**VIETNAM NATIONAL UNIVERSITY OF HO CHI MINH CITY**

**INTERNATIONAL UNIVERSITY**



A blue and white logo

Description automatically generated with low confidence

**Data Structure & Algorithms**

**FINAL PROJECT DSA**

**“Online Ordering Pizza System”**

**Personal Project: Hoàng Văn Mạnh – ITDSIU21099**

**ABOUT THE PROJECT**

In recent years, information technology has developed in all areas of life as well as other areas of social management. One of the areas that many people are interested in is the field of online shopping. Online shopping both saves travel time and increases income for technology vehicle drivers.

We choose the topic “Online Ordering Pizza System”.Aim of the project is to develop an online facility for customers to order pizzas from a pizza shop.

**Project goal:**

* Learn about DSA in Java and apply it to build games.
* Improve qualifications, self-study, thinking, teamwork, and presentation ability for students, as a solid foundation for students after graduation.

**Research focus and scope:**

* Learn DSA in Java and its application to game programming.
* Understand the classes and methods needed to build the program.
* Research on Data Structure & Algorithms in Java.

**Table Content**

[**1. Introduction 4**](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.1f09sfu2r88l)

[**2. Team Members 5**](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.qr9t895sxb70)

[**3. Technology and Platforms 6**](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.a8m3zpgtku4t)

[*3.1 Programming Language 6*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.sbxeok5do3re)

[*3.2 Development Environment 6*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.vro405wc4of3)

[*3.3 GUI Library 6*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.kgw8ut3qkw5y)

[*3.4 Game Flow 7*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.oojutn6kumjy)

[*3.5 Class Diagrams 8*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.3c7dzyi5glns)

[*3.6 Version Control 8*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.hnvnwdfrig0y)

[*3.7 Testing 8*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.eyc0tqmsjd3g)

[*3.8 Documentation 8*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.dj6g2tx7zq1r)

[**4. Game Implementation 9**](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.5dxg46uf6a1o)

[*4.1 Game Logic 9*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.f1xc9bk03fa6)

[*4.1.1 Game\_Collect 9*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.s56ex4iqfgxl)

[*4.1.2 Player\_Collect 11*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.8uk9zs16uk2d)

[*4.1.3 Point\_Collect 12*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.cyxiotwej9rr)

[*4.1.4 Map\_Collect 14*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.eht2s0t8hzqy)

[*4.2 Key Controls 14*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.bh1ib1mfc9qp)

[*4.3 Game State Management 15*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.8aqgoi20iq44)

[*4.4 Conclusion 15*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.vqt9uqr686o)

[**5. Challenges and Solutions 16**](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.bvp0raemgw0i)

[*5.1 Efficient Game State Management 16*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.a69x4xhfavui)

[*5.2 Smooth User Interaction 16*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.4fatc9ba490b)

[*5.3 Creating an Engaging User Interface 16*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.uae7tte0ixd6)

[*5.4 Conclusion 17*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.cop1ujfl5xin)

[**6. Testing 18**](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.abn8eadq97we)

[*6.1 Login Screen 19*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.7pilrjbp17lb)

[*6.2 Main Menu ( Home screen ) 19*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.s8svcp6crj1w)

[*6.3 Configuration ( Change background ) 20*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.3cqpcjnj4yfo)

[*6.4 Difficulties Screen 20*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.zerkzpxsgk95)

[*6.5 Gameplay 21*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.54ukiq7czcto)

[*6.6 Finish ( Game Over screen ) 21*](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.3qwtvkyts1ml)

[**7. Conclusion 22**](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.hbb9r319fjys)

[**8. References 23**](https://docs.google.com/document/d/1Zr-SzmrJbdPMj-Bj1rF91TEpss9Xb8VHi9NHZ2nMTNM/edit#heading=h.djlxzzhnohu4)

1. **Introduction**

In the rapidly evolving world of technology, the realm of game development has emerged as a dynamic field that combines creativity, problem-solving, and technical prowess. This report presents an in-depth exploration of a unique game—the ‘Collecting’ game—developed by our team.

The ‘Collecting’ game is an interactive computer game meticulously crafted using the Java programming language. It is a testament to the power of object-oriented programming, showcasing how encapsulation, inheritance, and polymorphism can be leveraged to create complex, yet manageable, software systems.

The game revolves around a simple yet engaging concept: a player navigates a map to collect points. Despite its apparent simplicity, the game provides an immersive experience, challenging players to think strategically and react quickly.

This report will delve into the various aspects of the game’s development process, from the initial design stages to the final testing and debugging. It will highlight the technologies and platforms used, the implementation of the game logic and user interface, and the challenges encountered along the way.

Through this report, we aim to provide valuable insights into the game development process, shedding light on the intricate balance between technical implementation and user experience design. We hope that our journey in creating the ‘Collecting’ game will serve as a useful reference for future game developers and enthusiasts alike.Team Members

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Seq.** | **Full Name** | **ID** | **Contribution** | **Note** |
| **1** | Trần Bảo Phúc | ITDSIU21109 | 25% |  |
| **2** | Phan Danh Đức | ITDSIU21012 | 25% |  |
| **3** | Nguyễn Nam | ITDSIU21101 | 25% |  |
| **4** | Hoàng Văn Long | ITDSIU21096 | 25% |  |

Our team for the development of the ‘Collecting’ game consisted of dedicated individuals, each contributing their unique skills and expertise. Here’s a brief introduction to our team members:

1. **Trần Bảo Phúc:** was the lead developer for the project, who was responsible for implementing the game logic and ensuring the smooth functioning of the game.

1. **Phan Danh Đức:** was in charge of the game’s user interface, and designed and implemented the game’s GUI, ensuring it was intuitive and engaging.

1. **Nguyễn Nam:** was responsible for testing the game and conducted rigorous testing to identify and fix bugs and ensure the game provided a smooth user experience.

1. **Hoàng Văn Long:** was the project’s documentarian, who was responsible for documenting the game’s development process, maintaining the project’s documentation, and preparing this report.
2. **Technology and Platforms**

**3.1 Programming Language**

The primary programming language used in this project is Java. Java was chosen for its object-oriented programming features, which are ideal for game development. It also has a robust standard library that supports a wide range of functionalities.

**3.2 Development Environment**

The project was developed using the Integrated Development Environment (IDE) Intellij. Intellij offers features like code completion, debugging, and refactoring that enhance productivity and code quality.

The version of JDK is 19.0.2 or 19(2).

**3.3 GUI Library**

The graphical user interface (GUI) of the game was created using Java's Swing library. Swing provides a set of 'lightweight' (all-Java language) components that, to the maximum degree possible, work the same on all platforms.

**3.4 Game Flow**

A diagram of a flowchart

Description automatically generated

**3.5 Class Diagrams**

Class Diagrams: [Link…](https://drive.google.com/file/d/1rEnkVq1-zJNSQX-H6NfxK0zcaQpPSFba/view?usp=sharing)

A computer screen shot of a computer

Description automatically generated

**3.6 Version Control**

We used Git for version control, with a shared repository on GitHub. This allowed the team to collaborate effectively, track changes, and maintain different versions of the project.

**3.7 Testing**

JUnit was used for unit testing to ensure the correctness of the code. It helped us to check the logic of individual pieces of the program and debug issues.

**3.8 Documentation**

Javadoc was used for documenting the code. This helped in understanding the functionality of different classes and methods in the project.

1. **Game Implementation**

**4.1 Game Logic**

The game logic is encapsulated within several classes, each responsible for a specific aspect of the game.

**4.1.1 Game\_Collect**

This class extends the abstract Game class and controls the game state, including the player’s movements, the addition of points, and the game’s win/lose conditions. It also features a ticking mechanism, where certain actions are performed at each tick.

A screen shot of a computer program

Description automatically generated

**run() function shows the movement of these blocks**

**A screen shot of a computer program

Description automatically generated**

**operate() function works when we control the line(player)**

**A screen shot of a computer code

Description automatically generated**

**Responsible for updating the game's points based on the current difficulty level.**

**4.1.2 Player\_Collect**

This class represents the player, which can move left or right on the map. The player’s position is represented as an array of Cell objects.

A screen shot of a computer program

Description automatically generated

**Responsible for moving the player (represented by three cells) to the left within the game map.**

**A screen shot of a computer code

Description automatically generated**

**Responsible for moving the player (represented by three cells) to the right within the game map.**

**4.1.3 Point\_Collect**

This class represents points in the game. Points can move and charge, and they appear to be randomly placed on the map.

A computer screen shot of code

Description automatically generated

**Responsible for moving a point within the game and checking when the collision between the point and player is happening.**

A screen shot of a computer code

Description automatically generated

**This provides the queue of columns, and it will randomly peek out of the queue to locate the point position.**

**4.1.4 Map\_Collect**

This class represents the game map, which is a 2D grid where each cell can be free or occupied by a player or a point.

The GUI of the game is handled by the Panel\_Collect class, which extends the GamePanel class. The game panel handles key press events to control the player’s movements and the game’s pause state. The GraphicCell class handles the visual representation of the cells in the game’s GUI.

A screenshot of a computer

Description automatically generated

**4.2 Key Controls**

The game uses key listeners to handle user inputs. When a key is pressed, the game checks if the game is paused. If not, it checks the key code and performs the corresponding operation. For example, if the left arrow key or the ‘A’ key is pressed, the player moves left.

**4.3 Game State Management**

The game state, including the player’s score, level, and time, is managed by the Game\_Collect class. The game state is updated in response to game events, such as the player collecting a point or the game timer ticking.

**4.4 Conclusion**

The implementation of the ‘Collecting’ game demonstrates effective use of object-oriented programming principles, data structures, and algorithms. By encapsulating game logic and user interface code into separate classes, the game achieves a high degree of modularity and readability. The use of data structures and algorithms enables efficient game state management and user interaction handling.

1. **Challenges and Solutions**

The development of the ‘Collecting’ game was an enriching experience that presented several challenges. This section outlines some of the key challenges we faced and the solutions we implemented.

**5.1 Efficient Game State Management**

One of the primary challenges in developing the ‘Collecting’ game was managing the game state efficiently. The game state includes the player’s position, the positions of the points, the score, and the game status (running, paused, or game over).

**Solution**: We used a combination of data structures to manage the game state. The player’s position and the positions of the points were stored in arrays of Cell objects. The score and game status were stored in variables within the Game\_Collect class. This allowed us to quickly and efficiently update and retrieve the game state.

**5.2 Smooth User Interaction**

Ensuring smooth user interaction was another challenge. The game needed to respond quickly and accurately to user inputs.

**Solution**: We used key listeners to handle user inputs. When a key was pressed, the game checked the key code and performed the corresponding operation. This ensured that the game responded immediately to user inputs.

**5.3 Creating an Engaging User Interface**

Creating an engaging and intuitive user interface was a significant challenge. The interface needed to be visually appealing and easy to understand.

**Solution**: We used the Swing library to create the game’s GUI. The GraphicCell class was used to handle the visual representation of the cells in the game’s GUI. Each cell in the game map had a corresponding GraphicCell object that determined how the cell was displayed on the screen.

**5.4 Conclusion**

Overcoming these challenges required a deep understanding of Java and object-oriented programming principles. It also required careful planning and problem-solving skills. Despite these challenges, the development of the ‘Collecting’ game was a rewarding experience that provided valuable insights into game development.

1. **Testing**

**Source code (link github):**

[DSBaoPhuc/DSA-ProjectGame: IU'2023 DSA Game Project - Minesweeper (github.com)](https://github.com/DSBaoPhuc/DSA-ProjectGame)

**Demo video:**

[demo\_1.mp4 - Google Drive](https://drive.google.com/file/d/1jdtFAZ6KCboY9iEuIqM84btHXIpnPJn2/view?fbclid=IwAR0EThOgab6zoMwgcFR1Yo7GSMod76f8h3AU1efR5Pv7F5MvO1GyBxxbJqg)

**GAME RULEs:**

* Player must use A and D (or Left and Right buttons) to move left or right, respectively.
* For each block player collected, the score will increase 100 points.
* In Easy mode, player need 500 points to advance one level. Similarly, Medium mode is 300 points and Hard mode is 200 points.
* The game only finishes when a player misses one block.

**FEATUREs:**

* User login interface
* Can choose the background color according to your preferences.
* Sound effects.
* Many different difficulty levels.
* Time system.
* Highscore system.

**6.1 Login Screen**

A screenshot of a computer login screen

Description automatically generated

**6.2 Main Menu ( Home screen )**

A screenshot of a game

Description automatically generated

**6.3 Configuration ( Change background )**

A screenshot of a computer screen

Description automatically generated

**6.4 Difficulties Screen**

A green rectangular box with black text

Description automatically generated

**6.5 Gameplay**

A screen shot of a game

Description automatically generated

**6.6 Finish ( Game Over screen )**

A screen shot of a computer screen

Description automatically generated

1. **Conclusion**

The development of the ‘Collecting’ game was a challenging yet rewarding journey. It provided us with the opportunity to apply and deepen our understanding of Java and object-oriented programming principles. The game’s development process highlighted the importance of efficient data structures, algorithms, and user interface design in creating an engaging and smooth gaming experience.

Overcoming the various challenges we encountered along the way required careful planning, problem-solving skills, and a deep understanding of the technologies involved. Through rigorous testing, we were able to refine the game, fix bugs, and enhance the user experience.

The ‘Collecting’ game is more than just a game - it is a testament to the power of programming and the creativity it enables. It showcases how abstract concepts can be transformed into a tangible, interactive experience.

As we conclude this report, we hope that our journey in creating the ‘Collecting’ game serves as a valuable reference for future game developers and enthusiasts. We look forward to continuing to explore the exciting world of game development and pushing the boundaries of what is possible with programming.

1. **References**

In the development of the ‘Collecting’ game, we relied on a variety of resources to guide us and provide necessary information. Here are some of the key references:

1. [**Stack Overflow**:](https://stackoverflow.com/) This online community was a great source of solutions for specific programming challenges we encountered during development. Stack Overflow

1. [**GitHub**:](https://github.com/) We used GitHub for version control, which was crucial for managing our codebase and collaborating effectively. GitHub

1. [**Oracle Swing Tutorial**:](https://www.oracle.com/) This tutorial was a helpful guide in creating the game’s GUI using the Swing library. Oracle Swing Tutorial

1. [**GeeksforGeeks:**](https://www.geeksforgeeks.org/)A Computer Science portal for geeks.

1. [**GameDemo:**](https://drive.google.com/file/d/1jdtFAZ6KCboY9iEuIqM84btHXIpnPJn2/view)How to access and play game.

1. [**JDK**:](https://www.oracle.com/java/technologies/javase/jdk19-archive-downloads.html)  19.0.2 or 19(2).

1. **Game Development Tutorials**: Various online tutorials and articles on game development provided insights into game design principles and best practices.