Gameboard

- Square \*theBoard[40];

- static GameBoard\* instance;

- TextDisplay \*td;

- Academic \*Arts1[2];

...

- map<string, int> buildingCatalogue;

- GameBoard();

+ static void cleanup();

+ static GameBoard\* getInstance();

+ Square\* getSquare(int n);

+ int getNumSiblings(string block);

ostream &operator<<(ostream &out, const Gameboard &g);

TextdDisplay

- char \*\* td;

- map<string, string> playerLocation; // <playername, location>

- map<string, string> improvementLocation; // <building name, number of improvements>

+ void notify(string name, int numImprovement); // improvements (numImprvements can be negative)

+ void notify(string name, string current);

ostream &operator<<(ostream &out, const TextDisplay &td);

Square

+ string name;

// money related method?

Buildings

# int purchaseCost;

# bool owned;

# bool mortgaged;

# Player\* owner;

# int numSiblings;

+ virtual bool isOwned() = 0;

+ virutal int getPurchaseCost() = 0;; // need for checking tuition non-property sqaure

+ virtual bool isMortgaged() = 0;

+ virtual void mortgage() =0;

+ virtual void unmortgage() = 0;

+ virtual void purchase(Player \*p) = 0;

+ virtual void pay(Player \*p) = 0;

Academic

- int improvementCost;

- int tuition[6];

- int improvementLevel;

- string block;

- bool blockOwned;

- TextDisplay \* td;

- void notifyTextDisplay();

+ bool isOwned();

+ int getPurchaseCost();

+ bool isMortgaged();

+ void mortgage();

+ void unmortgage();

+ void purchase(Player \*p); // check map of player with map in GameBoard

+ void improve(int numImprovements);

+ void pay(Player \*p);

+ void getTuition();

Residences

- int numResidencesOwned;

- int rent[4];

+ bool isOwned();

+ int getPurchaseCost();

+ bool isMortgaged();

+ void mortgage();

+ void unmortgage();

+ void purchase(Player \*p);

+ void pay(Player \*p);

+ void getRent();

Gyms

- bool bothOwned;

- int rent;

+ void pay(Player \*p);

+ void getRent();

NonProperty

- string name;

+ void land(Player\* p);

RUTRCup

- static int cupCount;

// constructor checks the number of cup counts

// destructor subtracts one from the number of cups

Player

- int money;

- string name;

- Square\* currPosition;

- int position;

- char piece;

- int numAssets;

- string property[28];

- RUTRCUP \*cup[4];

- int numCups;

- bool DCTimsLine;

- int payOut; // amount of money owed for rent/tuition

- bool bankruptcy; // update if payOut > money (getRent/getTuition)

+ map<string, int> buildingCatalogue;

+ virtual void roll() = 0; // update position number + the roll, set currPosition = gameBoard[position]

+ virtual void notify() = 0;

+ virtual void trade(int give, int receive) = 0; //actually trading

+ virtual void trade(int give, string receive) = 0;

+ virtual void trade(string give, int receive) = 0;

+ virtual void trade(string give, string receive) = 0;

+ virtual void isBankrupt() = 0; //

+ virtual void declareBankruptcy() = 0;

Human

Computer

+ bool accept(int give, int receive);

+ bool accept(int give, string receive);

+ bool accept(string give, int receive);

+ bool accept(string give, string receive);

+ int getMoney();