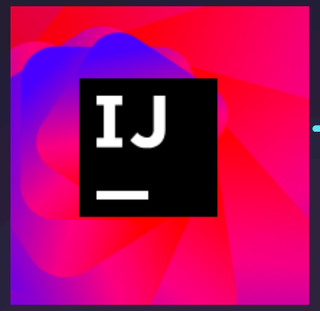
DSA Project

MINESWEEPER

CONTRIBUTOR

NGUYỄN ĐỨC NGUYÊN PHÚC
ITDSIU21108

Tools

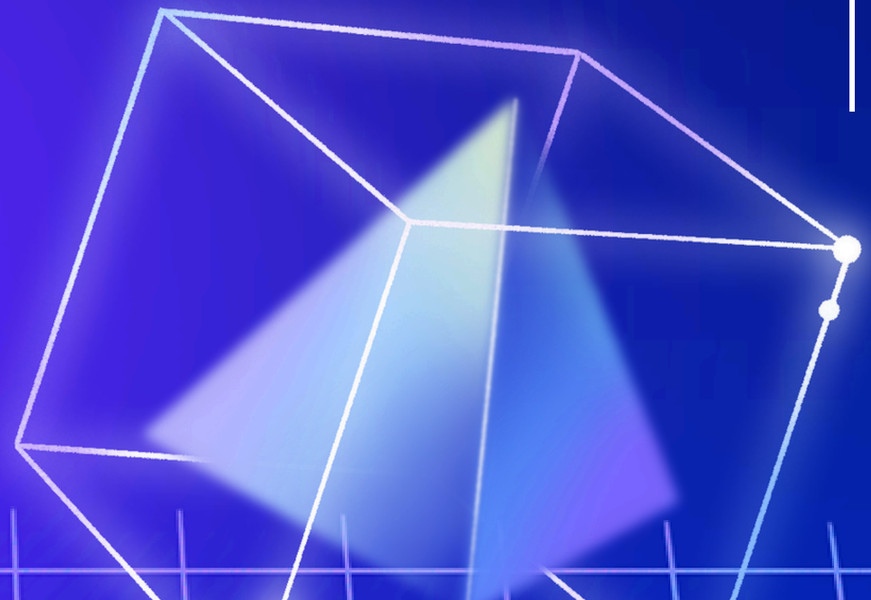


IntelliJ IDEA

GITHUB

ECLIPSE IDE

TABLE OF CONTENTS

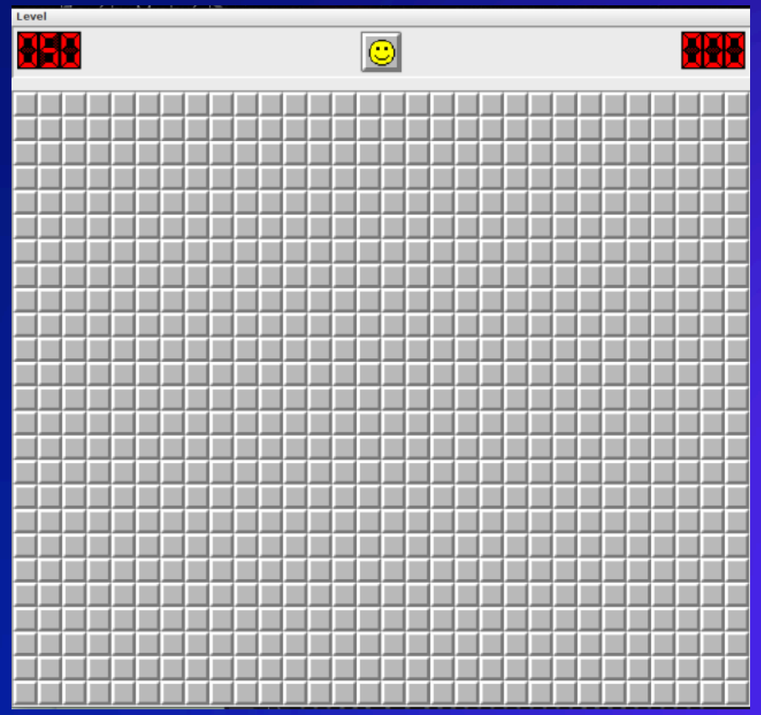


INTRODUCTION

CODED BY JAVA

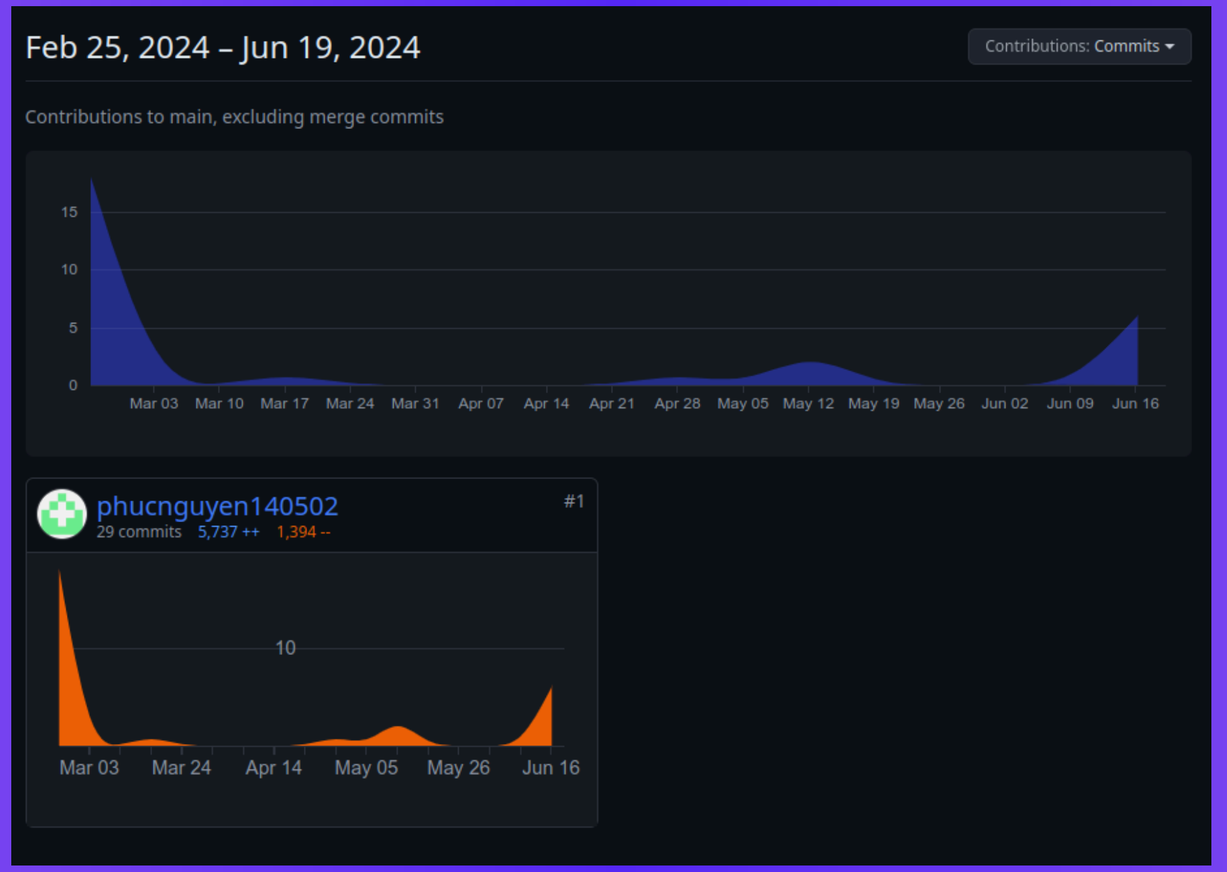
- A game of logic.
- Make use of your intelligence to locate the mines.

GAME RULES

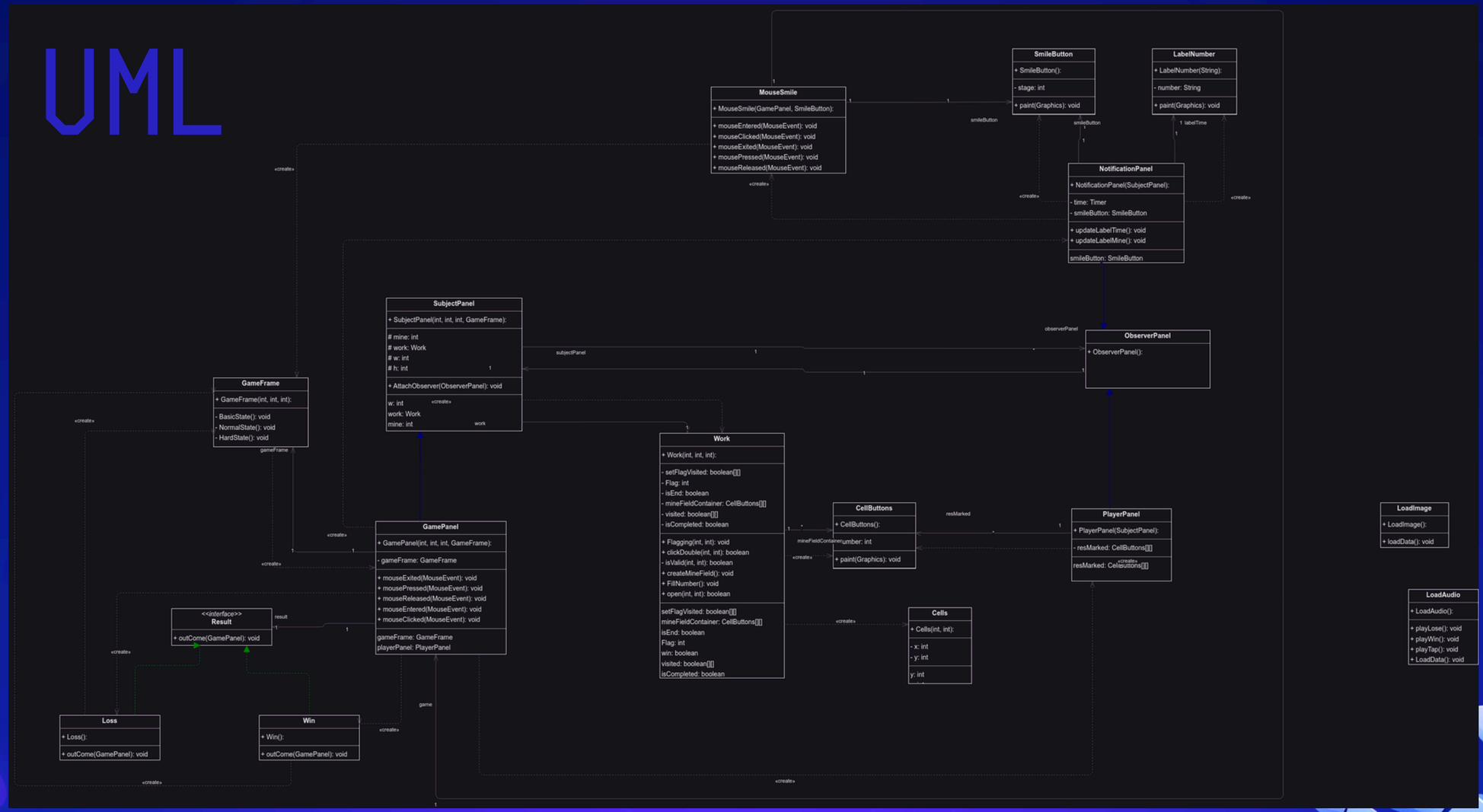


- Right Click to reveal a cell
- Left Click to place a flagging or unflagging
- Click on the bomb -> LOSE !
- 3 Levels: Beginner, Intermediate and Hard

GIT CONTRIBUTION



UML


MouseSmile
+ MouseSmile(GamePanel, SmileButton):
+ mouseEntered(MouseEvent): void
+ mouseClicked(MouseEvent): void
+ mouseExited(MouseEvent): void
+ mousePressed(MouseEvent): void
+ mouseReleased(MouseEvent): void
SmileButton
+ SmileButton():
- stage: int
+ paint(Graphics): void
LabelNumber
+ LabelNumber(String):
- number: String
+ paint(Graphics): void
NotificationPanel
+ NotificationPanel(SubjectPanel):
- time: Timer
- smileButton: SmileButton
+ updateLabelTime(): void
+ updateLabelMine(): void
smileButton: SmileButton
SubjectPanel
+ SubjectPanel(int, int, int, GameFrame):
# mine: int
# work: Work
# w: int
# h: int
+ AttachObserver(ObserverPanel): void
w: int
work: Work
mine: int
ObserverPanel
+ ObserverPanel():
GameFrame
+ GameFrame(int, int, int):
- BasicState(): void
- NormalState(): void
- HardState(): void
Work
+ Work(int, int, int):
- setFlagVisited: boolean[][]
- Flag: int
- isEnd: boolean
- mineFieldContainer: CellButtons[][]
- visited: boolean[][]
- isCompleted: boolean
+ Flagging(int, int): void
+ clickDouble(int, int): boolean
- isValid(int, int): boolean
+ createMineField(): void
+ FillNumber(): void
+ open(int, int): boolean
setFlagVisited: boolean[][]
mineFieldContainer: CellButtons[][]
isEnd: boolean
Flag: int
win: boolean
visited: boolean[][]
isCompleted: boolean
CellButtons
+ CellButtons():
number: int
+ paint(Graphics): void
PlayerPanel
+ PlayerPanel(SubjectPanel):
- resMarked: CellButtons[][]
resMarked: CellButtons[][]
LoadImage
+ LoadImage():
+ loadData(): void
GamePanel
+ GamePanel(int, int, int, GameFrame):
- gameFrame: GameFrame
+ mouseExited(MouseEvent): void
+ mousePressed(MouseEvent): void
+ mouseReleased(MouseEvent): void
+ mouseEntered(MouseEvent): void
+ mouseClicked(MouseEvent): void
gameFrame: GameFrame
playerPanel: PlayerPanel
<<interface>>
Result
+ outCome(GamePanel): void
Cells
+ Cells(int, int):
- x: int
- y: int
y: int
LoadAudio
+ LoadAudio():
+ playLose(): void
+ playWin(): void
+ playTap(): void
+ LoadData(): void
Loss
+ Loss():
+ outCome(GamePanel): void
Win
+ Win():
+ outCome(GamePanel): void
«create»
smileButton
labelTime
observerPanel
subjectPanel
work
gameFrame
result
resMarked
mineFieldContainer
game

DATA STRUCTURE & ALGORITHM

Applying Advanced Graph (BFS)

- The algorithm in class work
- click work.java to see

Time complexity : O(N^2)

DATA STRUCTURE & ALGORITHM

Applying Recursion

```
// Apply BFS Algorithm and matrix to open when click to cell "0" using recursion
public boolean open(int i, int j) {  3 usages  phucnguyen140502 *
    if (!isCompleted && !isEnd) {
        if (visited[i][j]) {
            if (mineField[i][j] == 0) { // open around until the cells have mines around
                visited[i][j] = false;
                mineFieldContainer[i][j].setNumber(0);
                mineFieldContainer[i][j].repaint();
                if (isWin()) {
                    isEnd = true;
                    return false;
                }
                for (int k = 0; k < 8; k++) {
                    int x = i + drow[k];
                    int y = j + dcol[k];

                    if (isValid(x, y) && visited[x][y]) {
                        open(x, y);
                    }
```

Time Complexity : O(N^2)

DATA STRUCTURE & ALGORITHM

Applying HashTable

```
public class LoadAudio {  11 usages  phucnguyen140502
    public static HashTable<String, Clip> listAudio;  7 usages

    Explain | Test | Document | Fix | Ask
    public static void LoadData() {  4 usages  phucnguyen140502
```

```
    Explain | Test | Document | Fix | Ask
    public static void loadData() {  3 usages  phucnguyen140502
        listImage = new HashTable<>( capacity: 35);
```

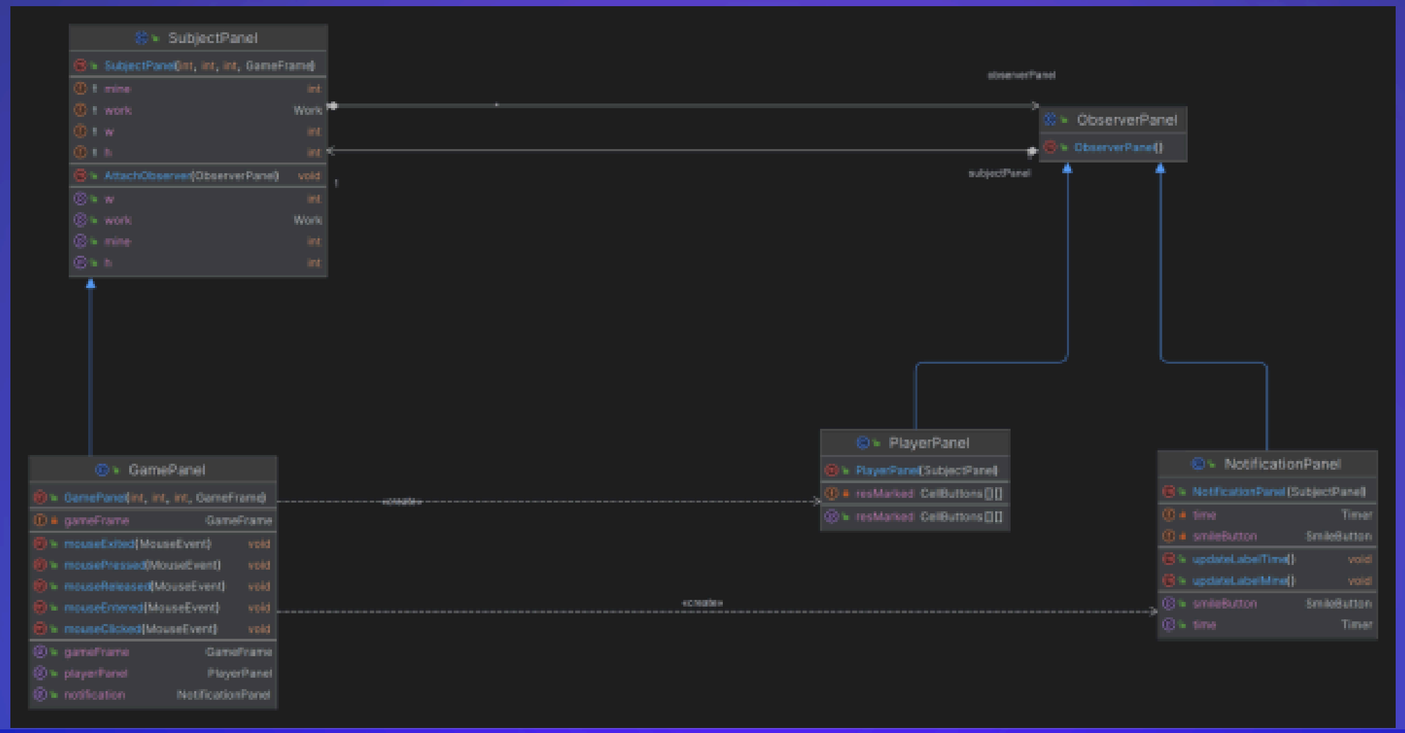
Time Complexity : O(1)

Create the board

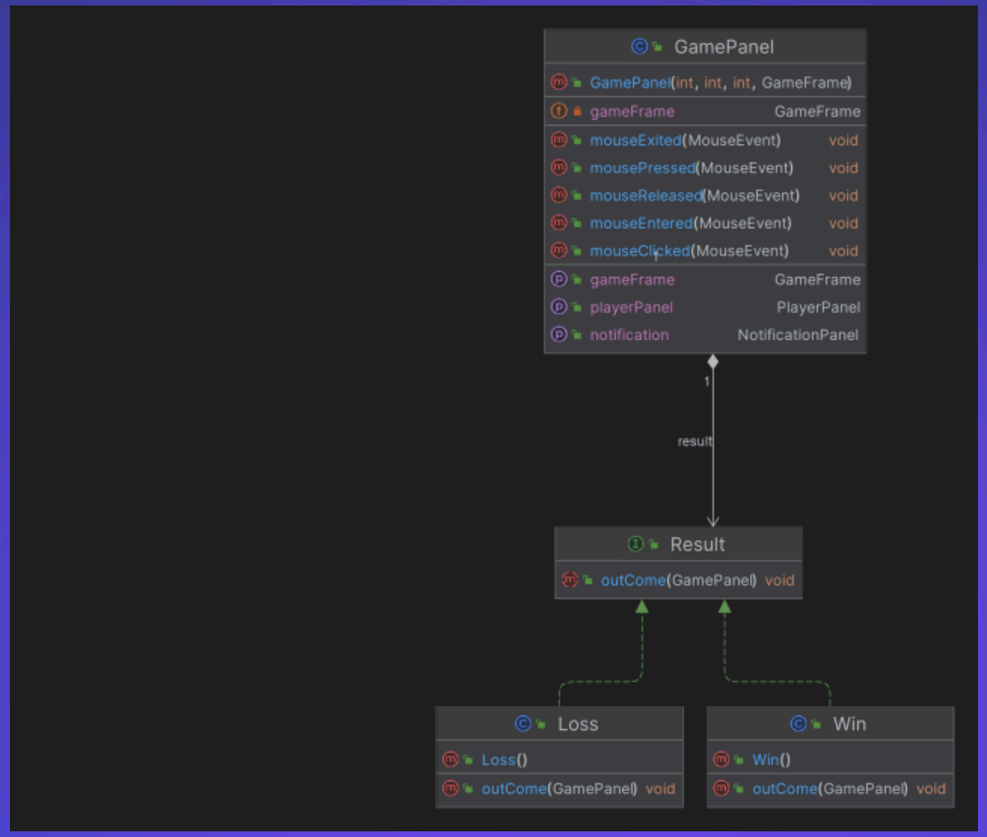
```
    }
}

Explain | Test | Document | Fix | Ask
public void createMineField(){  1 usage  phucnguyen140502
    int count;
    do{
        int locationX = random.nextInt(w);
        int locationY = random.nextInt(h);
        if (mineField[locationX][locationY] == -1) {
            break;
        }
        mineField[locationX][locationY] = -1;
        visited[locationX][locationY] = true;

        count = 0;
        for (int i = 0; i < w; i++) {
            for (int j = 0; j < h; j++) {
                if (mineField[i][j] == -1) {
                    count++;

                }
            }
        }

    }while (count != mine);



}
```

Time Complexity : O(w*h)

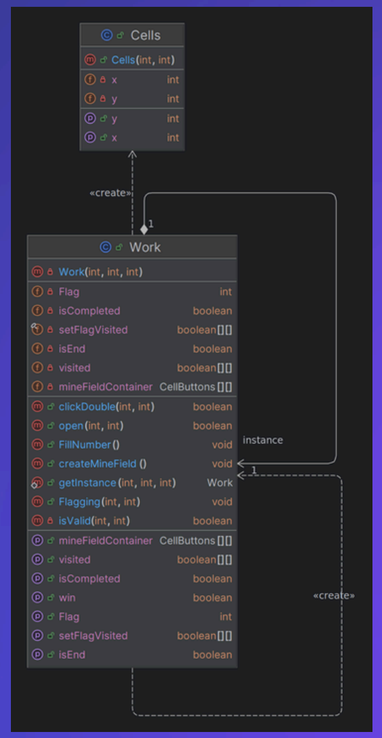
Observer Design Pattern

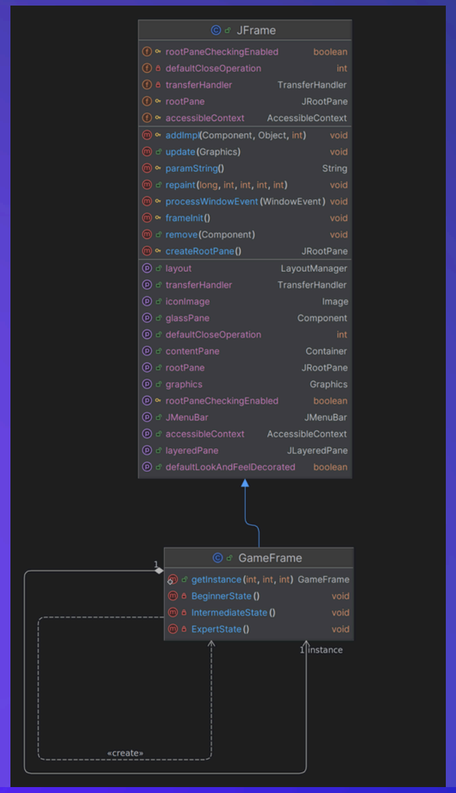


Strategy Design Pattern



Singleton Pattern





GAME DEMO


Thank you
for
listening

RESOURCE PAGE