MONOPOLY

LLIURAMENT INTERMITG – PROJECTE PROGRAMACIÓ

GERARD ROVELLAT I MARC GOT   - 25/04/2021

Monopoly.java

Atributs

```
private ArrayList<Player> players;
private Board board;
private Pair<Integer,Integer> dice_result;
private int player_iterator = 0;
private Player actual_player = players.get(player_iterator);
private ArrayList<optionalActions> optionalActions;
private Stack<Card> cards;
```

Metodes

```
public Monopoly(Board board,ArrayList<optionalActions> optionalActions)
pre true
post Create Monopoly with the input attributes

public void play()
pre true
post General that manage the flow of the game turns

private void movePlayer()
pre true
post Returns the number of boxes that player have to cross

private Box getActualBox()
pre true
post Returns the actual Box

private Boolean checkEndGame()
pre true
post Returns TRUE if the game its end FALSE otherwise
```

private int activePlayers()

pre true

post Returns the number of players without bankruptcy

private void endTurn()

pre true

post Do the final possible actions in a turn and select the next Player

private void throwDice()

pre true

post Returns the dice result

private void startGame()

pre true

post initialize the game start conditions

private void endGame()

pre true

post finalize the game and associated outputs and data (and Kill Thanos)

PLayer.java

Atributs

private Box actual_position;

private Player active_player;

Metodes

public void Movement(Box box,Player player)

pre \p actual_posicion and \p active_player valid

post Create a movement with box and player

public void startAction()

pre true

post Gives the reward when the player cross or falls into the start box

public void fieldAction()

pre true

post Manages movement when player falls into the property box

public void betAction()

pre true

post Gives the amount of the bet to the player that is doing the movement

public void directComand()

pre true

post Does the movement depending of the type of direct order it is

public void runCard(Card card)

pre true

post execute the argument card

public void optionalActions(ArrayList<optionalActions> possible_actions)

pre true

post ask player and execute posible optional actions interface optionalActions

Metodes

public String toString()

pre true

post output management

public void execute()

pre true

post implementation optional action is executed

Buy implements optionalActions

Atributs

private ArrayList<Player> players_list;

Metodes

public String toString()

pre true

post output management

public void execute()

pre true

post Buy optional action executed

Sell implements optionalActions

Atributs

private ArrayList<Player> players_list;

Metodes

public String toString()

pre true

post output management

public void execute()

pre true

post Sell optional action executed

LuckCard implements optionalActions

Atributs

private ArrayList<Player> players_list;

Metodes

public String toString()

pre true

post output management

public void execute()

pre true

post LuckCard optional action executed

Card.java

Atributs

    private String type;

    private boolean postposable;

Metodes

    public Card(String type,boolean postposable)

    pre true

    post main constructor

    public boolean isPostposable()

    pre true

    post true if the card is postposable false if not

CardCharge extends Card

Atributs

    private int quantity;

Metodes

    public CardCharge (String type, boolean postposable, int quantity)

    pre true

    post card constructor

public void execute(ArrayList<Player> players, Board board,int quantity)

pre true

post Charge card executed

CardFine extends Card

Atributs

private int quantity;

Metodes

public CardFine (String type, boolean postposable, int quantity)

pre true

post card constructor

public void execute(ArrayList<Player> players, Board board,int quantity)

pre true

post Fine card executed

CardGet extends Card

Metodes

public CardGet (String type, boolean postposable)

pre true

post card constructor

public void execute(ArrayList<Player> players, Board board)

pre true

post Get card executed

CardGive extends Card

Metodes

public CardGive (String type, boolean postposable)

pre true

post card constructor

public void execute(ArrayList<Player> players,Board board)

pre true

post Give card executed

CardGo extends Card

Atributs

private int position;

Metodes

public CardGo (String type, boolean postposable,int position)

pre true

post card constructor

public void execute(ArrayList<Player> players, Board board,int position)

pre true

post Go card executed

CardPay extends Card

Atributs

private int quantity

Metodes

public CardPay (String type, boolean postposable, int quantity)

pre true

post card constructor

public void execute(ArrayList<Player> players, Board board,int quantity)

pre true

post Pay card executed

Box.java

Atributs

public int position;

Metodes

public Box(int position)

pre true

post main constructor

public int getPosition()

pre true

post returns number of box position

Bet extends Box

Metodes

public Bet(int position)

pre true

post main constructor

Field extends Box

Atributs

private String name;

private int price;

private String group;

private int basic_rent;

private int group_rent;

private String buildable;

private int max_buildings;

private int building_price;

private boolean hotel;

private int hotel_price;

private ArrayList<Integer> buildings_rent;

private int hotel_rent;

private Player owner;

private int builded;

private boolean bought = false;

Metodes

public Field(int position,String name,int price,String group,int basic_rent,int group_rent,String buildable,int max_buildings,int building_price,boolean hotel,int hotel_price,ArrayList<Integer> buildings_rent,int hotel_rent)

pre true

post Creates a Property with the input attributes

public void buy(Player owner)

pre true

post Add player as owner and change state of field to true

public void sell()

pre true

post Remove player as owner and change state of field to false

public int getPrice()

pre true

post Returns price of property

public int getRent()

pre true

post Returns rent of the property

public Player getOwner()

pre true

post Returns owner of the property

public void build()

pre houseBuildable() = true and player has already pay the building price

post Build one house on the property

public boolean houseBuildable()

pre true

post Returns TRUE if the property its buildable FALES otherwise

public boolean hotelBuildable()

pre true

post Returns TRUE if the property its buildable FALES otherwise

public int priceToBuild()

pre true

post return the price the build

public boolean isBought()

pre true

post true if the field is already bought

Start extends Box

Atributs

private String type; // field,money,both

private Field field_reward;

private int money_reward;

Metodes

public Start(int position,String reward_type)

pre true

post Create a start box

public void setFieldReward(Field field_reward)

pre true

post Sets the property that is given as a reward

public void setMoneyReward(int money_reward)

pre true

post Sets the the amount of money that is given as a reward

public String getType()

pre true

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

public Field fieldReward()

pre true

post Returns the property that is given as a reward

public int moneyReward()

pre true

post Returns the amount of money that is given as a reward directComand extends Box

Atributs

    private Card function;

Metodes

    public directComand(int position,Card function)

    pre true

    post main constructor

    public Card getCard()

    pre true

    post return direct comand card associated

Board.java

Atributs

    private SortedMap<Integer,Box> board;

    private HashMap<String,Player> players;

Metodes

    public Board ()

    pre true

    post main constructor

public void addPlayer(Player player)

pre true

post add player to players in the board

public void movePlayer (Player player, int position)

pre true

post move argument player to defined position

public void addBox(Box box)

pre true

post add box to board

public boolean haveOwner(Field box)

pre true

post true if argument board is already bought flase otherwise

public Box getBox(Player player)

pre true

post get player actual box (box in player position)

JSONManager.java

Atributs

private Monopoly monopoly;

private String rules_file;

private String board_file;

Metodes

public JSONManager(String rules, String board)

pre true

post Create JsonManager class with name of files

public Monopoly readFile()

pre true

post Returns the Monopoly game with configurations from rules and board files

public void writeFile()

pre true

post Write the development file of the game

private void readRules()

pre true

post Read the rules file

private Board readBoard()

pre true

post Read the board file