Minesweeper Game - Basic CLI implementation of the Minesweeper game.

Project Overview:

The **Minesweeper Game** project is a Java-based command-line interface (CLI) game that replicates the classic Minesweeper experience, allowing players to reveal cells and mark potential mines on a grid. The game leverages Maven to manage dependencies and build processes, making it easy to integrate and maintain within an IDE like Eclipse.

Key Features:

Randomised Game Grid: Generates a new game grid with hidden mines placed at random locations each time the game starts.

CLI Commands for Game Actions:

1) Reveal cells to uncover safe spots or mines

2) Mark cells as mines based on player predictions.

Game Mechanics: Shows hints (numbers) indicating how many mines are adjacent to each revealed cell.

End Conditions:Victory:

1) Revealing all non-mine cells.

2) Loss: Revealing a cell containing a mine.

Project Setup:

Prerequisites

Java JDK: Ensure that JDK 8 or higher is installed.

Maven: This project uses Maven for dependency management and builds.

Eclipse IDE: The project is set up for Eclipse, but any IDE with Maven support will work.

Folder Structure:

src/main/java: Holds core game logic and classes

src/test/java: Contains unit tests to validate game logic.

pom.xml: Configures Maven dependencies, plugins, and build settings.

Step by step and execution code:

1) Java Installation procedure:

a) Go to the [Oracle Java SE Downloads](#) page.

ORACLE Products Industries Resources Customers Partners Developers Company View Accounts Contact Sales

WARNING: These older versions of the JDK are provided to help developers debug issues in older systems. **They are not updated with the latest security patches and are not recommended for use in production.**

For production use Oracle recommends downloading the latest JDK version.

Only developers and enterprise administrators should download these releases.

For current Java releases, please visit Oracle Java SE Downloads.

Java SE Development Kit 18.0.2.1

This software is licensed under the Oracle No-Fee Terms and Conditions License.

Product / File Description	File Size	Download
Linux Arm 64 Compressed Archive	172.70 MB	https://download.oracle.com/java/18/archive/jdk-18.0.2.1_linux-aarch64_bin.tar.gz (sha256)
Linux Arm 64 RPM Package	154.18 MB	https://download.oracle.com/java/18/archive/jdk-18.0.2.1_linux-aarch64_bin.rpm (sha256)
Linux x64 Compressed Archive	173.85 MB	https://download.oracle.com/java/18/archive/jdk-18.0.2.1_linux-x64_bin.tar.gz (sha256)
Linux x64 Debian Package	149.19 MB	https://download.oracle.com/java/18/archive/jdk-18.0.2.1_linux-x64_bin.deb (sha256)
Linux x64 RPM Package	155.75 MB	https://download.oracle.com/java/18/archive/jdk-18.0.2.1_linux-x64_bin.rpm (sha256)

b) Scroll down and select "JDK 18" under the **Java SE Development Kit** section.

c) Accept the **Oracle licence agreement** if prompted, and download the installer suitable for your operating system

Verify the Installation:

a) Open a command prompt or terminal.

b) Type java -version and press Enter.

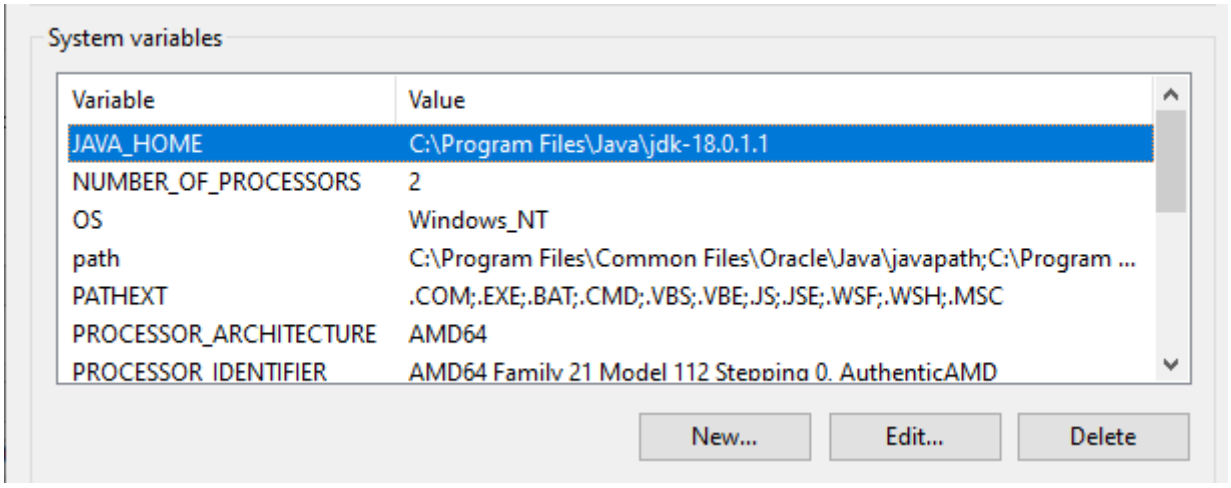
c)You should see output indicating Java 18 is installed.

Command prompt:

C:\Windows\system32\cmd.exe

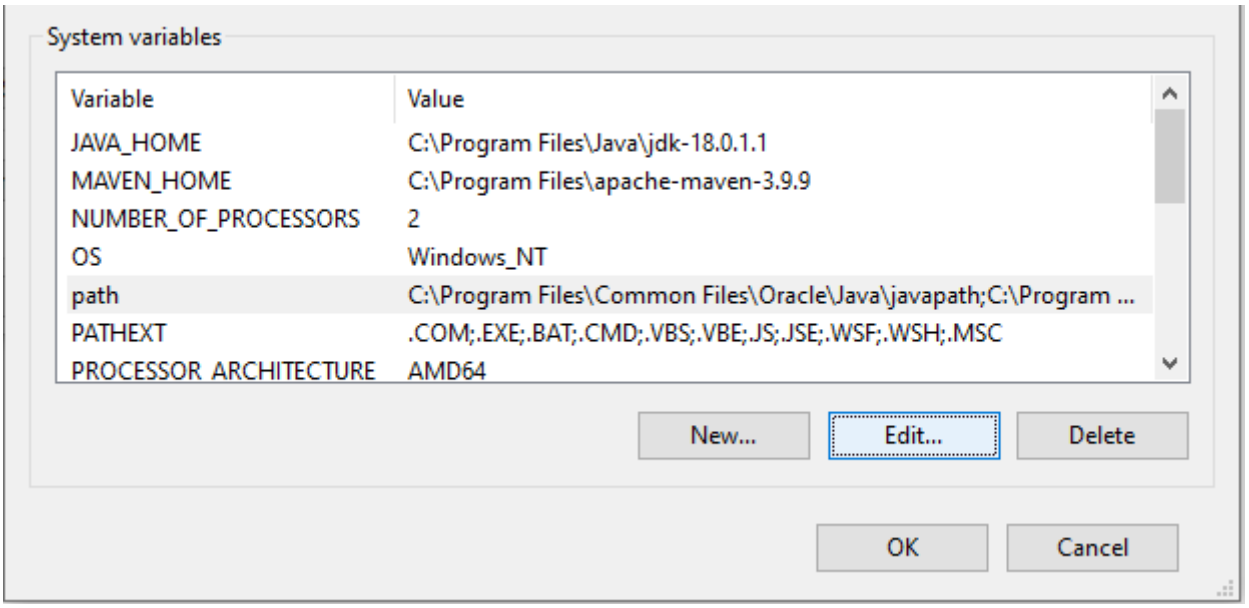
```
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Users\manik>java -version
java version "18.0.2.1" 2022-08-18
Java(TM) SE Runtime Environment (build 18.0.2.1+1-1)
Java HotSpot(TM) 64-Bit Server VM (build 18.0.2.1+1-1, mixed mode, sharing)
```

Add Jdk Path In Environment Variable:



After click on environment variables in the system variable click on path and just click on edit and Give new and just paste the jdk bin path.

C:\Program Files\Java\jdk-18.0.1.1\bin



2) Maven Installation procedure:

a) Go to the Apache Maven download page.

b) Under the **Files** section, select **Binary zip archive** or **Binary tar.gz archive** based on your operating system.

c) Download the latest version of Maven

Maven webpage:

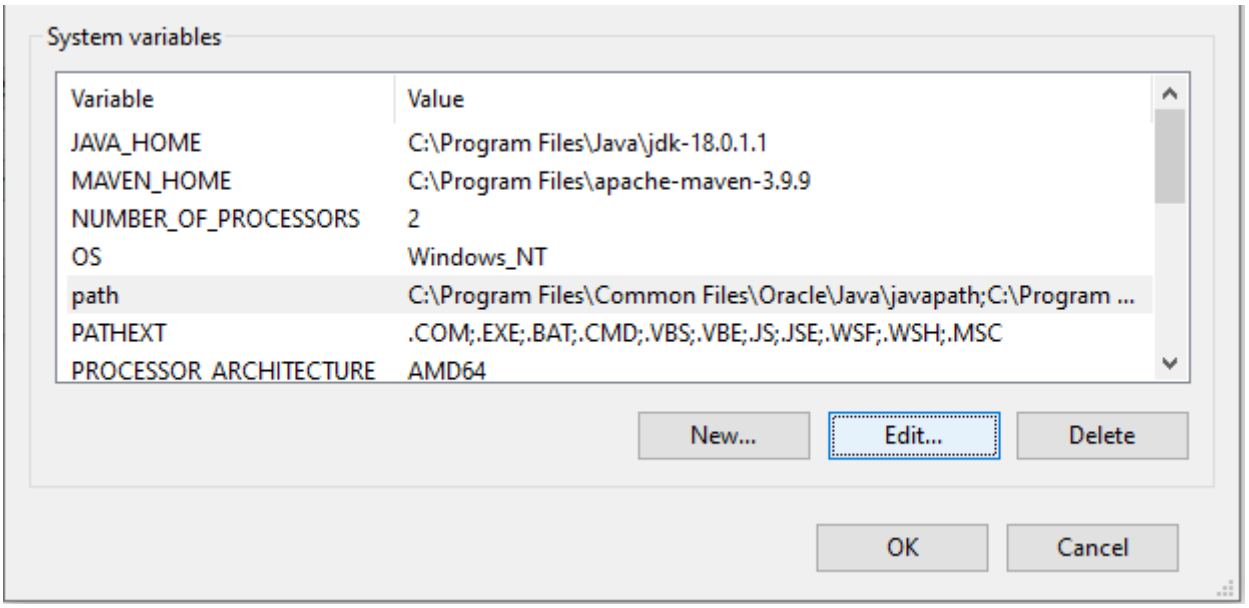
Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the installation instructions. Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to verify the signature of the release bundles against the public KEYS used by the Apache Maven developers.

	Link	Checksums	Signature
Binary tar.gz archive	apache-maven-3.9.9-bin.tar.gz	apache-maven-3.9.9-bin.tar.gz.sha512	apache-maven-3.9.9-bin.tar.gz.asc
Binary zip archive	apache-maven-3.9.9-bin.zip	apache-maven-3.9.9-bin.zip.sha512	apache-maven-3.9.9-bin.zip.asc
Source tar.gz archive	apache-maven-3.9.9-src.tar.gz	apache-maven-3.9.9-src.tar.gz.sha512	apache-maven-3.9.9-src.tar.gz.asc
Source zip archive	apache-maven-3.9.9-src.zip	apache-maven-3.9.9-src.zip.sha512	apache-maven-3.9.9-src.zip.asc

a) Add a bin path from the maven folder to system variable.



b) Check the command prompt the maven is installed.

Command Prompt:

```
C:\Users\Manik>mvn -version
Apache Maven 3.9.9 (57804ffe001d7215be7bcb531cf83df38f93546)
Maven home: C:\Program Files\apache-maven-3.9.9
Java version: 18, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-18
Default locale: en_US, platform encoding: Cp1252
OS name: "windows server 2019", version: "10.0", arch: "amd64", family: "windows"
```

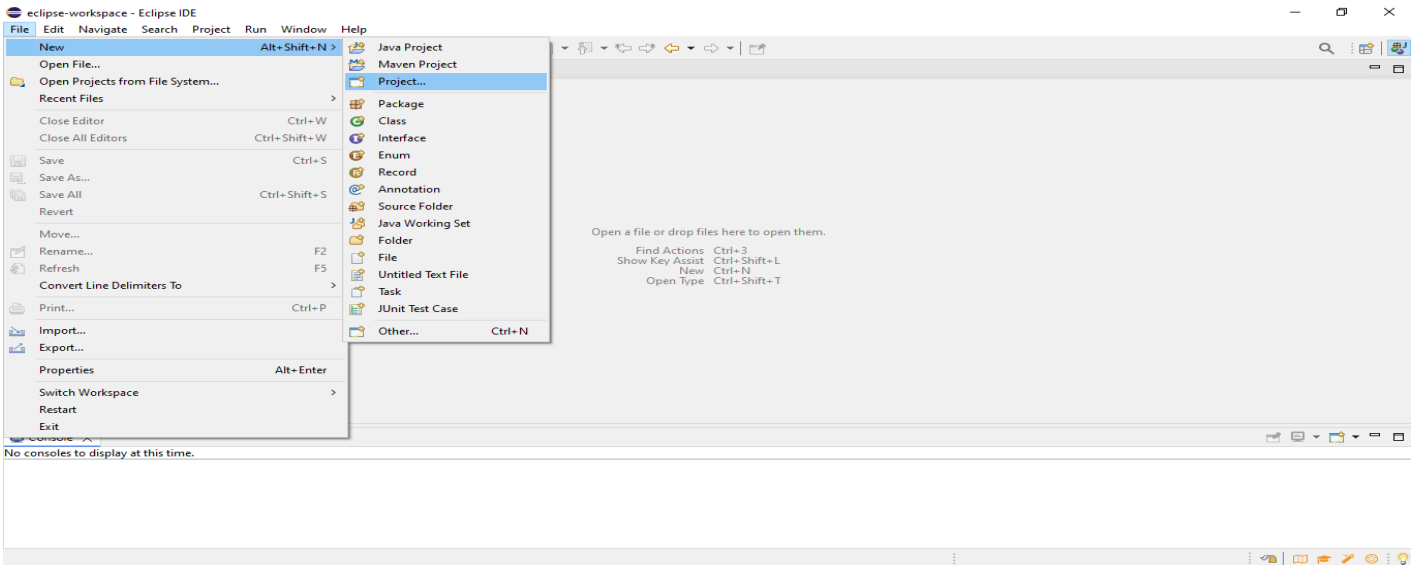
Step 1: Set Up a New Maven Project in Eclipse.

1. Open Eclipse:

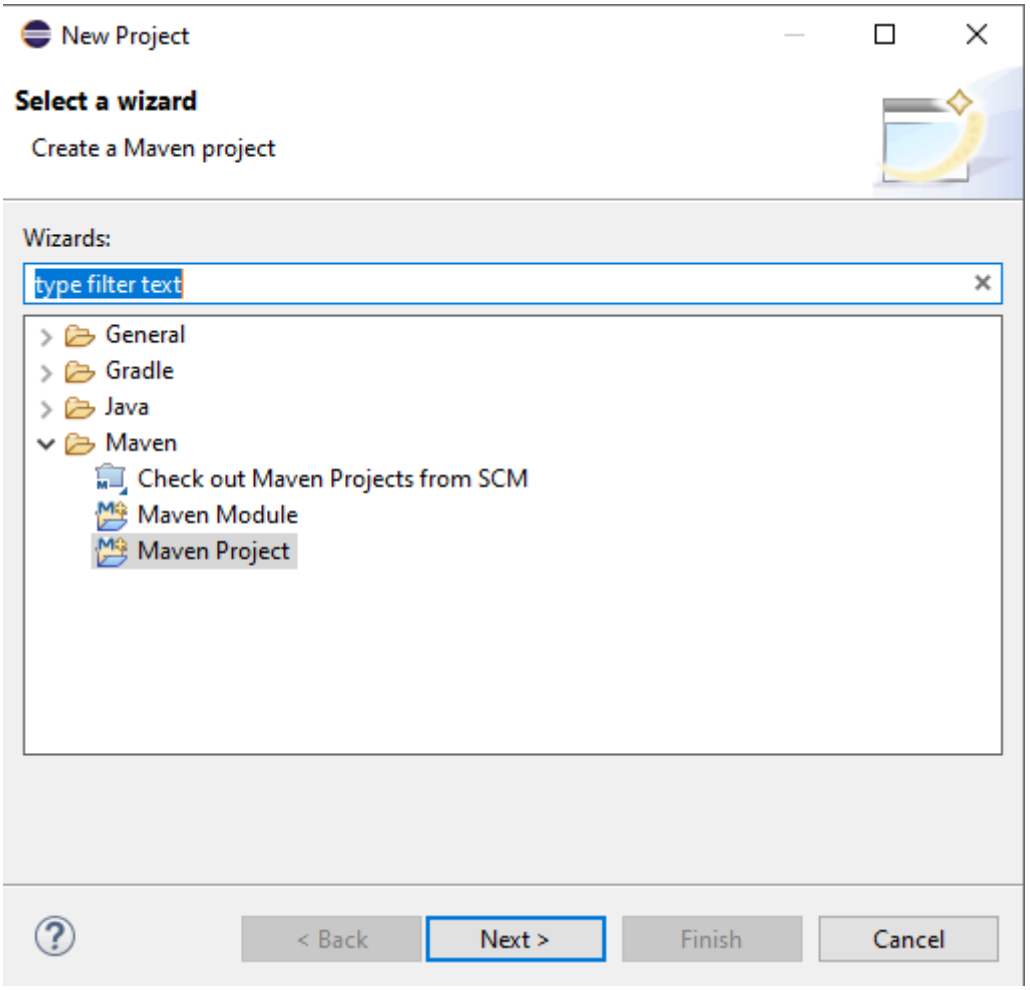
Launch your Eclipse IDE and ensure that the Maven plugin is installed (it comes pre-installed in most recent versions).

2. Create a New Maven Project:

Go to File > New > Other.



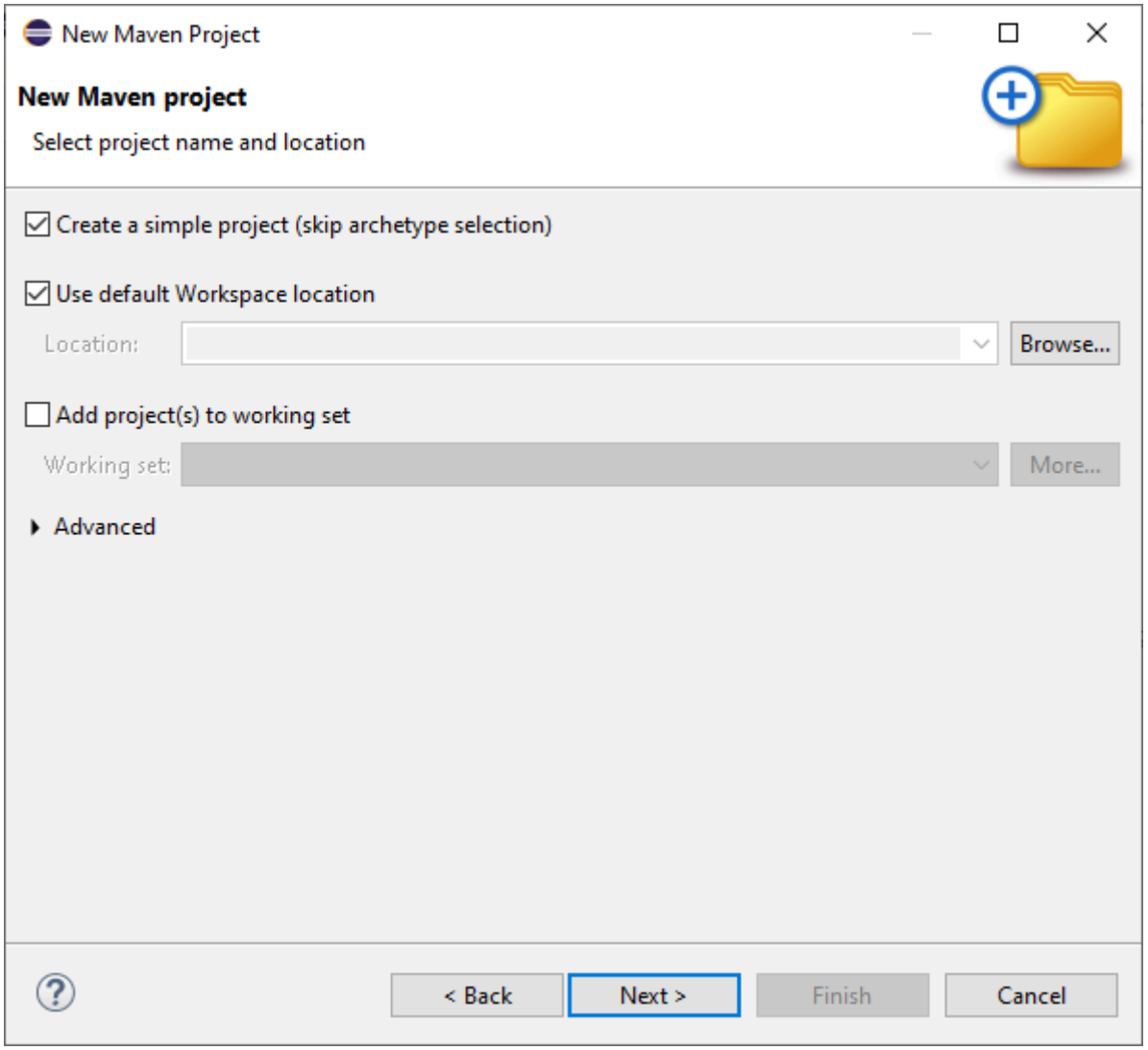
3) In the section of the project select **“Maven project”.**



4) click on the Maven project.

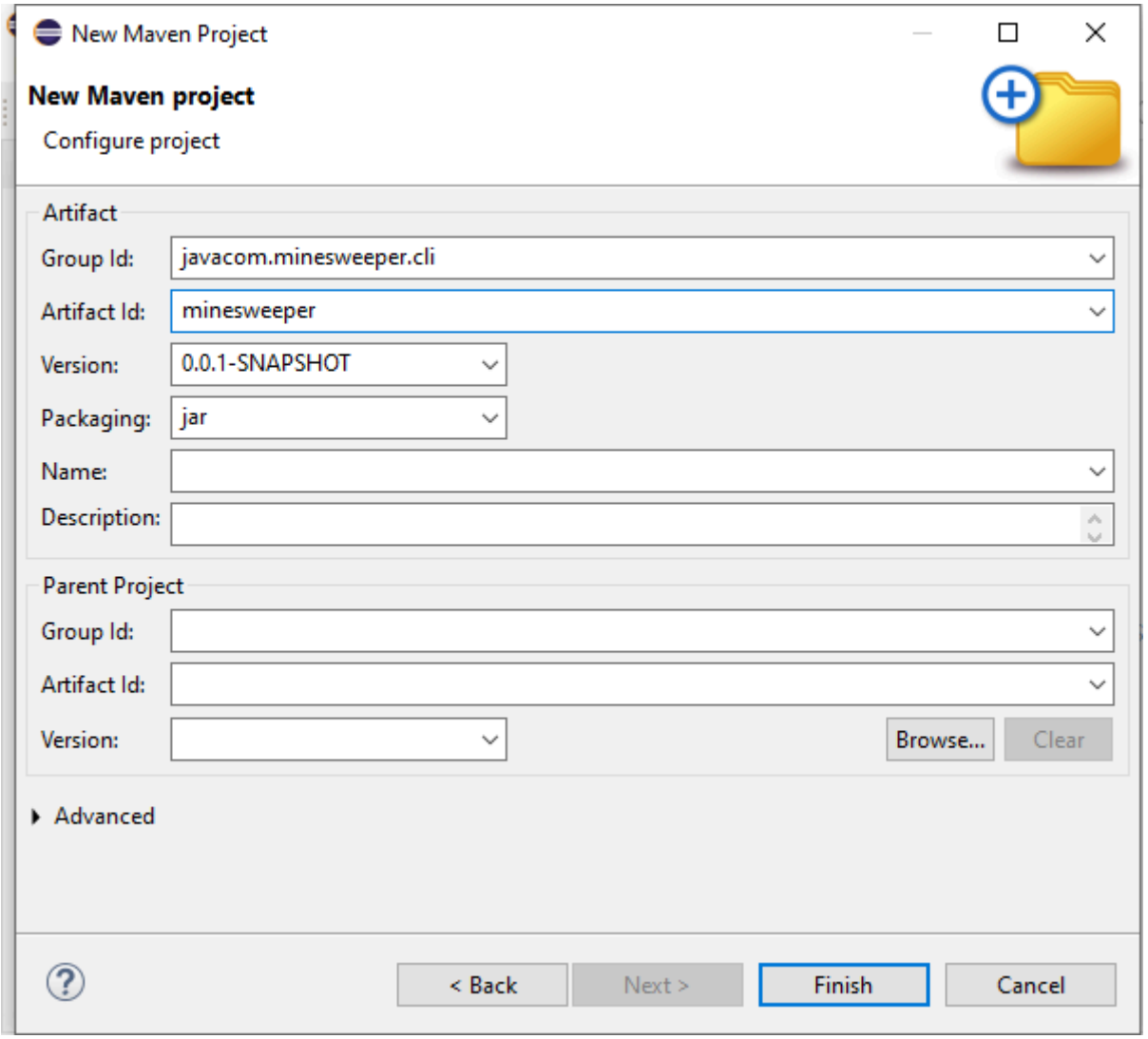
In the section of new maven project choose the create a simple project(skip archetype selection

And also the use default workspace location.



5) Fill the group id and artifact id.

In the section of new maven project, give the name for **group id: java.com.minesweeper.cli** and **Artifact id:minesweeper.**



Step3: Add dependencies

1. Open the pom.xml file in the root directory of your maven project.

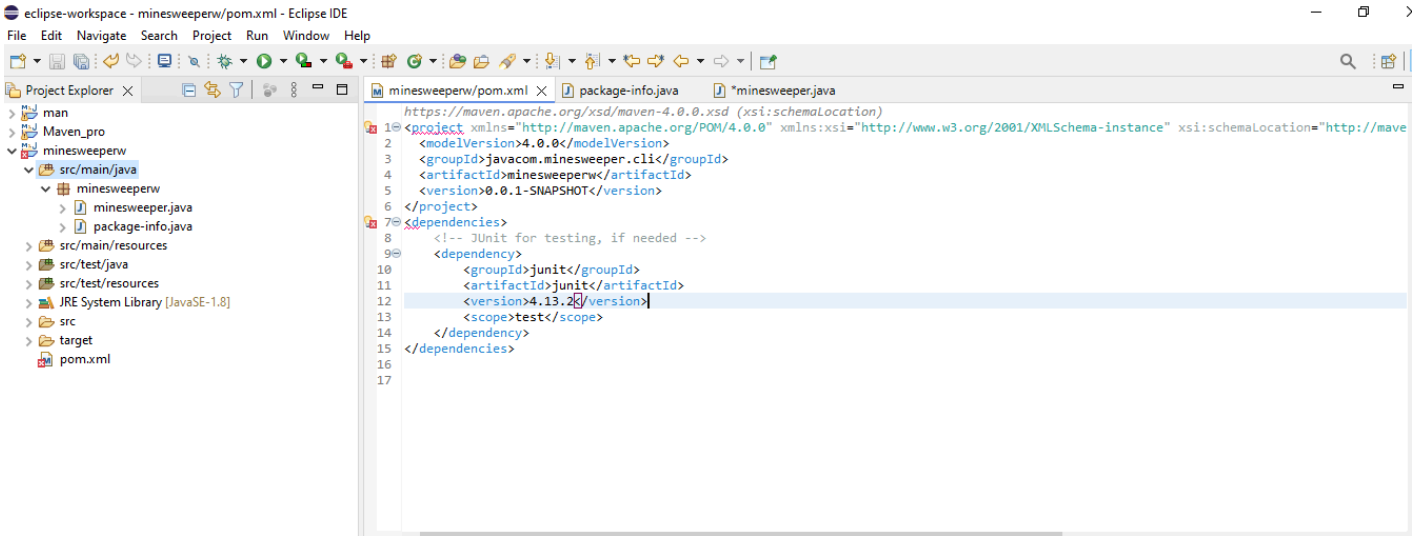
2.Add any required dependencies (if necessary like j unit for testing) within the <dependencies> tag.

```xml
<dependencies>
    <!-- JUnit for testing, if needed -->
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13.2</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

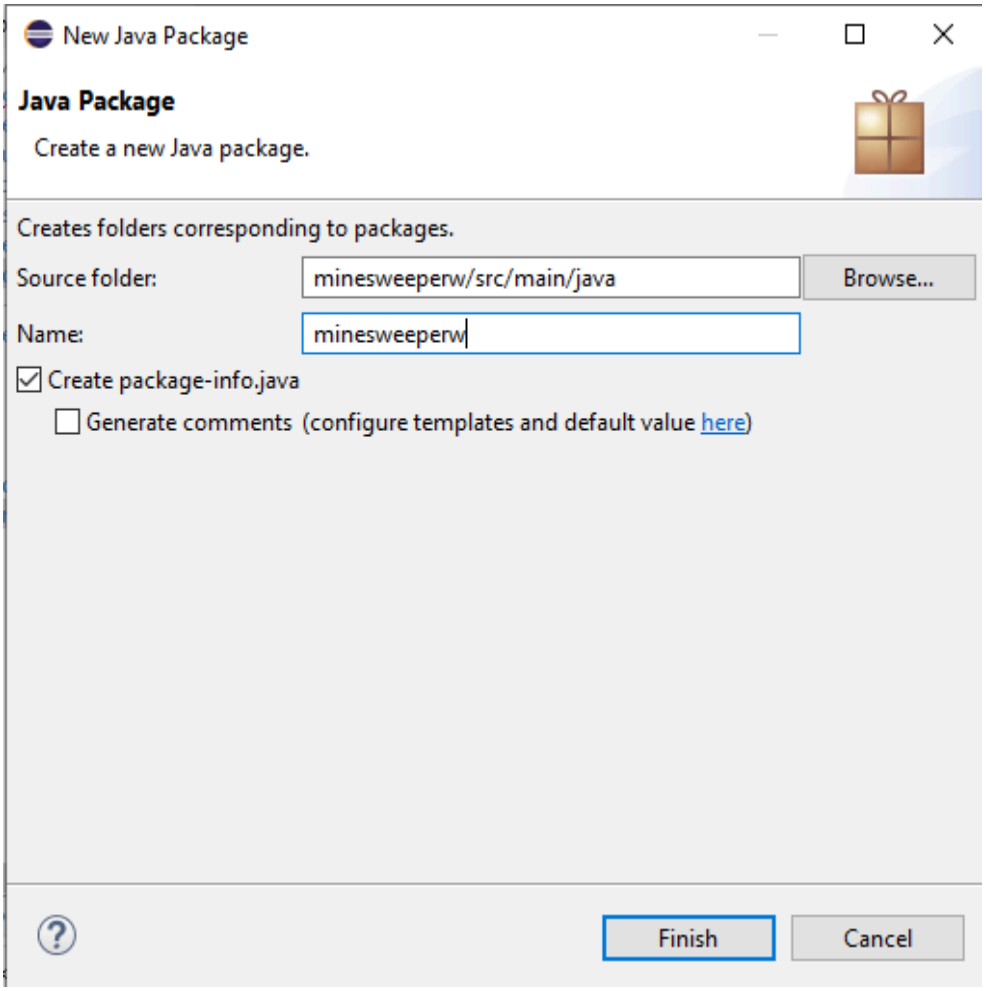
3) Save the **pom.xml** file. Maven will download the dependencies automatically.

Step 4: Implement the Minesweeper Game

1) **Locate the src directory**: Open **src/main/java** in your project.

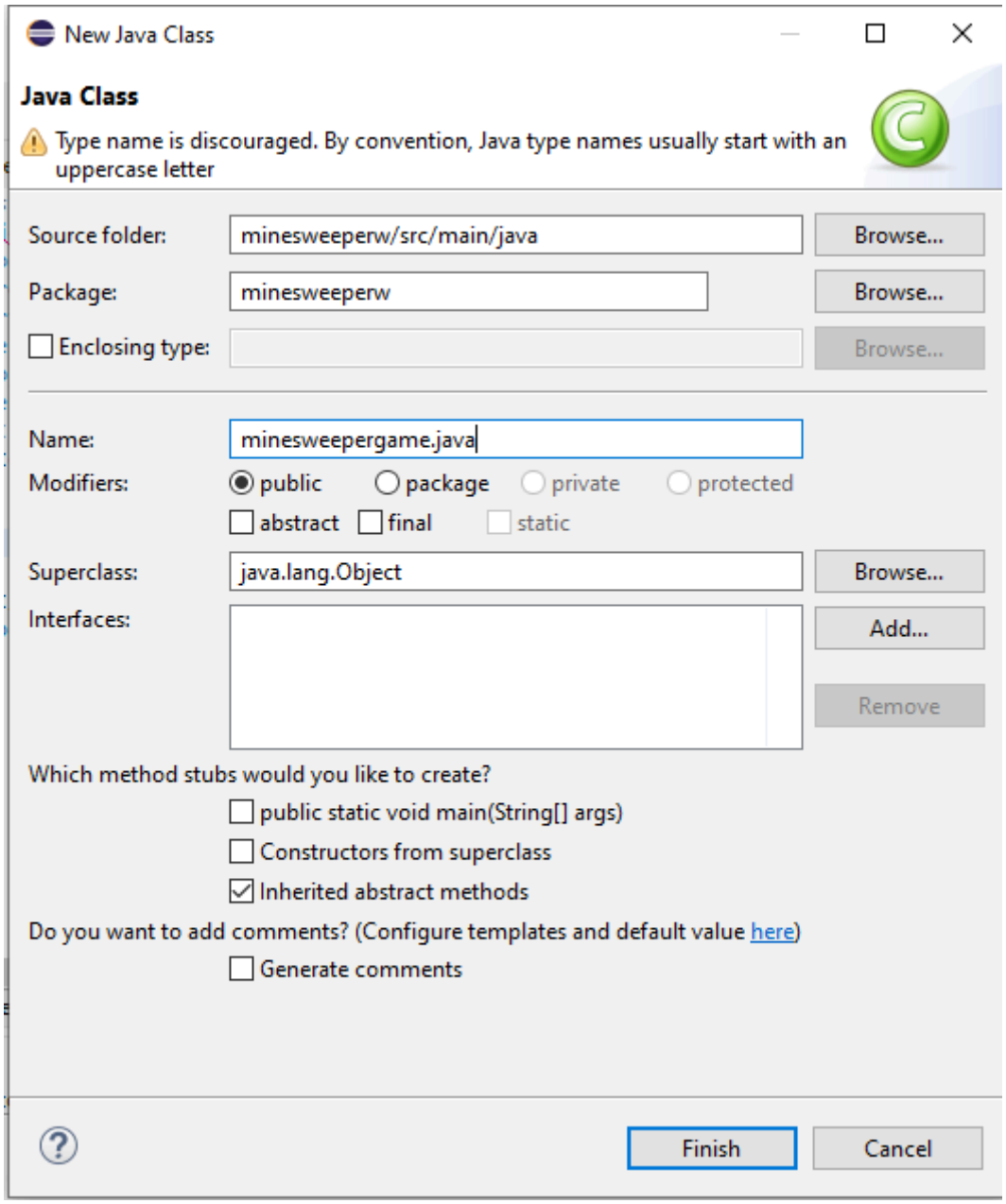


2) Create a **package** named"**minesweepergame.java** and click on finish to create to New java class inside the package in eclipse integrated development environment.

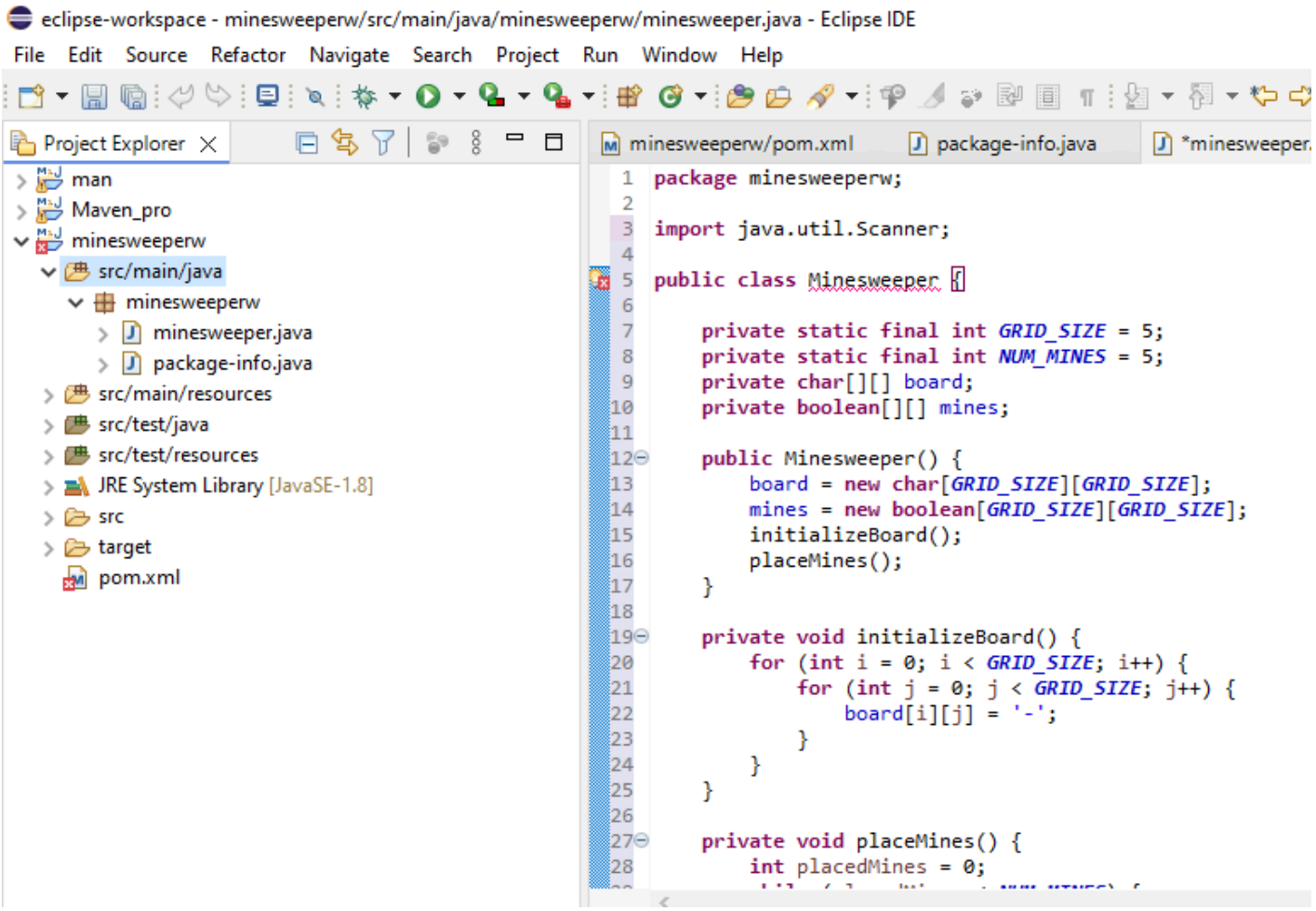


,

3) Inside the package, create a **Java class** file and click on finish.



4) Implement the Minesweeper game logic in **MinesweeperGame.java in src/main/java**



Sample Code for Minesweeping program:

```
package com.minesweeper.cli;

import java.util.Scanner;

public class MinesweeperGame {

    private static final int GRID_SIZE = 5;

    private static final int NUM_MINES = 5;

    private char[][] board;

    private boolean[][] mines;


    public MinesweeperGame() {
```

```
        board = new char[GRID_SIZE][GRID_SIZE];

        mines = new boolean[GRID_SIZE][GRID_SIZE];
        initializeBoard();

        placeMines();
    }

    private void initializeBoard() {

        for (int i = 0; i < GRID_SIZE; i++) {

            for (int j = 0; j < GRID_SIZE; j++) {

                board[i][j] = '-';
            }
        }
    }

    private void placeMines() {

        int placedMines = 0;

        while (placedMines < NUM_MINES) {

            int x = (int) (Math.random() * GRID_SIZE);

            int y = (int) (Math.random() * GRID_SIZE);
            if (!mines[x][y]) {

                mines[x][y] = true;

                placedMines++;
            }
        }
    }

    private void printBoard() {

        for (int i = 0; i < GRID_SIZE; i++) {

            for (int j = 0; j < GRID_SIZE; j++) {

                System.out.print(board[i][j] + " ");
            }
            System.out.println();
        }
    }

    public void startGame() {

        Scanner scanner = new Scanner(System.in);

        boolean gameRunning = true;

        System.out.println("Welcome to Minesweeper!");
```

```
        while (gameRunning) {

            printBoard();
            System.out.print("Enter row and column (e.g., 0 1): ");
            int row = scanner.nextInt();
            int col = scanner.nextInt();
            if (mines[row][col]) {
                System.out.println("Boom! You hit a mine. Game Over.");
                gameRunning = false;
            } else {
                board[row][col] = 'O'; // O for Opened cell
                System.out.println("Safe! Keep going.");
            }
        }
        scanner.close();
    }

    public static void main(String[] args) {
        MinesweeperGame game = new MinesweeperGame();
        game.startGame();
    }
}
```

5) **Run the Application:**

1) Right-click on MinesweeperGame.java > Run As > Java Application.

2) The game will launch in the Eclipse console. Follow the prompts to play the game by entering coordinates.

eclipse-workspace - minesweeperw/src/main/java/minesweeperw/minesweeper.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- man
- Maven_pro
  - src/main/java
  - src/main/resources
  - src/test/java
  - src/test/resources
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - src
  - target
  - pom.xml
- minesweeperw
  - src/main/java
    - minesweeperw
      - minesweeper.java
      - package-info.java
  - src/main/resources
  - src/test/java
  - src/test/resources
  - JRE System Library [JavaSE-1.8]
  - src
  - target
  - pom.xml

minesweeperw/pom.xml | package-info.java | minesweeper.java

```java
package minesweeperw;

import java.util.Scanner;

public class Minesweeper.java {

    private static final int GRID_SIZE = 5;
    private static final int NUM_MINES = 5;
    private char[][] board;
    private boolean[][] mines;

    public Minesweeper() {
        board = new char[GRID_SIZE][GRID_SIZE];
        mines = new boolean[GRID_SIZE][GRID_SIZE];
        initializeBoard();
        placeMines();
    }

    private void initializeBoard() {
        for (int i = 0; i < GRID_SIZE; i++) {
            for (int j = 0; j < GRID_SIZE; j++) {
                board[i][j] = '-';
            }
        }
    }

    private void placeMines() {
        int placedMines = 0;
        while (placedMines < NUM_MINES) {
            int x = (int) (Math.random() * GRID_SIZE);
            int y = (int) (Math.random() * GRID_SIZE);
            if (!mines[x][y]) {
                mines[x][y] = true;
                placedMines++;
            }
        }
    }

    private void printBoard() {
```

Writable | Smart Insert | 20 : 46 : 514

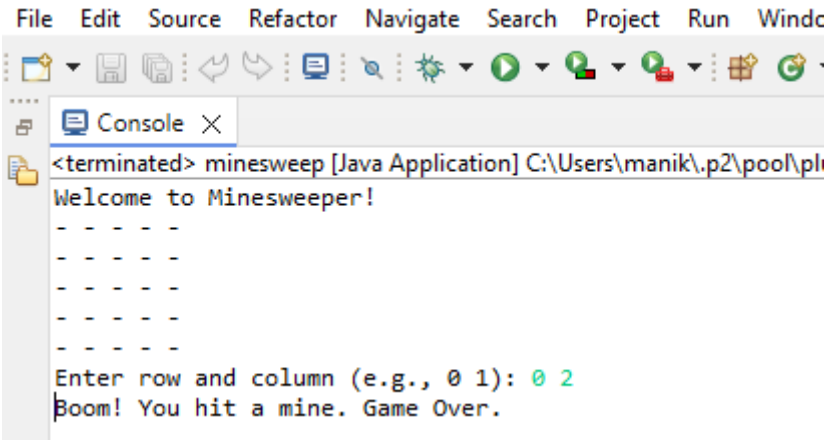
The console displays,each method prints a success message upon successful reading or writing of the file, or an error message if there is an issue.

Input:

Console

```
minesweep [Java Application] C:\Users\manik\.p2\pool\plug
Welcome to Minesweeper!
- - - - -
- - - - -
- - - - -
- - - - -
- - - - -
Enter row and column (e.g., 0 1): 0 2
```

Output:



conclusion:

In summary, we created a CLI Minesweeper game in Java with functionality to save and load the game state using a CSV file. By implementing **saveGameToCSV()** and **loadGameFromCSV(),** players can store and resume their game. Console messages confirm successful saving and loading, enhancing user interaction.

This project covers key Java concepts, including Maven for dependency management, file handling, and user input, providing a strong foundation for more complex projects.