TEAM “Captain Atom”

CONSOLE GAME “Monopoly”

We are:

- Delyan Nikolov - delyan_nikolov_1992
- Denislava Shentova - denislava.shentova
- Mihail Penov - hustled
- Mirela Napetova - mimirerelala
- Tanya Petkova - Tanya
- Zhulien Ivanov – zhulien

The URL of our repository:
https://captainmonopoly.codeplex.com

Game Description

We have decided to make a console version of the famous board game "Monopoly" using C# as a programming language. The rules we have implemented are very similar to the original ones:

The game begins with four players, staying on the Go space on the board. Each player begins with a total amount of $1500 cash. The bank is imaginary and never runs out of money. Each player, in turn, rolls both dices and then moves forward (clockwise around the board) the number of spaces indicated by the sum of the numbers rolled. Action is then taken depending on the space on which the player lands:

- If the player lands on an unowned property, he may buy it for the price listed on that property's space. If the player agrees to buy it, he pays the Bank the amount shown on the property space and receives the deed for that property. There can only be one player per colour. Railroads and utilities are also properties.
- If the player lands on a property owned by another player, he pays rent to that person, as specified on the property's deed. It is the property owner's responsibility to demand rent.
- If the player lands on his own property, nothing happens.
- If the player lands on Luxury Tax/Super Tax, he must pay the Bank $100
- If the player lands on Income Tax he must pay the Bank $200
- If the player lands on Chance or Community Chest, the player takes a card from the top of the respective pack and performs the instruction given on the card.
- If the player lands on the Jail space, he is "Just Visiting" and only pays a $50 tax.
- If the player lands on the Go to Jail square, he is moved directly to the Jail.
- If the player lands on or passes Go in the course of his or her turn, he or she receives $1 from the bank.
- If a player does not have sufficient funds to pay off a rent or fee, he must sell the most expensive property of his own, until he can cover his duties. Otherwise the player is out of the game. The winner is the last player left in the game.

Implementation Brief

The Main method initializes an instance of the Monopoly class which contains all needed components of the game (properties, players, board, etc). While there is more than one player (an instance of class Player) left, the game continues. Every player takes his turn by invoking the ThrowDice method. If the player has a pair and it's not his third pair in a row, he throws the dices again. If the dice pair is third in a row, the player is moved to the jail and obligated to pay $50 using the GiveMoney method (returns bool). If the player can not pay the $50, he is considered bankrupt and is deleted from the list of the players. When the player rolls the dices, he moves forward to the position that equals the sum of his current position + the sum of the two thrown dices (implemented with a one dimensional array and the method Move which belongs to the Player class). When the player is moved to his new position on the board, a CheckPosition method is invoked to determine the action that should be taken according to the current position of the player. The CheckPosition method implements the logic of the actions that every board field defines.

If the player has landed on some of the fields defined as Community Chest or Chance, the corresponding Draw method is invoked. The switch:case in the Draw methods imitates a drawing of a card, and every card has a specific instruction which determines the following actions. A counter is used to keep track of the cards and achieve an easy queue-like effect to guarantee that the drawn card is always the card on the top of the deck. There are various instructions written on the cards based on which a different action is taken. The actions can include changing the player's position, transferring money from one player to another, or simply spending or earning cash for some reasons. The giving of the money uses the GiveMoney method to first check if the player has enough cash for the transaction and if he does (the HasMoney method returns true), the transaction is made setting the appropriate cash value. If the player doesn't have enough money (HasMoney returns false) the player is required to start selling the properties he owns (using the SellProperty method) until he has enough cash for a successful transaction. If the player doesn't have enough cash though he has sold all of his properties, he is considered bankrupt and is deleted from the list of the players. The receiving of money from another player uses the method GetMoneyFromPlayers which invokes the GiveMoney method for every player from the list of active players, except for the current player who's to receive the money.

If the player finds himself on a regular property field, the field is first checked for if it’s already bought. If the propriety on the certain position is still unsold (indicated with a null owner value in a list of properties) and the player has enough cash to buy the property (HasMoney called with the price of the property returns true) the user is asked to decide whether to buy it or not, by simply pressing the Y (yes) or N(no) keyboard key. If the user decides to buy, the property object is added to the list of properties owned by the current player. If the property field already has an owner (value != null), a check is made to see if the owner is a player different than our current one. If the owner is different, the current player has to pay a rent to the player who owns the property using the GiveMoney method.
When the player position is beyond 40 (the length of the array containing the board fields) or he has drawn a card which makes him move to the start position, a certain amount of money is added to his cash. In the case of a position greater than 40, the new position becomes the remainder of the division of the position with 40.
The game ends when there is only one player left and he is considered the winner.
The graphical representation of the logic uses printing methods to inform the user what is happening in the game. The board is printed on the console implemented with a coloured char matrix. Card graphics are printed next to the board, again using char matrices generated from 8-color palette image files. On the right of the board statistics are shown, including the all players’ cash values and the list of their own properties.