CLASS STRUCTURE v2.0

PROJECTE DE PROGRAMACIÓ 2021 | MONOPOLY

GERARD ROVELLAT CARBÓ

MARC GOT CRITG

Class Movement:

Description: Runs (functional module) a possible movement in a roll of a player.

Methods

public Movement(Box box, Player player)

Pre: --

Post: Create a movement with box and player.

public void startAction()

Pre: --

Post: Gives the reward when the player cross or falls into the start box.

public void fieldAction()

Pre: --

Post: Manages movement when player falls into the property box.

public void betAction()

Pre: --

Post: Gives the amount of the bet to the player that is doing the movement.

public void jail()

Pre: --

Post: Add one turn into the jail for the player that is doing the movement.

public void goToJail(int jail_position)

Pre: jail_position > 0

Post: Takes the player that is doing the movement jail

public void directCommand()

Pre: --

Post: Does the movement depending of the type of direct order it is.

public void runCard(Card top_card)

Pre: Card is the top card in the stack of cards.

Post: Does the movement depending of the type of card it is.

public void optionalActions(List<optionalActions> possible_actions)

Pre: List of actions is not empty.

Post: Does an optional action of the player that is playing on this turn.

public void toSell(List<Players> player_list)

Pre: List player_list is not empty.

Post: Sell one property of the player if anyone buys it.

public void toBuy(List<Players> player_list)

Pre: List player_list is not empty.

Post: Buy one property of the player if this accepts the offer.

public void luckCard()
Pre: --

Post: Run a lucky card owned by the player playing this turn.

Class Start refine Box:

Description: Runs possible actions when star box is crossed by the player.

Methods

public Start(string reward_type)

Pre: --

Post: Create a start box.

public string getType()

Pre: --

Post: Gets the type of the reward that this start box gives (type = property / type = money)

public Field fieldReward()

Pre: --

Post: Returns the property that is given as a reward

public int moneyReward()

Pre: --

Post: Returns the amount of money that is given as a reward

Class Field:

Description: Contains all the characteristics and actions of a property.

Methods

public Field()

Pre: --

Post: Creates a Property with the input attributes.

public void changeBoughtState()

Pre: --

Post: Change state of a property.

public int getPrice()

Pre: --

Post: Returns price of property.

public int getRent()

Pre: --

Post: Returns rent of the property.

public Player getOwner()

Pre: --

Post: Returns owner of the property.

public void build()

Pre: --

Post: Build one house on the property.

public boolean buildable()

Pre: --

Post: Returns TRUE if the property its buildable FALES otherwise

Class Player:

Description: Contains all the characteristics and actions of a Player.

Methods

public Player()

Pre: --

Post: Creates a Player with the input attributes.

public void pay(int amount)

Pre: --

Post: Subtract money paid to another player or banker

public void charge(int amount)

Pre: --

Post: Add money paid to another player or banker.

public List<Card> getLuckCards()

Pre: --

Post: Returns a list of luck cards that player have.

public void addLuckCard()

Pre: --

Post: Add luck card to the player luck cards.

public void removeLuckCard()

Pre: --

Post: Remove luck card to the player luck cards.

Class DirectCommand:

Description: Executa una accio possible en una tirada d'un Jugador.

Methods

public DirectCommand()

Pre: --

Post: Create a DirectOrder with input attributes.

public Card getCommandType()

Pre: --

Post: Returns the card for this DirectOrder.

public int getFineAmount()

Pre: --
Post: Returns the amount of the fine.

Class Monopoly:

Description: Executa una accio possible en una tirada d'un Jugador.

Methods

public Monopoly(ArrayList<Player> player_list)

Pre: --

Post: Create Monopoly with the input attributes.

public void play()

Pre: --

Post: General that manage the flow of the game turns.

private int movePlayer()

Pre: --

Post: Returns the number of boxes that player have to cross.

private Box getActualBox()

Pre: --

Post: Returns the actual Box

private Boolean checkEndGame()

Pre: --

Post: Returns TRUE if the game its end FALSE otherwise

private int activePlayers()

Pre: --

Post: Returns the number of players without bankruptcy.

private void endTurn()

Pre: --

Post: Do the final possible actions in a turn and select the next player.

private Pair<Integer,Integer> throwDice()

Pre: --

Post: Returns the dice result.

private void startGame()

Pre: …

Post: …

private void endGame()

Pre: …

Post: …

Class JsonManager:

Description: Manage Json files (Functional module).

Methods

public JsonManager(string rules_file, string board_file)

Pre: --

Post: Create JsonManager class with name of files.

public Monopoly readFile()

Pre: --

Post: Returns the Monopoly game with configurations from rules and board files.

public void writeFile()

Pre: --

Post: Write the development file of the game.

private void readRules()

Pre: --

Post: Read the rules file.

private void readBoard()

Pre: --

Post: Read the board file.