHealth ()
        + return the current health of robot
      * int getMovement ()
        + return the total number of movements of robots
      * int getType ()
        + return the type of robot
      * <int, int> getLoc ()
        + return the current location of robot
      * int getDir ();
        + return the direction that the robot is currently facing
      * int getPlayerID ();
        + return the player ID of the player that own the robot
      * void setAlive (bool alive)
        + set the attribute alive
      * void setHealth (int health)
        + set the current health of robot
      * void setMovement (int movement)
        + set the total movement of robot
      * void setType (int type)
        + set the type of robot
      * void setLoc (<int, int>)
        + set the current location of robot
      * void setDir (int direction);
        + set the current direction the robot is facing
      * void setPlayerID (int playerID);
        + set the player ID