# CMPE 150 Fall 2015

# Project II

*Mustafa Enes ÇAKIR* ***2013400105***

## *Project Description:*

In this project, we need a board. It has 16 characters, and one of them is a star. User can change board configuration, or he/she can use default configuration. Before he/she starts to play game, he/she have to decide how many movements he/she want to do. User will change position of the star and try to gain points. P is one point, and G is five points. When user changes star's position, it swaps character at target index. If the character at target index is P or G, it alters to I. When you on edges, you can't do some movements. Move star to up means decrease its index by 4, or move star to down means increase its index by 4. In addition, move star to left means decrease its index by 1, or move star to right means increase its index by 1. We need to print current board configuration, and current score in each turn. In bonus part, we need to calculate possible maximum point with taken number of moves.

## *Project Solution:*

Totally I used 4 *for* loops. (taking board configuration, taking user movement inputs, printing board, swaping characters )

I have 6 (5 normal, 1 bonus) *static methods* in addition to my *main* method. Each method responses some part of the project.

I keep my board in a string, and it has 16 characters. Its name is *board*. I create it in my *main* method. At the beginning it has default configuration. *If statement* in the *main* method check user demand. Default configuration or not. If user wants to change it, he/she type "No". I set *board* variable to empty string in if statement. *For* loop iterates 4 times, because board has 4 lines. In each iteration entered line added up to board configuration. When configuration processes is end up, game starts.

My *play* method manages entire game. Its parameters are my board string and *keyboard* variable that is *Scanner*. I didn't play game in main method, and I separate it to different method. So with a board configuration, we can play game anywhere, anytimes. I took number of the moves from user, and holds in movesCount variable. I calculated possible maximum score at here and store in possibleMax

variable, because for this method we need number of moves, and initial board configuration. In next part of this method we change our board configuration based on user inputs. So we have to calculate possible maximum point before changes. Next for loop manages turns of the game. If user enter 5 for the movesCount, it iterates 5 times. Each time it takes movement from user and keeps in move string; finds index of star and holds in index integer; calculates star's new targetted position and stores in targetIndex variable; then makes movement and change board variable's value, in the other word swap star with character on target index; prints current board; finally prints current score. This for loop make them in each turn. At the last part of the play method prints thanks message and possible maximum point that we stored in possibleMax variable.

My *calcTargetIndex* method calculates target index and return it. Target index is index that user want to move star. Its parameters are move string and index integer. We don't need our board configuration for calculating target index. Because we know it has 4 rows and 4 columns, it is enough for us. Move up means decrease index by 4. If our star is already at first row, we can't move up anymore. At first row indexs are 0,1,2,3. So for moving up, our star's index have to be bigger than 3. Move down means increase index by 4. If our star is already at last row, we can't move down anymore. At last row indexs are 12,13,14,15. So for moving down, our star's index have to be smaller than 12. Move left means decrease index by 1. If our star is already at first column, we can't move left anymore. At first column indexs are 0,4,8,12. So for moving left, our star's index have to be not equal to 0 in mod 4. Move right means increase index by 1. If our star is already at last column, we can't move right anymore. At last column indexs are 3,7,11,15. So for moving right, our star's index have to be not equal to 3 in mod 4. if and else if statements in this method checks this situations. If one of them is matched it returns target index immediately, so it exits from method. If none of them isn't matched it return unchanged index.

My *makeMove* method swaps our star with the character on target index and return new board configuration. Its parameters are our board, star's current index, and target index. This for loop comes over every character on our current board configuration. And add each character to newBoard variable in some conditions. When our i equals to target index, it means that we are on new position of our star. So we should add star(*) to new board instead of character that was here on old board. When our i equals to index, it means that we are on old position of our star. So we should add the character on target index on old board or "I" to new board. Because character on target index on old board might be "P" or "G", so we have to alter it. Nested if-else checks this conditions. For other characters we just add old board to new board. Finally this method returns new board configuration.

My *calcScore* method calculates new total score and return it. Its parameters are our board, target index, and score. If the character on this target index is G, it adds 5 to our score; If the character on this target index is P, it adds 1 to our score. And it returns new score.

My *printBoard* method prints the board. It doesn't return something. It just takes our current board configuration as parameter. This for loop manages line numbers. We have 4 lines so it iterates 4 times. In each iteration, it prints a row from board, so it prints a part of the board that contains 4 characters.

For bonus part:

My *calcMaxScore* method calculates possible maximum score and return it. Its parameters are my board and number of moves. At the beginning I put a if statement. When number of moves equals to 0, it should be return 0, otherwise it enters endless loop. I called same method in this method with different parameters. I don't want to change my initial board because I need it. So I create a temporary board and save new boards in it. Section 1,2,3, and 4 almost same in this method. So I explain just one of them. One thing that changed is direction of move. First of all, I calculated target index of my move, and calculate score. I put zero for score parameter, because I want to calculate score that moves bring to me. After I make my movement, I save new board to tempBoard. And I called *calcMaxScore* for calculating rest of the movements maximum score. But already I made one movements so I decrease my movesCount by 1. I am trying to find maximum score, for this purpose I use Math.max method. With this method I was able to choose higher one.

# *Implementation:*

```
import java.util.Scanner; // We need to import Scanner class, because we want read data from user

public class MEC2013400105 {
    /*
    * This method is our main method, so it says to compiler what it needs execute.
    */
    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in); //keyboard is used for taking input from user.
        String board = "ABGGRTFPPKVIGVJ*"; // board is variable that keeps board configuration. As default, it
is keeps default board configuration.
        System.out.println("Welcome to this weird game of P*G?");
        System.out.print("Do you want to use the default board configuration? ");

        if(keyboard.next().equals("No")){ // Does user wants default configuration or not?
            board = ""; // User didn't want default configuration. So I reset board.
            for(int i = 1; i <= 4; i++){ // This for loop manage number of lines in board.
                System.out.printf("Enter row %d of the board: ",i );
                board += keyboard.next(); //It adds up new lines to new board configuration.
            }
            System.out.println();
        }
        play(board, keyboard); // Let's play
    }

    /* This method control the entire game. It takes 2 parameters.
            board: Current board configuration
            keyboard: Used for taking input from user
    */
    public static void play(String board, Scanner keyboard) {
        int score = 0; // score: user's current total score. At the beginning of the it is 0, because user hasn't
played yet.
        printBoard(board); // Print board at the beginning.
        System.out.print("\nHow many moves do you want to make? ");
        int movesCount = keyboard.nextInt(); // movesCount keeps number of moves that user wants, it comes
from user input.
        int possibleMax = calcMaxScore(board , movesCount); // possibleMax stores possible maximum score
from initial board.
        System.out.println("Make a move and press enter. After each move, the board configuration and your
total points will \nbe printed. Use A for Left, S for Right, W for Up, and Z for Down.\n");
        for(int i = 0; i < movesCount; i++){ // This for loop manage turns of game. It iterate movesCount times.
            String move = keyboard.next(); // move holds movement that user wants, so it comes from user
input. It should be 'A','W','S', or 'Z'
            int index = board.indexOf('*'); // index stores current position of the star('*')
            int targetIndex = calcTargetIndex(move, index); // targetIndex stores index that user want to move
            score = calcScore(board, targetIndex, score); //  We add our new gained score to total score
            board = makeMove(board, targetIndex, index); //We make movement, and change board configuration.
            printBoard(board);
            System.out.printf("\nYour total score is %d.\n\n", score);// Print current board configuration
        }
        System.out.println("Thank you for playing this game.");
        System.out.printf("Possible maximum point with %d moves is %d." , movesCount, possibleMax);
    }
```

```
    /* This method calculate target index and return it. Target index is index that user want to move .
        It takes 2 parameters.
            move: Movement that user want to do
            index: current position of the star('*')
        It returns target index as an integer.
    */
    public static int calcTargetIndex(String move, int index){
        // This if-else tree checks movements that user wants
        if(move.equals("A")){ // User wants move left
            if(index % 4 != 0) // If our star is already at first column, we can't move left anymore.
                return index - 1; // If not, decrease index by 1
        }
        else if(move.equals("S")){ // User wants move right
            if(index % 4 != 3)  // If our star is already at last column, we can't move right anymore.
                return index + 1; // If not, increase index by 1
        }
        else if(move.equals("W")){ // User wants move up
            if(index > 3)  // If our star is already at first row, we can't move up anymore.
                return index - 4; // If not, decrease index by 4, so go up
        }
        else if(move.equals("Z")){ // User wants move down
            if(index < 12) // If our star is already at last row, we can't move down anymore.
                return index + 4; // If not, increase index by 4, so go down
        }

        return index;  // If user enter unvalid movement, do not anything. Just return same index.
    }

    /* This method swap our star with the character on target index. Target index is index that user want to move .
        It takes 3 parameters.
            oldBoard: Our current board configuration
            targetIndex: Index that we want to move our star('*') to
            index: current position of the star('*')
        It returns new board configuration as an string. */
    public static String makeMove(String oldBoard, int targetIndex, int index){
        String newBoard = ""; // newBoard: It will be store our new board configuration, at the beginning it is
empty.
        for(int i = 0; i < 16; i++){ //This for loop comes over every character on our board configuration.
            if(i == targetIndex) // When our i came target index, we add star(*) to new board instead of character
that was here on old board
                newBoard += "*";
            else if(i == index){ // When i came our star's index that on old board
                if(oldBoard.charAt(targetIndex) == 'G' || oldBoard.charAt(targetIndex) == 'P') // If the character on
this index is G or P, we add I to new board
                    newBoard += "I";
                else // If the character on this index is not G or P, we add the character on target index to new board
                    newBoard += oldBoard.charAt(targetIndex);
            }
            else  // If nothing special, we add the character on i to new board
                newBoard += oldBoard.charAt(i);
        }
        return newBoard; // Finally return new board configuration
    }

    /* This method calculate new total score and return it.
        It takes 3 parameters.
            board: Our current board configuration
            targetIndex: Index that we want to move our star('*') to
```

```
            score: Our current score
        It returns new total score as an integer.
    */
    public static int calcScore(String board, int targetIndex, int score){
        if(board.charAt(targetIndex) == 'G') // If the character on this target index is G, we add 5 to our score
            score += 5;
        else if(board.charAt(targetIndex) == 'P') // If the character on this target index is P, we add 1 to our score
            score += 1;
        return score; // Return new total score
    }

    /* This method prints the board. It doesn't return something.
        It takes 1 parameter.
            board: Our current board configuration
    */
    public static void printBoard(String board) {
        System.out.println("This is the board configuration now:");
        for(int i = 0; i < 4; i++){ // This for loop manages line number
            System.out.println(board.substring(i * 4, (i + 1) * 4)); // Print next 4 characters
        }
    }

    /* This method calculate possible maximum score and return it.
        It takes 2 parameters.
            board: Our current board configuration
            movesCount: number of moves that user wants
        It returns possible maximum score as an integer.
     */
    public static int calcMaxScore(String board, int movesCount ) {
        if (movesCount == 0)
            return 0;

        String tempBoard;
        int maxScore = 0;
        int index = board.indexOf('*');

        int targetIndex = calcTargetIndex("W", index);
        int upScore = calcScore(board, targetIndex, 0);
        tempBoard = makeMove(board, targetIndex, index);
        maxScore = Math.max(maxScore, upScore + calcMaxScore(tempBoard, movesCount - 1 ));

        targetIndex = calcTargetIndex("A", index);
        int rightScore = calcScore(board, targetIndex, 0);
        tempBoard = makeMove(board,targetIndex, index);
        maxScore = Math.max(maxScore, rightScore + calcMaxScore(tempBoard, movesCount - 1 ));

        targetIndex = calcTargetIndex("S", index);
        int leftScore = calcScore(board, targetIndex, 0);
        tempBoard = makeMove(board, targetIndex, index);
        maxScore = Math.max(maxScore, leftScore + calcMaxScore(tempBoard, movesCount - 1 ));

        targetIndex = calcTargetIndex("Z", index);
        int downScore = calcScore(board, targetIndex, 0);
        tempBoard = makeMove(board, targetIndex, index);
        maxScore = Math.max(maxScore, downScore + calcMaxScore(tempBoard, movesCount - 1 ));

        return maxScore;
    }
}
```

# *Output of The Program:*

```
Welcome to this weird game of P*G?
Do you want to use the default board configuration? Yes
This is the board configuration now:
ABGG
RTFP
PKVI
GVJ*

How many moves do you want to make? 2
Make a move and press enter. After each move, the board configuration and your total points will
be printed. Use A for Left, S for Right, W for Up, and Z for Down.

W
This is the board configuration now:
ABGG
RTFP
PKV*
GVJI

Your total score is 0.

W
This is the board configuration now:
ABGG
RTF*
PKVI
GVJI

Your total score is 1.

Thank you for playing this game.
Possible maximum point with 2 moves is 1.

How many moves do you want to make? 3
Make a move and press enter. After each move, the board configuration and your total points will
be printed. Use A for Left, S for Right, W for Up, and Z for Down.

Z
This is the board configuration now:
PPGG
IXXO
*OPP
KJHG

Your total score is 5.

A
This is the board configuration now:
PPGG
IXXO
*OPP
KJHG

Your total score is 5.

S
This is the board configuration now:
PPGG
IXXO
O*PP
KJHG

Your total score is 5.

Thank you for playing this game.
Possible maximum point with 3 moves is 7.
```

# *Conclusion:*

I think I solved problem with convient way for given tools. It was clean work. But if I can use global variables I can decrease number of loops. In this situation we can calculate one thing each method, because we can return one thing. But when we use global variables we can mixed up method.

For bonus part I use kind of recursion. I couldn't find any other way. I made some research about simulating recursion. But people use Stack class for simulating recursion. We didn't learn Stack too.