PLAYER 1

HIGHSCORE 2500

PLAYER 2

MINESWEEPER

START

MENU

SIGN IN

MOVE CAREFULLY !!!

MENU 01 07 12

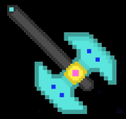
CONTENTS

TOPICS COVERED

INTRODUCTION

CODE

PLAYING DEMO GAME



Q&A

SIGN IN

BACK TO AGENDA PAGE

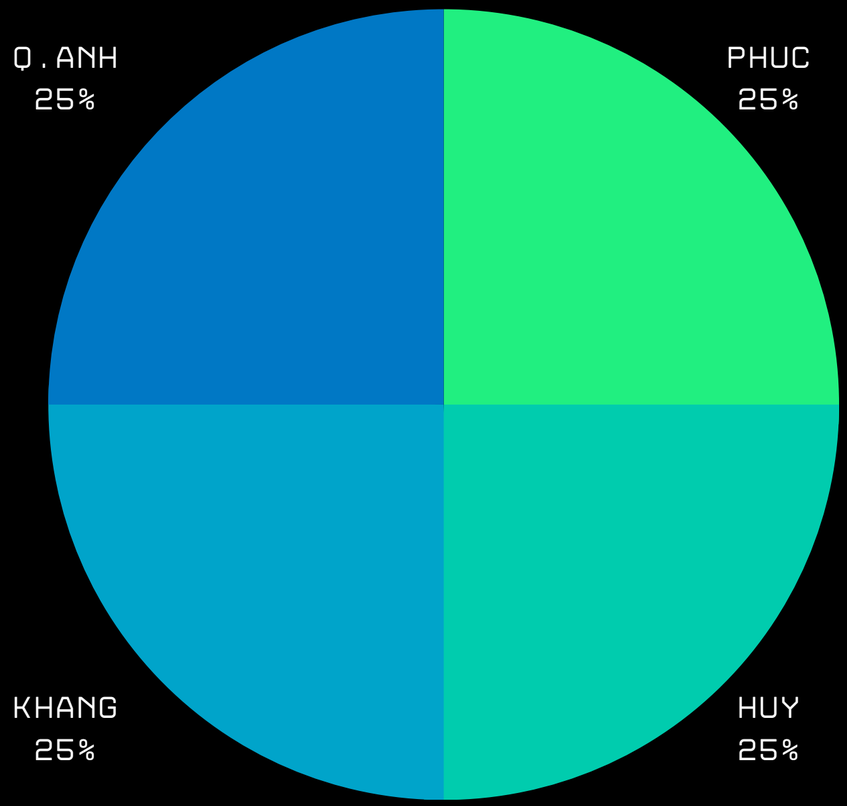
1.INTRODUCTION

HIGHSCORE 2500

PLAYER 2

TEAM MEMBER & CONTRIBUTION PERCENTAGE

Nguyễn Duy Phúc_ITDSIU21030
Nguyễn Quốc Huy_ITDSIU21067
Vũ Vỹ Khang_ ITDSIU21090
Phan Quốc Anh_ITDSIU21001



BACK TO AGENDA PAGE

ABOUT THIS PROJECT

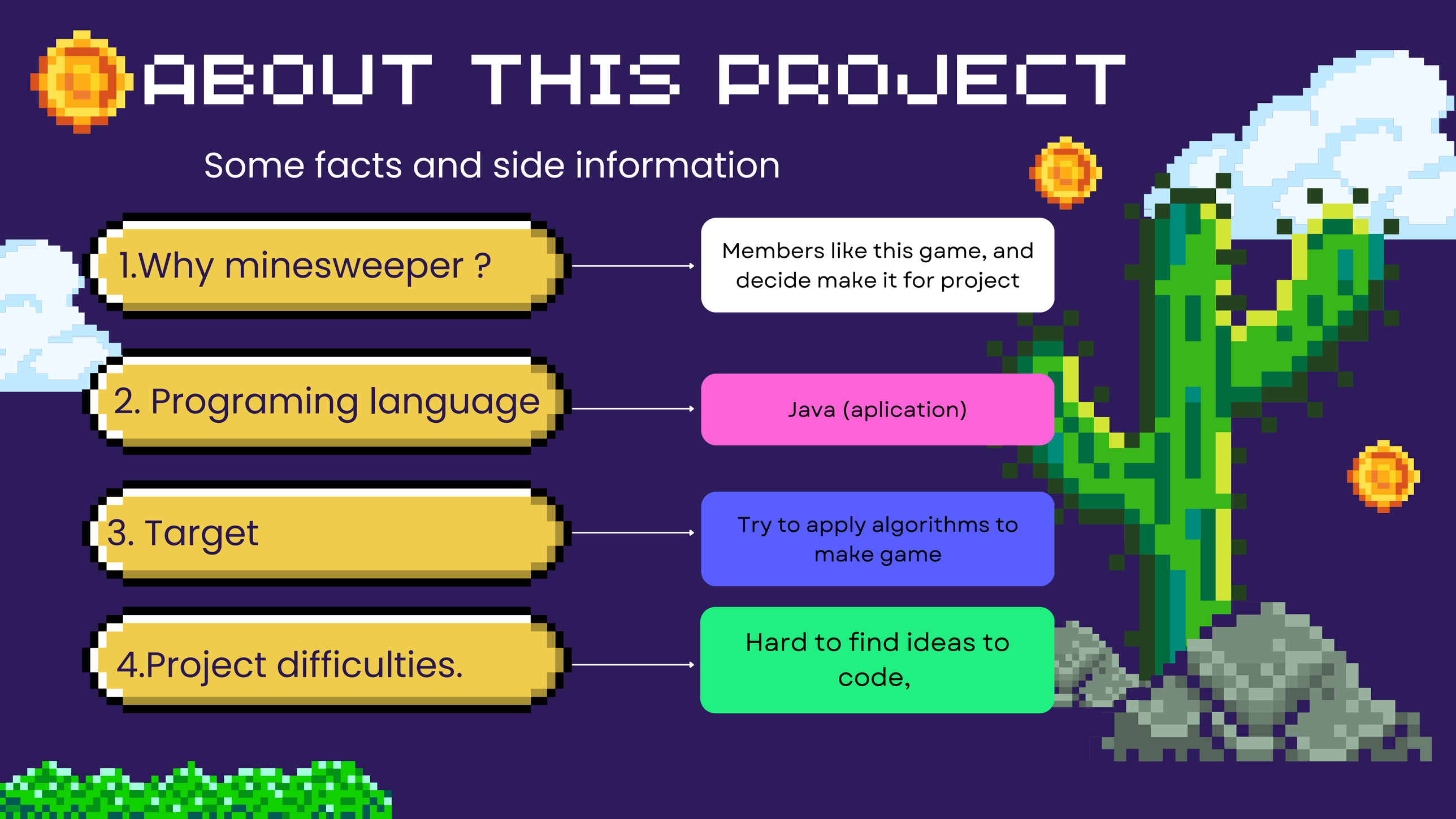
Some facts and side information

1.Why minesweeper ? → Members like this game, and decide make it for project

2. Programing language → Java (aplication)

3. Target → Try to apply algorithms to make game

4.Project difficulties. → Hard to find ideas to code,



RULE

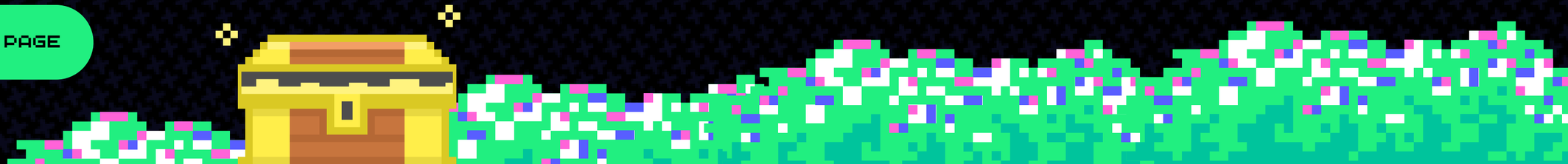
Clear the minefield without detonating hidden mines. Click squares to uncover them. Avoid mines. Use numbers as clues. Mark suspected mines. Win by uncovering all safe squares.

2.CODE

SOME CODE AND EXPLAIN



BACK TO AGENDA PAGE



BACK TO AGENDA PAGE

2.CODE
CREATE MAP

```java
public class Board extends JPanel implements ActionListener {
    //DEFINE FIELDS
        //Fields related to board and cells
        private final int CELL_SIZE = 15;
        private final int N_MINES = 10;
        private static int N_ROWS = 16;
        private static int N_COLS = 16;
        private static String CELL_SPLITTER = " - ";
        private static String OBJECT_SPLITTER = "$";
        private final int BOARD_WIDTH = N_ROWS * CELL_SIZE + 1;
        private final int BOARD_HEIGHT = N_COLS * CELL_SIZE + 1;
        private int minesLeft;//keeps track of how many mines are left based on what user has flagged
        //2D array to represent game board
        protected static Cell[][] gameBoard;
        //total number of cells
        private int allCells;
        //Fields related to images used in our game to represent cells and bombs
        private final int NUM_IMAGES = 13;
        //Using map as collection to store images and their names, which can make it more easily retrievable
        private java.util.Map<String, Image> images;
        //Fields related to game status
        private boolean inGame;
        private static JLabel statusbar;
        private static JButton bUndo;
        private static JButton bRule;
        private static JTextArea textArea;
        private static String STATUS_FILE = "Status.txt";
        private Stack gameSteps = new Stack();
```

BACK TO AGENDA PAGE

SHOW IMAGE ON BOARD

2.CODE

```
private void initBoard() throws IOException {

    setPreferredSize(new Dimension(BOARD_WIDTH, BOARD_HEIGHT));
    images = new java.util.HashMap<>();


  //Put all relevant images in the map, some images named with integers, others named with descriptors
    for (int i = 1; i < 9; i++) {
        String path = "D:/Game/Project/src/resources/" + i + ".png";
        images.put(Integer.toString(i), (new ImageIcon(path)).getImage());
    }
    images.put(key:"Bomb", (new ImageIcon(filename:"D:/Game/Project/src/resources/Bomb.png")).getImage());
    images.put(key:"Covered", (new ImageIcon(filename:"D:/Game/Project/src/resources/Covered.png")).getImage());
    images.put(key:"Empty", (new ImageIcon(filename:"D:/Game/Project/src/resources/Empty.png")).getImage());
    images.put(key:"Marked", (new ImageIcon(filename:"D:/Game/Project/src/resources/Marked.png")).getImage());
    images.put(key:"Wrongmarked", (new ImageIcon(filename:"D:/Game/Project/src/resources/Wrongmarked.png")).getImage());

    addMouseListener(new MinesAdapter());

    showRules();
```

BACK TO AGENDA PAGE

2.CODE

SPAWN BOM

```
//set up the grid
while (i < N_MINES) {
    Random random = new Random();
    int positionX = (int) (random.nextInt(15 - 0 + 1) + 0);
    int positionY = (int) (random.nextInt(15 - 0 + 1) + 0);

    //randomly place the bomb cell
    if(gameBoard[positionX][positionY].getCellType() != CellType.Bomb) {
        gameBoard[positionX][positionY] = new BombCell();


        //sets up neighbor cells
        for(int dx = -1; dx <= 1; dx++) {
            for(int dy = -1; dy <= 1; dy++) {
                if((dx != 0 || dy != 0) && positionX + dx < N_COLS && positionY + dy < N_ROWS
                        && positionX + dx >= 0 && positionY + dy >=0) {
                    CellType typeOfCell = gameBoard[positionX + dx][positionY + dy].getCellType();
                    if(typeOfCell != CellType.Bomb) {//not already a neighbor cell
                                if (typeOfCell != CellType.BombNeighbor) {
                                    NeighborOfBombCell neighbor = new NeighborOfBombCell();
                                       neighbor.cellCount();
                                       gameBoard[positionX + dx][positionY + dy] = neighbor;
                                }
                                else {//already a neighbor cell, just need to update the neighbor count

                                gameBoard[positionX + dx][positionY + dy].cellCount();
                    }
                    }
```

for loop to random bomb

2.CODE

CHECK NEIGHBOR AREA

```
//checks this for all neighbors
  public void find_empty_cells(int x, int y) {

    //int current_col = j % N_COLS;
    gameBoard[x][y].flipUp();
    gameSteps.push(x * N_COLS + y);//add steps to gameSteps Stack
    for(int dx = -1; dx <= 1; dx++) {
        for(int dy = -1; dy <= 1; dy++) {//set bounds
            if((dx != 0 || dy != 0) && x + dx < N_COLS && y + dy < N_ROWS
                    && x + dx >= 0 && y + dy >= 0) {

                CellType typeOfCell = gameBoard[x + dx][y + dy].getCellType();
                    //if(typeOfCell == CellType.BombNeighbor && gameBoard[x + dx][y + dy].isCoveredCell()) {
                    //    gameBoard[x + dx][y + dy].flipUp();
                    //}
                    //else
                    if(typeOfCell == CellType.Empty && gameBoard[x + dx][y + dy].isCoveredCell()) {
                    find_empty_cells(x + dx, y + dy);
                }
            }
            }
    }



    }
```

Using recursion

BACK TO AGENDA PAGE

BACK TO AGENDA PAGE

2.CODE

UNDO

```
private void undo() {
    if (!gameSteps.empty()) {
        int i = (Integer)gameSteps.pop();//gets most recent game step
        //corresponding cell to the game step
        Cell cell = gameBoard[i / N_COLS][i % N_ROWS];

        //Handle flagged cells situation, which are covered
        if (cell.isCoveredCell()) {
            cell.changeWhetherMarked();
            if (cell.isMarkedCell()) {
                minesLeft--;
            } else {
                minesLeft++;
                if (!inGame) {
                    inGame = true;
                }
            }
        }

        else if (cell.getCellType() == CellType.Bomb) {
            cell.isCovered = true;
            inGame = true;

        }

    else if (cell.getCellType() == CellType.BombNeighbor) {
            cell.isCovered = true;
        }

        String msg = Integer.toString(minesLeft);
        this.statusbar.setText("Flags Left: " + msg);
```

ADDITIONAL FEATURE

SAVE GAME

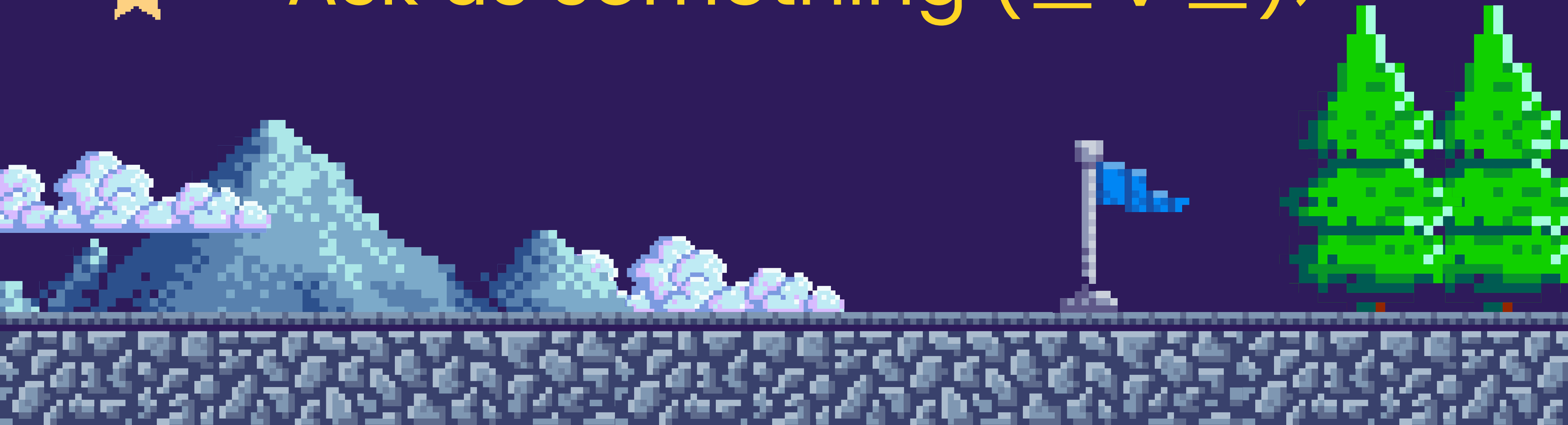
```java
//Saving game status
protected static void saveGameStatus2File() throws IOException {
    String userName = "";
    //lets user know that
    if ("".equals(textArea.getText()) || textArea.getText().equals(anObject:"Input your name here...")) {
        JOptionPane.showMessageDialog(parentComponent:null, message:"We gave you a default user name, you may input your name next time.");
        userName = "Default user";
    } else {
        userName = textArea.getText();
    }


    if (gameBoard.length == 0) {
        System.exit(status:0);
    }


    //Writes user name
    //Goes through the entire game board, records the state of each cell, writes in a text file
    FileWriter writer = new FileWriter(STATUS_FILE, append:false);
```

3.START THE GAMES

START

4. Q&A

Ask us something (≧▽≦)ﾉ




THANK YOU FOR WATCHING
Hope to see you again, but not in this course <3