File - /Users/alexlatz/IdeaProjects/APCS-A/Unit2/ChooseYourOwnAdventure/src/main/java/GameManager.java

```java
package main.java;

import java.util.ArrayList;

public class GameManager {
    public char[][] maze;
    public int playerX, playerY;
    private char lastMove = ' ';
    public GameManager(int width, int height) {
        MazeGenerator mazeGenerator = new MazeGenerator(width, height);
        maze = mazeGenerator.getMaze();
        playerX = width*2-1;
        playerY = height*2-1;
        maze[playerX][playerY] = 'P';
        maze[playerX][playerY+1] = ' ';
        maze[1][1] = 'G';
        maze[1][0] = ' ';
    }
    public boolean moveUp() {
        if (!MazeGenerator.isWall(maze[playerX][playerY-1])) {
            maze[playerX][playerY--] = ' ';
            maze[playerX][playerY] = 'P';
            lastMove = 'u';
            return true;
        } else return false;
    }
    public boolean moveDown() {
        if (!MazeGenerator.isWall(maze[playerX][playerY+1])) {
            maze[playerX][playerY++] = ' ';
            maze[playerX][playerY] = 'P';
            lastMove = 'd';
            return true;
        } else return false;
    }
    public boolean moveLeft() {
        if (!MazeGenerator.isWall(maze[playerX-1][playerY])) {
            maze[playerX--][playerY] = ' ';
            maze[playerX][playerY] = 'P';
            lastMove = 'l';
            return true;
        } else return false;
    }
    public boolean moveRight() {
        if (!MazeGenerator.isWall(maze[playerX+1][playerY])) {
            maze[playerX++][playerY] = ' ';
            maze[playerX][playerY] = 'P';
            lastMove = 'r';
            return true;
        } else return false;
    }
    public boolean lastMove() {
        switch (lastMove) {
            case 'u':
                return moveUp();
            case 'd':
                return moveDown();
            case 'l':
                return moveLeft();
            case 'r':
                return moveRight();
        }
        return false;
    }
    public char[][] playerView() {
        char[][] view = new char[5][5];
        int newX=0, newY=0;
        for (int y = playerY-2; y <= playerY+2; y++) {
            for (int x = playerX-2; x <= playerX+2; x++) {
                if (x < 0 || x >= maze.length || y < 0 || y >= maze[0].length) view[newX++][newY] = ' ';
                else view[newX++][newY] = maze[x][y];
            }
            newY++;
            newX = 0;
        }
        return view;
    }
}

```

File - /Users/alexlatz/IdeaProjects/APCS-A/Unit2/ChooseYourOwnAdventure/src/main/java/MazeGenerator.java

```java
package main.java;

import java.util.ArrayList;

public class MazeGenerator {
    boolean[][] adjMatrix;
    boolean[] visited;
    int v;
    int width;
    char[][] maze;
    public MazeGenerator(int width, int height) {
        v = width * height;
        this.width = width;
        adjMatrix = new boolean[v][v];
        visited = new boolean[v];
        dfs(width*height-1);
        maze = maze();
        modifyMaze();
    }
    public boolean[][] getAdjMatrix() {
        return adjMatrix;
    }
    private char[][] maze() {
        //┌ ┬ ┐ ─
        //├ ┼ ┤
        //└ ┴ ┘
        char[][] maze = new char[width*2+1][(v/width)*2+1];
        maze[0][0] = '┌';
        maze[maze.length-1][0] = '┐';
        maze[0][maze[0].length-1] = '└';
        maze[maze.length-1][maze[0].length-1] = '┘';
        for (int i = 1; i < maze.length-1; i++) {
            maze[i][0] = '─';
            maze[i][maze[0].length-1] = '─';
        }
        for (int y = 1; y < maze[0].length-1; y++) {
            maze[0][y] = '|';
            maze[maze.length-1][y] = '|';
            if (y % 2 != 0) {
                for (int x = 1; x < maze.length - 1; x++) {
                    if (x % 2 != 0) maze[x][y] = ' ';
                    else {
                        int xVal = (x/2)-1;  //2->0, 4->1
                        int yVal = ((y-1)/2) * width;  //1->0, 3->1, 5->2
                        int xValT = x/2;  //2->1, 4->2
                        if (adjMatrix[yVal + xVal][yVal + xValT]) maze[x][y] = ' ';
                        else maze[x][y] = '|';
                    }
                }
            } else {
                for (int x = 1; x < maze.length - 1; x++) {
                    if (x % 2 != 0) {
                        int xVal = (x-1)/2;  //1->0, 3->1, 5->2
                        int yVal = ((y/2)-1) * width;  //2->0, 4->1
                        int yValT = (y/2) * width; //2->1, 4->2
                        if (adjMatrix[yVal + xVal][yValT + xVal]) maze[x][y] = ' ';
                        else maze[x][y] = '─';
                    } else maze[x][y] = '─';
                }
            }
        }
        return maze;
    }
    private void modifyMaze() {
        for (int i = 2; i < maze.length-1; i+=2) {
            if (maze[i][1] == '|') maze[i][0] = '┬';
            if (maze[i][maze[0].length-2] == '|') maze[i][maze[0].length-1] = '┴';
        }
        for (int i = 2; i < maze[0].length-1; i+=2) {
            if (maze[1][i] == '─') maze[0][i] = '├';
            if (maze[maze.length-2][i] == '─') maze[maze.length-1][i] = '┤';
        }
        for (int y = 1; y < maze[0].length-1; y++) {
            for (int x = 1; x < maze.length-1; x++) {
                if (isWall(maze[x][y])) {
                    if (isWall(maze[x][y-1])) {
                        checkUp(x, y);
                    } else if (isWall(maze[x][y+1])) {
                        checkDown(x, y);
                    } else maze[x][y] = '─';
                }
            }
        }
    }
    private void checkUp(int x, int y) {
        if (isWall(maze[x][y+1])) {
            if (isWall(maze[x-1][y])) {
```

File - /Users/alexlatz/IdeaProjects/APCS-A/Unit2/ChooseYourOwnAdventure/src/main/java/MazeGenerator.java

```
 88                 if (isWall(maze[x+1][y])) maze[x][y] = '┼';
 89                 else maze[x][y] = '┤';
 90             } else if (isWall(maze[x+1][y])) maze[x][y] = '├';
 91             else maze[x][y] = '|';
 92         } else if (isWall(maze[x-1][y])) {
 93             if (isWall(maze[x+1][y])) maze[x][y] = '┴';
 94             else maze[x][y] = '┘';
 95         } else if (isWall(maze[x+1][y])) maze[x][y] = '└';
 96         else maze[x][y] = '|';
 97     }
 98     private void checkDown(int x, int y) {
 99         if (isWall(maze[x - 1][y])) {
100             if (isWall(maze[x + 1][y])) maze[x][y] = '┬';
101             else maze[x][y] = '┐';
102         } else if (isWall(maze[x + 1][y])) maze[x][y] = '┌';
103         else maze[x][y] = '|';
104     }
105     public static boolean isWall(char c) {
106         return c == '-' || c == '|' || c == '┌' || c == '┬' || c == '┐' || c == '├' ||
107                 c == '┼' || c == '┤' || c == '└' || c == '┴' || c == '┘';
108     }
109     public char[][] getMaze() {
110         return maze;
111     }
112     private void dfs(int n) {
113         visited[n] = true;
114         int i = neighbor(n);
115         while (i != -1) {
116             adjMatrix[i][n] = true;
117             adjMatrix[n][i] = true;
118             dfs(i);
119             i = neighbor(n);
120         }
121     }
122     private int neighbor(int n) {
123         ArrayList<Integer> neighbors = new ArrayList<>();
124         if (n >= width && !visited[n-width]) neighbors.add(n-width);
125         if (n % width != 0 && !visited[n-1]) neighbors.add(n-1);
126         if (n % width != width-1 && !visited[n+1]) neighbors.add(n+1);
127         if (n < v - width && !visited[n+width]) neighbors.add(n+width);
128         if (neighbors.size() == 0) return -1;
129         return neighbors.get((int)(Math.random() * (double)neighbors.size()));
130     }
131
132     public static void main(String[] args) {
133         MazeGenerator mazeGenerator = new MazeGenerator(10, 10);
134         char[][] maze = mazeGenerator.getMaze();
135         for (int y = 0; y < maze[0].length; y++) {
136             for (int x = 0; x < maze.length; x++) {
137                 System.out.print(maze[x][y]);
138             }
139             System.out.println();
140         }
141     }
142 }
143
```

File - /Users/alexlatz/IdeaProjects/APCS-A/Unit2/ChooseYourOwnAdventure/src/main/java/PlayerInterface.java

```java
package main.java;

import java.util.Scanner;

public class PlayerInterface {
    static Scanner scanner = new Scanner(System.in);
    GameManager gameManager;
    public PlayerInterface(int width, int height) {
        gameManager = new GameManager(width, height);
    }
    public void spacer() {
        System.out.println("----------------------");
        System.out.println();
    }
    private void handleMove(boolean result, String direction, String command) {
        String[] move = command.split(" ");
        if (move.length > 1) {
            boolean hitWall = true;
            if (move[1].toUpperCase().equals("WALL")) {
                while (hitWall) hitWall = gameManager.lastMove();
            }
            else {
                try {
                    if (Integer.parseInt(move[1]) > 1) {
                        for (int i = 0; i < Integer.parseInt(move[1]); i++) {
                            if (hitWall) hitWall = gameManager.lastMove();
                            else break;
                        }
                    }
                } catch (NumberFormatException e) {
                    System.out.println("Please add a valid command after the direction.");
                }
            }
        }
        if (!result) System.out.println("You run into the wall of the maze, hitting your head.");
        else System.out.println("You move " + direction + " in the maze.");
    }
    public void gameLoop() {
        boolean gameRunning = true;
        while(gameRunning) {
            System.out.println("VIEW OF MAZE");
            spacer();
            char[][] view = gameManager.playerView();
            for (int y = 0; y < view[0].length; y++) {
                for (int x = 0; x < view.length*2; x++) {
                    if (x % 2 == 0) {
                        System.out.print(view[x/2][y]);
                    }
                    else System.out.print(" ");
                }
                System.out.println();
            }
            spacer();
            System.out.println("What would you like to do?");
            System.out.println("Enter \"UP\" to move upwards.");
            System.out.println("Enter \"DOWN\" to move downwards.");
            System.out.println("Enter \"LEFT\" to move to the left.");
            System.out.println("Enter \"RIGHT\" to move to the right.");
            System.out.println("Enter \"QUIT\" to quit the maze.");
            System.out.println("To move multiple spaces at once, add a number after the directional command
or add \"WALL\" to move to the next wall.");
            String command = scanner.nextLine();
            switch (command.toUpperCase().split(" ")[0]) {
                case "UP":
                    handleMove(gameManager.moveUp(), "upwards", command);
                    break;
                case "DOWN":
                    handleMove(gameManager.moveDown(), "downwards", command);
                    break;
                case "LEFT":
                    handleMove(gameManager.moveLeft(), "to the left", command);
                    break;
                case "RIGHT":
                    handleMove(gameManager.moveRight(), "to the right", command);
                    break;
                case "QUIT":
                    gameRunning = false;
                    break;
                default:
                    System.out.println("Unknown command. Please try again.");
            }
            if (gameManager.playerX == 1 && gameManager.playerY == 1) {
                System.out.println("You have found the exit! Congratulations on escaping the maze.");
                gameRunning = false;
            }
            spacer();
        }
```

File - /Users/alexlatz/IdeaProjects/APCS-A/Unit2/ChooseYourOwnAdventure/src/main/java/PlayerInterface.java

```
87      }
88
89      public static void main(String[] args) {
90          System.out.print("What width would you like the maze to be? ");
91          int width = Integer.parseInt(scanner.nextLine());
92          System.out.print("What height would you like the maze to be? ");
93          int height = Integer.parseInt(scanner.nextLine());
94          PlayerInterface playerInterface = new PlayerInterface(width, height);
95          System.out.println("You are the player, P. Your goal is to exit the maze. You can only see a 5x5
    portion of the maze at a time.");
96          playerInterface.gameLoop();
97      }
98  }
99
```