Emre YILMAZ

1901042606

Computer Engineering

Gebze Technical University

Some information about the code:

1. 2 objects are created for each game. Firstly, I let user plays these games until the games are finished or the user wants to exit the game. Secondly, the games are sent to playAutoAll function and I let the computer plays the games until the games are finished or user wants to exit the game.
2. Output is printed to terminal and txt files. Each game's TXT files are different. Different objects of same games are in the same TXT file.
3. The second game *EightPuzzle* can be finished but you should wait for a long time to see the finish while the computer plays automatically. If you want to observe the end of the game *EightPuzzle*, you should remove the printing part of that game to make computer faster. In this way, the computer can finish that game faster and you can observe the end of the game *EightPuzzle* easily. But, again, you do not need to worry since there will be EXIT button while the computer is playing. You do not have to wait that computer finishes the game.
4. All the functions are used in main function directly or indirectly

• Class BoardGame2D

- **playUser** function takes a string as a parameter and plays the game accordingly. The parameter depends on the game, for the game of Peg Solitaire the string could be "2B UP". It is chosen **as pure virtual** function since its implementation is specific for all derived classes and there is no need to implement in base class

- **final playUser** It does not take any parameters, it takes a string from the user for the next move in a loop and plays the game until it is over. It also prints the board between the moves. The function asks the user for input, then calls the function **playUser(string)** above to make movement. It also checks whether user wants to terminate the game.

- **playAuto** plays the game by the computer for one move. This function is written **as pure virtual** function since its implementation is specific for all derived classes and there is no need to implement in base class.

- **final playAutoAll** plays the game until it is over. This one calls playAuto for all the moves. It also prints the board between the moves after some pause.

- **Print** and **operator<<** prints the game on the screen starting from the top left corner of the terminal. See ANSI escape sequence to move your cursor anywhere on the screen. **Print** function is marked **as pure virtual** function since its implementation is specific for all derived classes and there is no need to implement in base class. **Operator<<** function calls the **print** function to print the board of the game according to true game type.

- **endGame** returns true if the game is ended. It Is marked as pure virtual function since its implementation is specific for all derived classes and there is no need to implement in base class.

- **boardScore** e returns an int score value for the current board. It returns a positive integer that indicates the goodness of the current board. Smaller the returned value, better the board. If the game is finished for the current board, it returns return 0, which is the best case. It Is marked as pure virtual function since its implementation is specific for all derived classes and there is no need to implement in base class.

- **initialize** initializes the board. For some games the initial board is the same, for other games the initial board is random. It Is marked as pure virtual function since its implementation is specific for all derived classes and there is no need to implement in base class.

- **static playVector** function takes a vector of BoardGame2D * objects. It plays all the games in the vector until they end. This function calls the **playAutoAll** function.

- There is **information** function in addition to the functions in the pdf. This function prints the board and tells the user input format.

• Class PegSolitaire

- This class has an inner class named **Cell.** This inner class has variable that indicates coordinates *row-column* and a variable indicates cell type *peg-empty-none.* Also, this class has its getter and setter functions to get and set the *row*, *column* and *celltype* variables

- **virtual playUser** asks user for coordinates and direction. After checking the input, it controls that the move is true or not using **moveValid** function. If the move is true, makes move using **makeMove** function and updates the board. This function uses **moveValid** and **makeMove** functions.

- **bool moveValid** function checks that the move is valid or not; if the move is valid, returns true. It checks that the coordinates that has been entered by user is *peg*, and the coordinates that will be moved is *empty*

- **makeMove** makes move. Updates the board according to valid move.

- **string getRandomMove** function generates randomly direction choice. (Up, Down, Left or Right). This function is marked as const since it is expected that the function will not change anything in class.

- **virtual playAuto** generates random coordinates and direction choice. Then it controls that the move is true or not. If the move is true, makes the move and updates the board. If the move is not true, it generates a new one until makes a valid movement. This function uses **getRandomMove**, **moveValid** and **makeMove** functions

- **virtual print** function prints the board with coordinates. This function is marked as const since it is expected that the function will not change anything in class. This function is marked as const since it is expected that the function will not change anything in class.

- **virtual endGame** tries to make move from all the cells, if there are no valid move then returns TRUE that means the game is finished. It uses moveValid function. The function tries to make move from all the cells, if there are no valid move then returns TRUE that means the game is finished. This function is marked as const since it is expected that the function will not change anything in class.

- **virtual boardScore** it counts true moves that has been made. This function is marked as const since it is expected that the function will not change anything in class.

- **virtual initialize** initiliaze the board as second board type. The board is generated manually. It calls the **initGameTypeTwo** function to initiliaze the board.

- **InitGameTypeTwo** function returns the type two peg solitaire board vector.

- **setBoard** function is a setter function to update the board.

- **information** function prints the board and input format to inform the user. This function is marked as const since it is expected that the function will not change anything in class.

• Class EightPuzzle

- **virtual playUser** asks user for coordinates and direction. After checking the input validity, it controls that the move is true or not using **moveValid** function. If the move is true, makes move using **makeMove** function and updates the board. This function uses **moveValid** and **makeMove** functions.

- **bool moveValid** function checks that the move is valid or not; if the move is valid, returns true. It checks that the coordinates that has been entered by user is not *empty*, and the coordinates that will be moved is *empty*

- **makeMove** makes move. Updates the board according to valid move.

- **string getRandomMove** function generates randomly direction choice. (Up, Down, Left or Right). This function is marked as const since it is expected that the function will not change anything in class.

- **virtual playAuto** generates random coordinates and direction choice. Then it controls that the move is true or not. If the move is true, makes the move and updates the board. If the move is not true, it generates a new one until makes a valid movement. This function uses **getRandomMove**, **moveValid** and **makeMove** functions

- **virtual print** function prints the board with coordinates. This function is marked as const since it is expected that the function will not change anything in class.

- **virtual endGame** tries to make move from all the cells, if there are no valid move then returns TRUE that means the game is finished. It uses moveValid function. The function tries to make move from all the cells, if there are no valid move then returns TRUE that means the game is finished.

- **virtual boardScore** returns number of numbers that is in the wrong place.

- **virtual initialize** calls the function init() to initiliaze the board.

- **init** function fills the table correctly as finished form and then shuffles it using the **shuffle** function. So we can make sure that the table is solvable.

- **shuffle** generates and makes 200 movement randomly to shuffle the board.

- **InitGameTypeTwo** function returns the type two peg solitaire board vector.

- **setBoard** function is a setter function to update the board.

- **information** function prints the board and input format to inform the user. This function is marked as const since it is expected that the function will not change anything in class.

• Class Klotski

- **virtual playUser** asks user for block name and direction. After checking the input validity, it controls that the move is true or not using **moveValid** function. If the move is true, makes move using **makeMove** function and updates the board. This function uses **moveValid** and **makeMove** functions.

- **bool moveValid** function checks that the move is valid or not; if the move is valid, returns true. It checks that the cell that will be moved is *empty.*

- **makeMove** makes move. Updates the board according to valid move.

- **string getRandomMove** function generates randomly direction choice. (Up, Down, Left or Right). This function is marked as const since it is expected that the function will not change anything in class.

- **virtual playAuto** generates random block names and direction choice. Then it controls that the move is true or not. If the move is true, makes the move and updates the board. If the move is not true, it generates a new one until makes a valid movement. This function uses **getRandomMove**, **moveValid** and **makeMove** functions.

- **virtual print** function prints the board. This function is marked as const since it is expected that the function will not change anything in class.

- **virtual endGame** returns true if the game is ended. It checks *that the block B* is in the exact exit coordinates *(3,1) (3,2) (2,2) (2,1)*

- **virtual boardScore** returns the distance between the point of left-top corner of the block B and (2,1) coordinates that its corner has to be.

- **virtual initialize** initiliazes the board.

- **setBoard** function is a setter function to update the board.

- **information** function prints the board and input format to inform the user. This function is marked as const since it is expected that the function will not change anything in class.