**VIETNAM NATIONAL UNIVERSITY – HCM INTERNATIONAL UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**



**Algorithms & Data Structure**

**PROJECT REPORT**

Semester 2, 2023 - 2024

**Instructor:**

### Topic: Minesweeper

Team:

17 pages

**TEAM MEMBERS**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Name** | **Student IDs** | **Contribution** |
| **1** | Nguyễn Khoa Minh Toàn | ITITIU21331 | 25% |
| **2** | Nguyễn Doãn Tiến Dũng | ITCSIU21119 | 25% |
| **3** | Hoàng Việt Tân | ITCSIU21231 | 25% |
| **4** | Phạm Huỳnh Thanh Quân | ITDSIU21110 | 25% |

**TABLE OF CONTENTS**

**CHAPTER 1: INTRODUCTION 2**

1. Prepare 2
2. What is the new? 2
3. About Boom 3

**CHAPTER 2: PROJECT TIMELINE 4**

**CHAPTER 3: DESIGN AND IMPLEMENTATION 6**

1. UML Diagram 6
2. Elements designing 11
3. Project structure 12
4. How it runs? 14

**CHAPTER 4: CONCLUSION 17**

1. Final words 17
2. References 18

# CHAPTER 1: INTRODUCTION

## I. Prepare

Welcome to the enchanting world of "Minesweeper - by BOOM,” a brilliantly reimagined version of the classic Minesweeper game. BOOM has meticulously crafted this rendition to honor the timeless puzzle game while elevating it to unprecedented levels. Dive into a gaming experience that marries the nostalgic charm of Minesweeper with cutting-edge innovations.

At the core of Minesweeper - by BOOM is a user-friendly interface designed to welcome both veterans of the game and new players. The seamless navigation and intuitive design make uncovering the mysteries hidden beneath the tiles a delightful and engaging experience.

While maintaining the familiar mechanics of the classic Minesweeper, BOOM introduces a plethora of new features that will redefine your gaming journey. Prepare yourself for an immersive experience that challenges your strategic thinking and problem-solving abilities. Whether you're a casual player looking for a relaxing break or a devoted Minesweeper enthusiast seeking new excitement, BOOM ensures a gaming environment that is inclusive and thoroughly enjoyable for all.

Embark on an adventure filled with suspense, excitement, and strategic decision-making as you navigate through the minesweeper board. The carefully designed levels strike a perfect balance between difficulty and entertainment, guaranteeing hours of captivating gameplay.

Join us in exploring the intricacies of Minesweeper - by BOOM, where the blend of classic appeal and modern ingenuity creates an unparalleled gaming masterpiece. Experience Minesweeper like never before, where each click and decision reveals a world of surprises. Are you ready to navigate the mines and emerge victorious in this thrilling adventure? The challenge awaits!

## II. What is the new?

### 1.Customization Options

Minesweeper - by BOOM allows players to personalize their gaming experience through the "Custom Level" option. Players can set the number of mines, rows, and columns according to their preferences, offering a tailored and adjustable challenge that caters to individual play styles.

### 2.Sound System

The game incorporates a sophisticated sound system with distinct audio cues for various in-game events. Players can enjoy immersive sound effects for tile uncovering, flag placement, and other actions. The ability to toggle sound on or off allows players to customize their auditory experience, enhancing the overall immersion.

### 3.Timekeeping and Competitive Element

A timekeeping feature adds a competitive edge to the game. Players can track the time taken to clear the minefield, introducing a strategic component to the gameplay. This feature encourages players to compete for faster completion times, adding a sense of accomplishment and urgency.

### 4.Rankings System

After completing a game, players can view a display of the top completion times. This ranking system fosters friendly competition among players, motivating them to refine their strategies and aim for quicker mine-clearing times, enhancing the replayability of the game.

### 5.Visual Enhancements

Minesweeper - by BOOM boasts a visually appealing interface with distinct icons that differentiate various elements on the minesweeper grid. These visual enhancements not only contribute to the overall aesthetic appeal but also improve gameplay clarity and enjoyment.

### 6.Dynamic Level Selection

The game offers three preset difficulty levels: Easy, Medium, and Hard. Players can select a level that matches their skill and experience. Additionally, the customizable difficulty option allows players to create a unique challenge tailored to their preferences. This dynamic level selection accommodates both newcomers and seasoned Minesweeper players.

### 7.Adaptive Grid Size

The grid size adjusts based on the selected difficulty level, providing a balanced and engaging gameplay experience. Different grid sizes add variety to the challenges presented at each level, keeping the game fresh and exciting.

### Enhanced Gameplay Features

* Custom Level Settings: Tailor your game with specific numbers of mines, rows, and columns.
* Sound Effects: Enjoy distinct sounds for different actions, with the option to toggle sound on or off.
* Timekeeping: Track your completion times and strive for faster records.
* Rankings Display: See top completion times and compete for higher rankings.
* Visual Appeal: Enjoy an enhanced interface with clear and appealing graphics.
* Difficulty Levels: Choose from Easy, Medium, Hard, or customize your difficulty.
* Adaptive Grids: Experience varied grid sizes for balanced gameplay

## III. About BOOM

* **THE DEVELOPER TEAM:**

Team Nowaycode is a dynamic group of four third-year students majoring in Computer Networks at the International University, National University Ho Chi Minh City. The team members, namely NGUYEN DOAN TIEN DUNG (ITCSIU21173), PHAM HUYNH THANH QUAN(ITDSIU21110), NGUYEN KHOA MINH TOAN (ITITIU21331), and HOANG VIET TAN (ITCSIU21231) are passionate and diligent individuals dedicated to exploring the realms of information technology.

Motivated by a shared curiosity and a desire to delve deeper into the mechanics of a childhood classic, Minesweeper, the team embarked on the journey of crafting a revamped and fully-fledged version of the game. Each member brought their unique skills and perspectives to the project, resulting in a collaborative effort that seamlessly blends traditional gameplay with contemporary enhancements.

As aspiring professionals in the field of computer networks, the team saw Minesweeper as an opportunity to dissect the inner workings of a nostalgic game while honing their technical expertise. Their energetic and industrious approach to the project reflects their commitment to both personal growth and the creation of a fresh, innovative gaming experience.

Choosing Minesweeper as their project allowed the team to not only revisit a cherished childhood game but also to challenge themselves in developing a comprehensive and novel iteration. The upgraded version stands as a testament to their dedication, showcasing their ability to transform a classic into a modern masterpiece.

In summary, Team Nowaycode is a vibrant collective of IT enthusiasts, eager to push boundaries, learn, and contribute to the world of technology. Their journey with Minesweeper reflects their ambition, teamwork, and pursuit of knowledge in the ever-evolving landscape of game development It is

# CHAPTER 2: PROJECT TIMELINE

|  |  |  |  |
| --- | --- | --- | --- |
| **Stage** | **Action** | **Member** | **Week No.** |
| Planning | Individual’s topic research | All | 1 |
| Set up project timeline | Toan |
| Topic confirmation | All |
| Project preparation | Create Github repository | Toan | 2 |
| Build and managing project files structure | Toan |
| Research for tutorial and sample code | Dung, Tan |
| Research for references and technical documents | All |
| Agree on means of communication, workflow and tools | All |
| Decide scopes, aims, goals and requirements of the project | All |
| Demo version 1 | Making the first game demo | Dung, Tan | 3 - 4 |
| Evaluate and testing | Quan |
| Research for in-game characters and game mechanisms | Quan |
| User Interface | Research for game concept | Dung, Tan | 5 - 7 |
| Agree on means of game concept | All |
| Evaluate and make adjustment | Toan |
|  |  |
| Demo version 2 | Create Time | Dung | 8 - 11 |
| Create Application | All |
| Create Display | Tan |
|  |  |
|  | Create Level | Toan |  |
| Create Mineseeper | Toan |
|  | Create MainTile | All |  |
| Create ScoreFileHandle | Quan |
| Create Sound | Quan |
| Finalize and clean up code | All |
| Presentation | Final report | All | 12 - 14 |
| Presentation slide | All |

# CHAPTER 3: Design and implementation

**A diagram of a computer

Description automatically generated with medium confidence**

**1**

**.**

**UML**

**Diagram**

Figure

1.

UML

diagram

**MainTile:**

* The MineTile class represents individual tiles or cells within the Minesweeper game grid. Each MineTile object corresponds to a specific location in the grid, identified by its row and column. The class primarily serves as a container for information related to a tile's position.
* Key Responsibilities:
  + Tile Position Information: The MineTile class stores information about the row and column of a tile within the Minesweeper grid.
  + Extending JButton: It extends the JButton class, implying that each MineTile is a clickable button in the graphical user interface.
  + Encapsulation of Tile Properties: The class encapsulates properties specific to a Minesweeper tile, such as its row, column, and any additional attributes that might be added in the future.
  + Consistent Tile Size: The TILESIZE constant ensures a consistent size for each tile in the graphical representation of the Minesweeper grid.
  + Getter Methods: Provides getter methods to retrieve the row and column of a MineTile object.

**Level:**

* The Level class manages the configuration and state of the Minesweeper game at different difficulty levels. It encapsulates information such as the number of rows, columns, mine count, and the current game level. This class provides methods to set and retrieve these parameters, making it a central hub for controlling the game's difficulty and tracking its progress. The players always start at an easy level.
* Key Responsibilities:
  + Configuring Game Parameters: The Level class is responsible for storing and updating parameters that define the game's configuration, including the number of rows (numRows), columns (numCols), mine count (mineCount), and the current game level (level).
  + Managing Game State: It manages the game state, including whether the player has won (winGame). This information is crucial for determining when the game is completed.
  + Setting and Retrieving Parameters: The class provides methods to set and retrieve values for the number of rows, columns, mine count, game level, and win state. These methods allow other parts of the program to interact with and modify the game's configuration.
  + Setting Default Levels: The class includes methods (setLevelEasy, setLevelMedium, setLevelHard) to set default levels, each with predefined values for rows, columns, and mine count. These methods facilitate easy initialization of the game at different difficulty levels.
  + Setting Custom Level: The setLevel method allows the customization of game parameters based on user input. It ensures that the specified number of mines is within a valid range, adjusting the game configuration accordingly. If the input is invalid, it defaults to an easy level.

**Sound:**

* The Sound class encapsulates functionality related to playing and managing sound effects in the Minesweeper game. It provides methods for loading sound files, playing, stopping, and toggling sound effects. The class allows for easy integration of audio feedback in the game, enhancing the overall user experience.
* Key Responsibilities:
* Loading Sound Files: The class initializes an array of Clip objects (clips) to store different sound effects. It loads sound files from the specified file path prefix and associates them with the corresponding Clip objects.
* Playing Sound Effects: The playSound method plays a specific sound effect identified by its index. It ensures that the sound is played from the beginning, stopping any ongoing playback of the same sound effect.
* Toggling Sound: The toggleSound method switches the state of sound playback between enabled and disabled. It prints a message indicating whether the sound is muted or unmuted.
* Stopping Sound: The stopSound method stops the playback of a specific sound effect identified by its index. If the sound effect is currently playing, it is halted.
* Starting Sound: The startSound method begins playing a specific sound effect from the beginning, irrespective of its current state. This is useful for repeated playback of a sound effect without interruption.
* Getting and Setting Sound Enabled State: The getSoundEnabled and setSoundEnabled methods provide access to the current state of sound playback. The state can be modified to control whether sound effects are played.
* All of these methods will be used in other classes to support playing/stopping audio

**Time:**

* The Time class is designed to measure the time difference between two distinct signals, named Signal A and Signal B, using Java's Instant and Duration classes. It provides methods to receive these signals, calculate the time difference between them, and obtain the time difference in seconds.
* Responsibilities:
  + Receiving Signals: The receiveSignalA method records the timestamp (Instant) when Signal A is received and prints a corresponding message. Similarly, the receiveSignalB method records the timestamp (Instant) when Signal B is received and prints a corresponding message.
  + Calculating Time Difference: The calculateTimeDifference method calculates the duration between Signal A and Signal B using the Duration.between method. If either Signal A or Signal B has not been received, it throws an IllegalStateException.
  + Getting Time Difference in Seconds: The getTimeDifferenceInSeconds method calls calculateTimeDifference to obtain the time difference in Duration and then converts it to seconds using toSeconds.

A screen shot of a computer

Description automatically generated

*Figure III.3.d.1: converting Duration to second.*

**ScoreFileHandle:**

* The ScoreFileHandler class is responsible for managing the scores recorded during the gameplay, including saving scores to files, reading scores from files, and generating a formatted string with completion time rankings.
* Key Responsibilities:
  + - Saving Score to File: The saveScoreToFile method takes a file name and a score as parameters and appends the score to the specified file. It uses FileWriter and BufferedWriter to achieve this.

A screen shot of a computer program

Description automatically generated

*Figure III.3.e.1: save the score to file txt.*

* + - Reading Scores from File: The readScoreFromFile method reads scores from a specified file and returns them as a list of Long values. It uses FileReader and BufferedReader for file reading.
    - Saving Score According to Level: The saveScore method determines the appropriate file name based on the current level (easy, medium, or hard) and then calls saveScoreToFile to save the score to that file.
    - Generating Formatted Score String: The toStringScore method takes a file name, reads scores from that file, sorts them, and then creates a formatted string with completion time rankings. It displays the top 10 scores and fills in "Null" for any remaining places.

**Display:**

* The Display class is responsible for managing the graphical user interface (GUI) elements of the Minesweeper game. It handles the creation and configuration of the game window, buttons, labels, and icons. Additionally, it provides methods to handle user interactions with the GUI, such as button clicks and level selections.

A screenshot of a computer program

Description automatically generated

*Figure III.3.f.1: declare the components in the application’s interface.*

* Key Components:
  + JFrame and GUI Components: The class uses a JFrame as the main window for the Minesweeper game. It includes buttons for level selection, custom game creation, play again, viewing rankings, and toggling sound. It also contains a label for displaying game information.
  + Icons: Icons, such as flagIcon, bombIcon, unclickedIcon, nullIcon, and numberIcons, are loaded for various game elements. These icons are essential for visual representation of the game board.
  + GUI Configuration: The class configures the layout, size, and appearance of GUI components, ensuring a visually appealing and user-friendly interface. It sets fonts, borders, and colors for different components.
  + Button Actions: The class provides methods for handling button actions, such as toggling sound, displaying rankings, starting a new game, and selecting game levels.
  + Level Configuration: The getTextSize method determines the font size based on the number of columns in the game, improving readability.
  + Visibility Control: The visible method controls the visibility of the game window.

**Minesweeper:**

* The Minesweeper class is the main class responsible for managing the game logic of Minesweeper. It handles the creation of the game board, player interactions, mine placement, and game outcome scenarios. Additionally, it integrates with the Display class to update the graphical user interface (GUI) and provide a seamless gaming experience.
* Key Components:
  + Game Initialization: The runGame method initializes the game board, sets up the GUI, and manages user interactions. It also incorporates buttons for level selection, custom game creation, play again, viewing rankings, and sound toggling.
  + Mine Placement: The initializeMines method places mines on the game board. It ensures that the first click is safe by excluding the clicked tile and its neighboring tiles from mine placement.
  + Mouse Event Handling: The class handles mouse events, distinguishing between left and right clicks. Left clicks reveal tiles, and right clicks place or remove flags. Sound effects are triggered based on these actions.
  + Recursive Tile Checking: The checkMine method recursively reveals tiles when an empty tile (with no adjacent mines) is clicked. It helps in uncovering connected areas of the game board.
  + Game Over and Win Conditions: The revealMines method reveals all mines when the game is lost, marking the end of the game. The win condition is checked when all non-mine tiles are clicked, triggering a win scenario with a congratulatory message and saving the completion time to a file.
  + Flag Management: The number of planted flags and the remaining flags are updated during flag placement and removal. The numOfPlantedFlags variable is used to keep track of the flags planted on the board.
  + Time Management: The Time class is used to measure the time taken to complete the game. Signals A and B are received to calculate the time difference.
  + Sound Effects: Sound effects are played during various game events, such as clicking on tiles, winning, losing, or toggling sound. The Sound class is utilized for sound management.
  + Integration with Display: The Display class is instantiated within the Minesweeper constructor, providing integration with the GUI. Methods from Display are called to handle button actions, update text labels, and control the visibility of the game window.

## Elements designing.

**A blue and white pattern on a white surface

Description automatically generated**

*Figure III.1.1: Unclicked tile.*

A blue and white background

Description automatically generated

*Figure III.1.2: Background (none of mine surrounded).*

**A red and white fish

Description automatically generatedA black and white fish

Description automatically generatedA group of fish swimming in water

Description automatically generatedA group of fish swimming in water

Description automatically generatedA group of fish swimming in water

Description automatically generatedA group of fish swimming in water

Description automatically generatedA group of fish swimming in water

Description automatically generated**

*Figures III.1.3 - 10: Koi-fish images represent the number of mines that surround them (from 1 to 8).*

A cartoon of a person fishing

Description automatically generated

*Figure III.1.11: The fishing man represents the flag (used to note the mine tile)*

A logo with a bomb and text

Description automatically generated

*Figure III.1.12: And the exploded mine (when clicking on the mine).*

## Project Structure

A screenshot of a computer program

Description automatically generated

Figure 2. Project structure

## 4. How it runs?

* Core Components:

Minesweeper Class:

* Initialization: The game starts by instantiating the Minesweeper class, which initializes the game board, handles user interactions, and integrates with the GUI provided by the Display class.
* User Interactions: Mouse events are crucial for gameplay. Left-clicking reveals tiles, and right-clicking places or removes flags. Sound effects accompany these actions, enhancing the player's experience.

A black background with text

Description automatically generated

*Figure III.4.1: mouse events listener.*

*A screen shot of a computer

Description automatically generated*

*Figure III.4.2:  attach the left mouse detector.*

*A black background with colorful text

Description automatically generated*

*Figure III.4.3: attach the right mouse detector.*

* Mine Placement: Mines are strategically placed on the board, ensuring the first click is safe. Recursive tile checking reveals empty areas when tiles are clicked, enhancing gameplay dynamics.

A screen shot of a computer program

Description automatically generated

*Figure III.4.4: check the tile then recursively into 8 surrounding tiles.*

* Win and Lose Conditions: The game checks for win conditions when all non-mine tiles are revealed. Losing conditions triggers mine tile revelations and ends the game. The time taken for completion is recorded for ranking purposes.

A screen shot of a computer program

Description automatically generated

*Figure III.4.5: reveal the mines.*

Display Class:

* Graphical User Interface (GUI): The Display class manages the game window, buttons, and labels. It offers buttons for level selection, custom game creation, play again, rankings, and sound toggling. The GUI is intuitive and responsive.

Button Actions: Buttons respond to player actions, such as selecting levels, creating custom games, playing again, viewing rankings, and toggling sound. The GUI provides a seamless experience throughout the game.

Sound Class:

* Sound Effects: The Sound class handles the game's audio aspects, playing sound effects for various events, including tile clicks, winning, losing, and sound toggling. The integration of sound enhances the overall gaming atmosphere.

Time Class:

* Time Measurement: The Time class measures the time taken to complete the game. Signals A and B are used to calculate the time difference, providing a time-based ranking system for players.

A screen shot of a computer code

Description automatically generated

*Figure III.4.6: point the time at the start.*

A screen shot of a computer program

Description automatically generated

*Figure III.4.7: point the time when the player wins the game then save that to file.*

* Gameplay Flow:

Upon game initiation, the player can interact with the GUI to select a level, customize a game, or toggle sound if they want.

A screen shot of a computer program

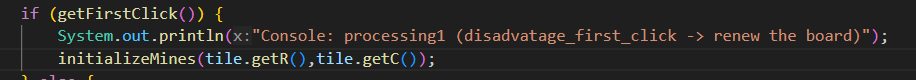
Description automatically generated  
 *Figure III.4.8: Minesweeper’s constructor.*

*A screen shot of a computer program

Description automatically generated*

*Figure III.4.9: class Application runs the game by using Minesweeper’s constructor.*

The game board is displayed, and the player makes the first click, ensuring a safe start.



*Figure III.4.10: if the first click is on the mine or Koi-fish image, renew the board.*

Mines are strategically placed on the board, and the game progresses with the player revealing tiles, placing flags, and avoiding mines.

A screen shot of a computer program

Description automatically generated

*Figure III.4.11: on the first-click-tile, initializeMine() method will avoid this tile and 8 tiles surround.*

Recursive tile checking reveals interconnected empty areas, contributing to the game's strategic depth.

Win and lose conditions are checked, triggering corresponding events and providing feedback to the player.

The player can view rankings and toggle sound during and after the game.

A screen shot of a computer program

Description automatically generated

*Figure III.4.12: Show the rankings from files.*

# CHAPTER 4: Conclusion

1. **Final words**

The Minesweeper game developed by the "Nowaycode" team has been a comprehensive project that goes beyond basic GUI development. After nearly two months of dedicated work, the team has come to a crucial realization: a game's success hinges not only on its software but on various other factors that collectively contribute to the overall player experience.

The project has emphasized the importance of providing rich and engaging content to make the game enjoyable. Analogous to a web server's reliance on content, a game's quality cannot be accurately measured without considering elements such as the game environment, storyline, gameplay mechanics, artwork, and animations.

Throughout the development process, the team encountered and addressed numerous challenges, including debugging and refining the software. This hands-on experience allowed the team to apply lessons learned in class, fostering a deeper understanding of the subject matter. The project catalyzed self-directed learning, prompting team members to extend their studies beyond the classroom.

The Minesweeper project has provided a platform for the team to consolidate their knowledge, update their technological skill set, and gain new experiences. The realization that success in the field of Computer Science often requires substantial self-study has become evident. The team, known as "Nowaycode," is committed to completing the Minesweeper game as their first project and aims to publish it promptly. Additionally, they plan to develop more Android applications to cater to a broader user base.

In conclusion, the Minesweeper game by Nowaycode is not just a software project but an immersive learning experience that encapsulates the multifaceted nature of game development. The team's dedication to self-improvement and their commitment to delivering a quality product highlight the broader challenges and rewards of working in the dynamic field of Computer Science and Engineering. Thank you, Mr. T. Tung and Mr. Nghia Nguyen, for your unwavering support and dedication to our growth as aspiring developers.

1. **References:**

* Sounds: *meme* sound effects on [YouTube](https://www.youtube.com/)
* Logic game: [[Giải Bài Tập Luyện Tư Duy Lập Trình - Giải Thuật] - Bombs (Dò Mìn) ICTU.OLP.CACHHOC.NET - YouTube](https://www.youtube.com/watch?v=3-KOYpyvHXs&t=3564s),  [Stack Overflow - Where Developers Learn, Share, and build Careers](https://stackoverflow.com/),
* Java GUI: [Java GUI: Full Course ☕ (FREE) (youtube.com)](https://www.youtube.com/watch?v=Kmgo00avvEw&t=16281s)