Work on project. Stage 5/5: Battle!

Project: Minesweeper

Hard 1 hour

436 users solved this problem. Latest completion was **30 minutes ago**.

Description

In this stage, you will upgrade your program to act just like the original Minesweeper game! We won't show all the hints from the beginning anymore, but we will allow the player to explore the minefield by themselves, which is much more challenging and fun.

The game starts with an unexplored minefield that has a user-defined number of mines.

The player can:

- Mark unexplored cells as cells that potentially have a mine, and also remove those marks. Any empty cell can be marked, not just the cells that contain a mine. The mark is removed by marking the previously marked cell.
- Explore a cell if they think it does not contain a mine.

There are three possibilities after exploring a cell:

1. If the cell is empty and has no mines around, all the cells around it, including the marked ones, can be explored, and it should be done automatically. Also, if next to the explored cell there is another empty one with no mines around, all the cells around it should be explored as well, and so on, until no more can be explored automatically.
2. If a cell is empty and has mines around it, only that cell is explored, revealing a number of mines around it.
3. If the explored cell contains a mine, the game ends and the player loses.

There are two possible ways to win:

1. Marking all the cells that have mines correctly.
2. Opening all the safe cells so that only those with unexplored mines are left.

11 / 11 Prerequisites

- Stack
- Queue
- Deque
- Interface
- Inheritance

Show all

Join a study group for the project Minesweeper

Discuss your current project with fellow learners and help each other.

Objectives

In this final stage, your program should contain the following additional functionality:

1. Print the current state of the minefield starting with all unexplored cells at the beginning, ask the player for their next move with the message `Set/unset mine marks or claim a cell as free:`, treat the player's move according to the rules, and print the new minefield state. Ask for the player's next move until the player wins or steps on a mine. The player's input contains a pair of cell coordinates and a command: `mine` to mark or unmark a cell, `free` to explore a cell.
2. If the player explores a mine, print the field in its current state, with mines shown as `X` symbols. After that, output the message `You stepped on a mine and failed!`.
3. Generate mines like in the original game: the first cell explored with the `free` command cannot be a mine; it should always be empty. You can achieve this in many ways – it's up to you.

Use the following symbols to represent each cell's state:

- `.` as unexplored cells
- `/` as explored free cells without mines around it
- Numbers from 1 to 8 as explored free cells with 1 to 8 mines around them, respectively
- `X` as mines
- `*` as unexplored marked cells

Examples

The greater-than symbol followed by a space (`> `) represents the user input. Note that it's not part of the input.

Example 1: *the user loses after exploring a cell that contains a mine*

```
How many mines do you want on the field? > 10

 |123456789|
-|---------|
1|.........|
2|.........|
3|.........|
4|.........|
5|.........|
6|.........|
7|.........|
8|.........|
9|.........|
-|---------|
Set/unset mines marks or claim a cell as free: > 3 2 free

 |123456789|
-|---------|
1|.1///1...|
2|.1//12...|
3|11//1....|
4|////1....|
5|11111....|
6|.........|
7|.........|
8|.........|
9|.........|
-|---------|
Set/unset mines marks or claim a cell as free: > 1 1 free

 |123456789|
-|---------|
1|11///1...|
2|.1//12...|
3|11//1....|
4|////1....|
5|11111....|
6|.........|
7|.........|
8|.........|
9|.........|
-|---------|
Set/unset mines marks or claim a cell as free: > 1 2 mine

 |123456789|
-|---------|
1|11///1...|
2|*1//12...|
3|11//1....|
4|////1....|
5|11111....|
6|.........|
7|.........|
8|.........|
9|.........|
-|---------|
Set/unset mines marks or claim a cell as free: > 8 8 free

 |123456789|
-|---------|
1|11///1...|
2|*1//12...|
3|11//1....|
4|////1....|
5|11111....|
6|.........|
7|.........|
8|.......1.|
9|.........|
-|---------|
Set/unset mines marks or claim a cell as free: > 7 8 free

 |123456789|
-|---------|
1|11///1...|
2|*1//12...|
3|11//1....|
```

```
4|////1....|
5|11111....|
6|.........|
7|.........|
8|......11.|
9|.........|
-|---------|
Set/unset mines marks or claim a cell as free: > 6 8 free

 |123456789|
-|---------|
1|11///1...|
2|*1//12...|
3|11//1....|
4|////1....|
5|11111....|
6|.........|
7|.........|
8|.....211.|
9|.........|
-|---------|
Set/unset mines marks or claim a cell as free: > 2 7 free

 |123456789|
-|---------|
1|11///1...|
2|*1//12...|
3|11//1....|
4|////1....|
5|11111....|
6|.........|
7|.3.......|
8|.....211.|
9|.........|
-|---------|
Set/unset mines marks or claim a cell as free: > 5 6 free

 |123456789|
-|---------|
1|11///1X..|
2|X1//12...|
3|11//1X...|
4|////1....|
5|11111....|
6|.X..X....|
7|.3X...X..|
8|.X..X211.|
9|...X.....|
-|---------|
You stepped on a mine and failed!
```

Example 2: *the user wins by marking all mines correctly*

```
How many mines do you want on the field? > 8

 |123456789|
-|---------|
1|.........|
2|.........|
3|.........|
4|.........|
5|.........|
6|.........|
7|.........|
8|.........|
9|.........|
-|---------|
Set/unset mines marks or claim a cell as free: > 5 5 free

 |123456789|
-|---------|
1|..1//1...|
2|111//2...|
3|/////1...|
4|/////11..|
5|//////1..|
6|/111//1..|
7|23.1//111|
8|..21/////|
9|..1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 2 1 mine

 |123456789|
-|---------|
1|.*1//1...|
2|111//2...|
3|/////1...|
4|/////11..|
5|//////1..|
6|/111//1..|
7|23.1//111|
8|..21/////|
9|..1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 3 7 mine

 |123456789|
-|---------|
1|.*1//1...|
2|111//2...|
3|/////1...|
4|/////11..|
5|//////1..|
6|/111//1..|
7|23*1//111|
8|..21/////|
9|..1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 2 8 mine

 |123456789|
-|---------|
1|.*1//1...|
2|111//2...|
3|/////1...|
4|/////11..|
5|//////1..|
6|/111//1..|
7|23*1//111|
8|.*21/////|
9|..1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 1 8 mine

 |123456789|
-|---------|
1|.*1//1...|
2|111//2...|
3|/////1...|
```

```
4|/////11..|
5|//////1..|
6|/111//1..|
7|23*1//111|
8|**21/////|
9|..1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 7 3 mine

 |123456789|
-|---------|
1|.*1//1...|
2|111//2...|
3|/////1*..|
4|/////11..|
5|//////1..|
6|/111//1..|
7|23*1//111|
8|**21/////|
9|..1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 8 3 free

 |123456789|
-|---------|
1|.*1//1...|
2|111//2...|
3|/////1*1.|
4|/////11..|
5|//////1..|
6|/111//1..|
7|23*1//111|
8|**21/////|
9|..1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 9 3 free

 |123456789|
-|---------|
1|.*1//1...|
2|111//2.31|
3|/////1*1/|
4|/////111/|
5|//////111|
6|/111//1..|
7|23*1//111|
8|**21/////|
9|..1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 8 6 mine

 |123456789|
-|---------|
1|.*1//1...|
2|111//2.31|
3|/////1*1/|
4|/////111/|
5|//////111|
6|/111//1*.|
7|23*1//111|
8|**21/////|
9|..1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 7 2 free

 |123456789|
-|---------|
1|.*1//1...|
2|111//2231|
3|/////1*1/|
4|/////111/|
5|//////111|
6|/111//1*.|
7|23*1//111|
8|**21/////|
9|..1//////|
-|---------|
```

```
Set/unset mines marks or claim a cell as free: > 7 1 mine

 |123456789|
-|---------|
1|.*1//1*..|
2|111//2231|
3|/////1*1/|
4|/////111/|
5|//////111|
6|/111//1*.|
7|23*1//111|
8|**21/////|
9|..1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 9 1 mine

 |123456789|
-|---------|
1|.*1//1*.*|
2|111//2231|
3|/////1*1/|
4|/////111/|
5|//////111|
6|/111//1*.|
7|23*1//111|
8|**21/////|
9|..1//////|
-|---------|
Congratulations! You found all the mines!
```

Example 3: *the user wins by exploring all safe cells*

```
How many mines do you want on the field? > 5

 |123456789|
-|---------|
1|.........|
2|.........|
3|.........|
4|.........|
5|.........|
6|.........|
7|.........|
8|.........|
9|.........|
-|---------|
Set/unset mines marks or claim a cell as free: > 5 5 free

 |123456789|
-|---------|
1|/////////|
2|/////111/|
3|111//1.1/|
4|..1//1.21|
5|111//1...|
6|/////1.21|
7|/////111/|
8|111//////|
9|..1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 1 9 free

 |123456789|
-|---------|
1|/////////|
2|/////111/|
3|111//1.1/|
4|..1//1.21|
5|111//1...|
6|/////1.21|
7|/////111/|
8|111//////|
9|1.1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 1 4 free

 |123456789|
-|---------|
1|/////////|
2|/////111/|
3|111//1.1/|
4|1.1//1.21|
5|111//1...|
6|/////1.21|
7|/////111/|
8|111//////|
9|1.1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 7 4 free

 |123456789|
-|---------|
1|/////////|
2|/////111/|
3|111//1.1/|
4|1.1//1121|
5|111//1...|
6|/////1.21|
7|/////111/|
8|111//////|
9|1.1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 7 5 free

 |123456789|
-|---------|
1|/////////|
2|/////111/|
3|111//1.1/|
```

```
4|1.1//1121|
5|111//11..|
6|/////1.21|
7|/////111/|
8|111//////|
9|1.1//////|
-|---------|
Set/unset mines marks or claim a cell as free: > 8 5 free

 |123456789|
-|---------|
1|/////////|
2|/////111/|
3|111//1.1/|
4|1.1//1121|
5|111//112.|
6|/////1.21|
7|/////111/|
8|111//////|
9|1.1//////|
-|---------|
Congratulations! You found all the mines!
```

Report a typo

See hint

Write a program

Code Editor IDE

Java

```java
package minesweeper;
import java.util.*;

public class Main {

    public static void initialize(char[][] array){
        for(int i = 0; i < 9; i++){
            for(int j = 0; j < 9; j++){
                array[i][j] = '.';
            }
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        char[][] array = new char[9][9];
        initialize(array);
        System.out.print("How many mines do you want on the field? ");
        int n = sc.nextInt();
        printArray(array);
        mines(array,n);
        setFree(array, n);
        sc.close();
    }


    public static void setFree(char[][] array, int n){
        Scanner sc = new Scanner(System.in);
        int x, y;
        String str;
        while(n != 0){
            System.out.print("Set/unset mines marks or claim a cell as free: ");
            x = sc.nextInt();
            y = sc.nextInt();
            str = sc.next();

            if(isMine(array, x, y, str)){
                return;
            }

            else if(isCovered(array)){
                printMineArray(array);
                System.out.println("Congratulations! You found all the mines!");
                return;
            }

            else if(str.equals("mine") && array[y-1][x-1] == 'X'){
                array[y-1][x-1] = '$';
```

```
                n--;
            }

            else if(str.equals("mine")){
                if(array[y-1][x-1] == '.'){
                    array[y-1][x-1] = '*';
                }
                else if(array[y-1][x-1] == '*'){
                    array[y-1][x-1] = '.';
                }
                else if(array[y-1][x-1] == '$'){
                    array[y-1][x-1] = 'X';
                }
            }

            else if(str.equals("free")){
                setNumber(array, x, y);
            }

            printMineArray(array);

        }
        System.out.println("Congratulations! You found all the mines!");
    }

    public static void setNumber(char[][] array, int x, int y){
        int startI = getStartI(y - 1);
        int startJ = getStartJ(x - 1);
        int endI = getEndI(y - 1);
        int endJ = getEndJ(x - 1);
        int count = 0;
        for(int m = startI; m <= endI; m++){
            for(int n = startJ; n <= endJ; n++){
                if(array[m][n] == 'X'){
                    count++;
                }
            }
        }
        if(count != 0){
            array[y-1][x-1] = (char)(count + 48);
            return;
        }
        else if(count == 0){
            array[y-1][x-1] = '/';
            isPossible(array, x, y);
            for(int m = startI; m <= endI; m++){
                for(int n = startJ; n <= endJ; n++){
                    isPossible(array, n + 1, m + 1);
                    if(array[m][n] == '.' ){
                        setNumber(array, n + 1, m + 1);
                    }
                }
            }
        }
    }

    public static void isPossible(char[][] array, int x, int y){
        int startI = getStartI(y - 1);
        int startJ = getStartJ(x - 1);
        int endI = getEndI(y - 1);
        int endJ = getEndJ(x - 1);

        for(int m = startI; m <= endI; m++){
            for(int n = startJ; n <= endJ; n++){
                if(array[m][n] == '*' || array[m][n] == '.' || array[m][n] == '$'){
                    setNumber(array, n + 1, m + 1);
                }
            }
        }
    }

    public static boolean isCovered(char[][] array){
        int count = 0;
        for(int i = 0; i < 9; i++){
            for(int j = 0; j < 9; j++){
                if(array[i][j] == '.'){
                    continue;
                }
                else{
                    count++;
                }
            }
        }
        if(count == 81){
            return(true);
        }
```

```
        return(false);
    }

    public static boolean isMine(char[][] array, int x, int y, String str){
        if(array[y - 1][x - 1] == 'X' && str.equals("free")){
            printArray(array);
            System.out.println("You stepped on a mine and failed!");
            return(true);
        }
        return(false);
    }

    public static int getStartI(int i){
        if(i == 0){
            return(i);
        }
        return(i - 1);
    }

    public static int getStartJ(int j){
        if(j == 0){
            return(j);
        }
        return(j - 1);
    }

    public static int getEndI(int i){
        if(i == 8){
            return(i);
        }
        return(i + 1);
    }

    public static int getEndJ(int j){
        if(j == 8){
            return(j);
        }
        return(j + 1);
    }


    public static void mines(char[][] array, int n){
        int i,j;
        Random rand = new Random();
        while(n != 0){
            i = rand.nextInt(9);
            j = rand.nextInt(9);
            if(array[i][j] == '.'){
                n--;
                array[i][j] = 'X';
            }
        }
    }

    public static void printMineArray(char[][] array){
        System.out.println("\n |123456789|");
        System.out.println("-|---------|");
        for(int i = 0; i < 9; i++){
            System.out.print((i+1) + "|");
            for(int j = 0; j < 9; j++){
                if(array[i][j] == '$'){
                    System.out.print('*');
                }
                else if(array[i][j] == 'X'){
                    System.out.print('.');
                }
                else{
                    System.out.print(array[i][j]);
                }
            }
            System.out.println("|");
        }
        System.out.println("-|---------|");
    }

    public static void printArray(char[][] array){
        System.out.println("\n |123456789|");
        System.out.println("-|---------|");
        for(int i = 0; i < 9; i++){
            System.out.print((i+1) + "|");
            for(int j = 0; j < 9; j++){
                if(array[i][j] == '$'){
                    System.out.print('*');
                }
                else{
                    System.out.print(array[i][j]);
```

```
                }
            }
            System.out.println("|");
        }
        System.out.println("-|---------|");
    }
}
```

✓ **Correct.**

You're doing great!

44 users liked this problem. 2 didn't like it. **What about you?**

Continue | Solve again | Solutions (62)

Comments (27) | Hints (6) | Useful links (1) | Solutions (62) | Show discussion