**VIETNAM NATIONAL UNIVERSITY – HCM**

**INTERNATIONAL UNIVERSITY**



**SCHOOL OF COMPUTER SCIENCE**

**AND ENGINEERING**

**OBJECT-ORIENTED PROGRAMMING**

**Semester 1, 2023 - 2024**

**Instructor: Dr. Tran Thanh Tung**

**Topic: Super Mario Bros: Lavaland**

|  |  |  |
| --- | --- | --- |
| **NAME** | **STUDENT ID** | **CONTRIBUTION** |
| Phạm Nguyễn Tiến Đạt | ITDSIU21080 | 20% |
| Trịnh Bình Nguyên | ITDSIU21103 | 20% |
| Lê Trung Kiên | ITDSIU21093 | 20% |
| Phan Nguyễn Hùng Cường | ITDSIU21078 | 20% |
| Dương Nhật Huy | ITDSIU21016 | 20% |

**TABLE OF CONTENTS**

[**I. INTRODUCTION**](#_m2j6b03alpua) **3**

[A. Motivation](#_7r9ck3mp9usv) 3

[B. Task allocation](#_mkd5kzofvorv) 3

[C. Technologies](#_1qqh4h5z202w) 4

[D. Launch](#_szow60tenebr) 4

[**II. THE PROJECT SUPER-MARIO-BROS: LAVALAND**](#_bl7trzqb1wes) **4**

[A. Rules](#_3a0wd2m83cyw) 4

[B. Design](#_r30sbfa0z6f7) 5

[C. Improvements](#_gcvcu3wrunca) 8

[D. Others](#_ntfv2nzdlgfy) 16

[**III. OOP CONCEPTS IMPLEMENTED**](#_6b1n6lkflb8g) **19**

[A. Encapsulation](#_8pditkmzsxqm) 19

[B. Polymorphism](#_4crvl7jfuezv) 19

[C. Inheritance](#_t2uustrfpc4) 20

[D. Abstract](#_6vei8aineajt) 21

[**IV. CONCLUSION**](#_9j96n05exozh) **21**

[A. Achieved Goals](#_6159bbcaan7y) 21

[B. Limitations](#_wke4s69z1fhf) 22

[C. Future Enhancements](#_j3fwnffk9ujt) 23

[**V. REFERENCES**](#_ckb2bwperwll) **23**

# **I. INTRODUCTION**

## **A. Motivation**

Introducing "Super Mario Bros: Lavaland," our latest project that takes a fresh spin on the beloved Mario game. Developed using Java game development, this innovative creation meets our course requirements and goes beyond mere academic fulfillment. Our main goal is to fuse the nostalgia of the classic game with modern elements, resulting in a dynamic and captivating 2D gaming experience. As passionate game developers, this project allows us to fully immerse ourselves in the iconic Mario universe and showcase our creativity. We have no doubt that this project will not only meet course standards, but also make a significant impact on the evolution of 2D gaming.

## **B.** **Task allocation**

|  |  |
| --- | --- |
| **Name** | **Main contribute** |
| Pham Nguyen Tien Dat | Github repository host, file management, and redesign UI. |
| Trinh Binh Nguyen | Develop new mechanisms, new objects based on the original game, improved gameplay, and write README. |
| Le Trung Kien | Accompany with Nguyen to improve the game play, mechanism, and create UML. |
| Phan Nguyen Hung Cuong | Fix bugs in the original game, accompany Kien to create UML, and write the report. |
| Duong Nhat Huy | Make the sfx, accompany Dat to redesign the UI, and prepare powerpoint. |

## **C.** **Technologies**

● Java 21

● Java AWT

● Javax.swing

● Java 2D

## **D.** **Launch**

Game entrance: src/java/manager/GameEngine.java

# **II. THE PROJECT SUPER-MARIO-BROS: LAVALAND**

## **A.** **Rules**

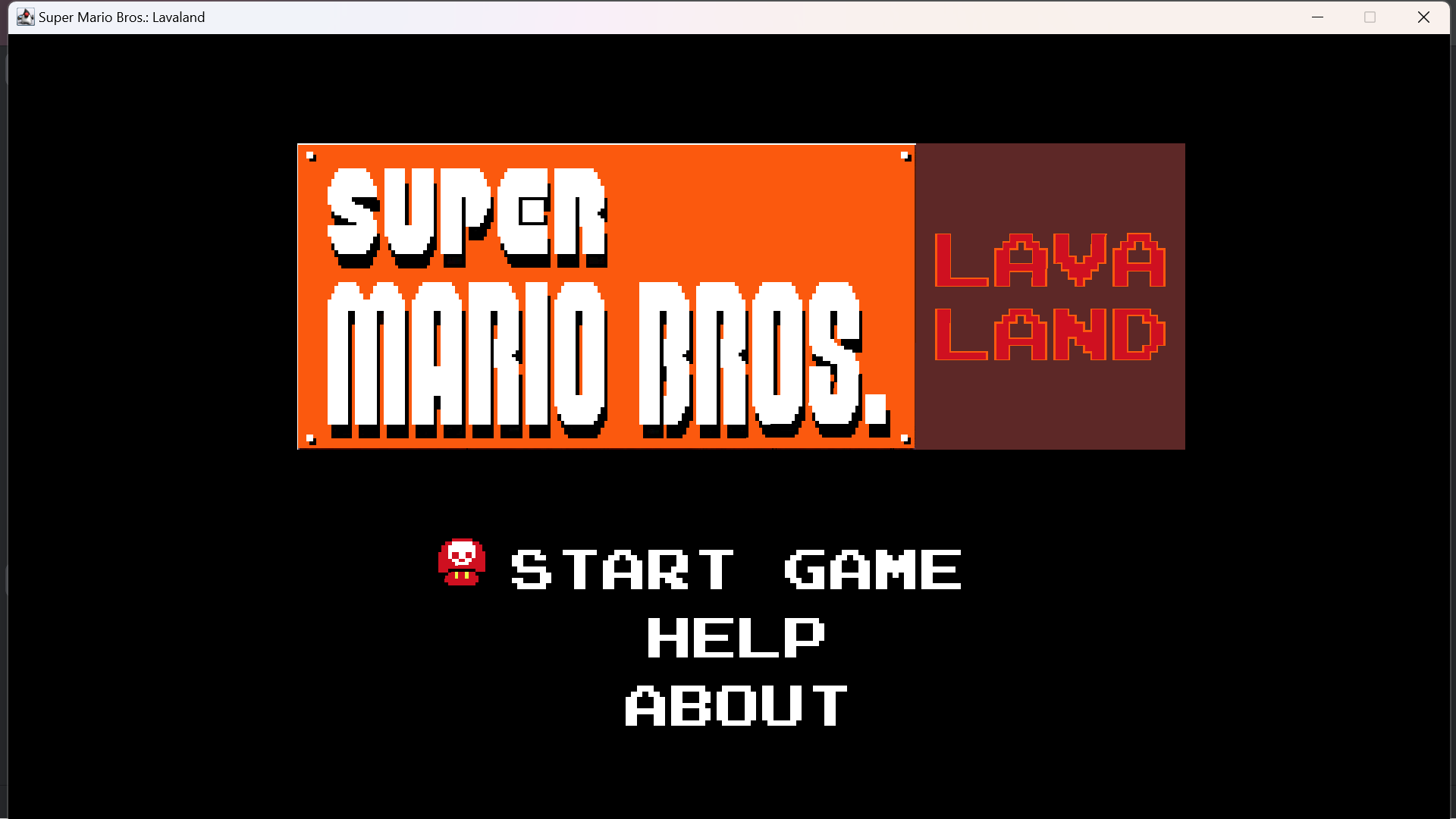
* **About the rules to play the game**
  + The objective of the game is to guide the player-controlled character, Mario, through challenges and adversaries to defeat his villain, Bowser.
  + For the duration of the game, the player is given three lives. In order to secure survival and advancement in the game, strategic preparation and quick thinking are therefore necessary.
  + Player uses the arrow keys to move Mario left or right and press the 'up' arrow key to make Mario jump.
  + There are three forms of Mario, players can utilize power-ups, such as mushrooms and flowers, by interacting with them during gameplay.
  + Players can jump on enemies to defeat them and earn coins to raise their score. Players can specifically use fireballs to defeat adversaries when Mario is in his second or third form.
  + When Mario comes into contact with an enemy without a power-up, he loses a life. Gathering 1-Up mushrooms can give players the opportunity to gain more lives.
  + Overcome obstacles and beat the boss (if applicable) to reach the end of the level. When all lives are gone, the player will see a "Game Over" screen.
* **General**
  + Mouse click:
    - PLAY: start the game.
  + Pressing button:
    - START: start the game.
    - HELP: show instructions on how to play the game.
    - ABOUT: introduce authors and applications used to create the game
    - MAP: display 2 options – Map 1 and Map 2

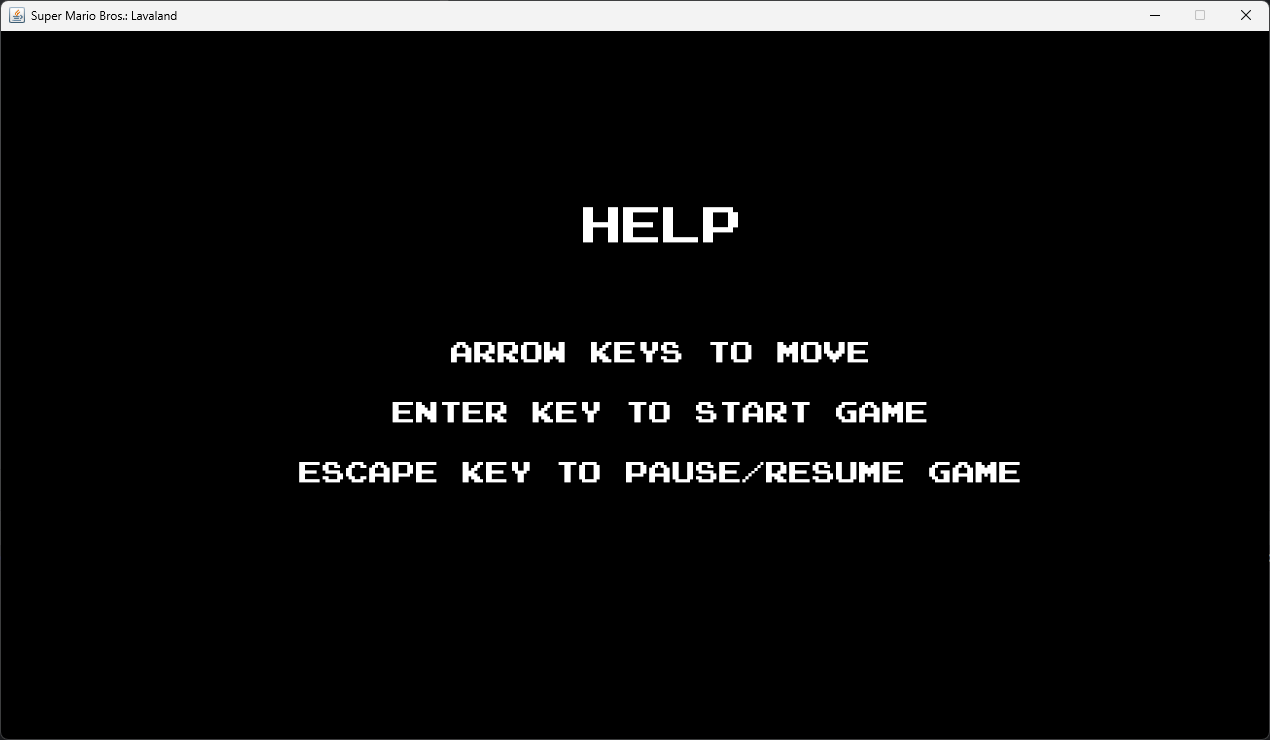
## **B.** **Design**

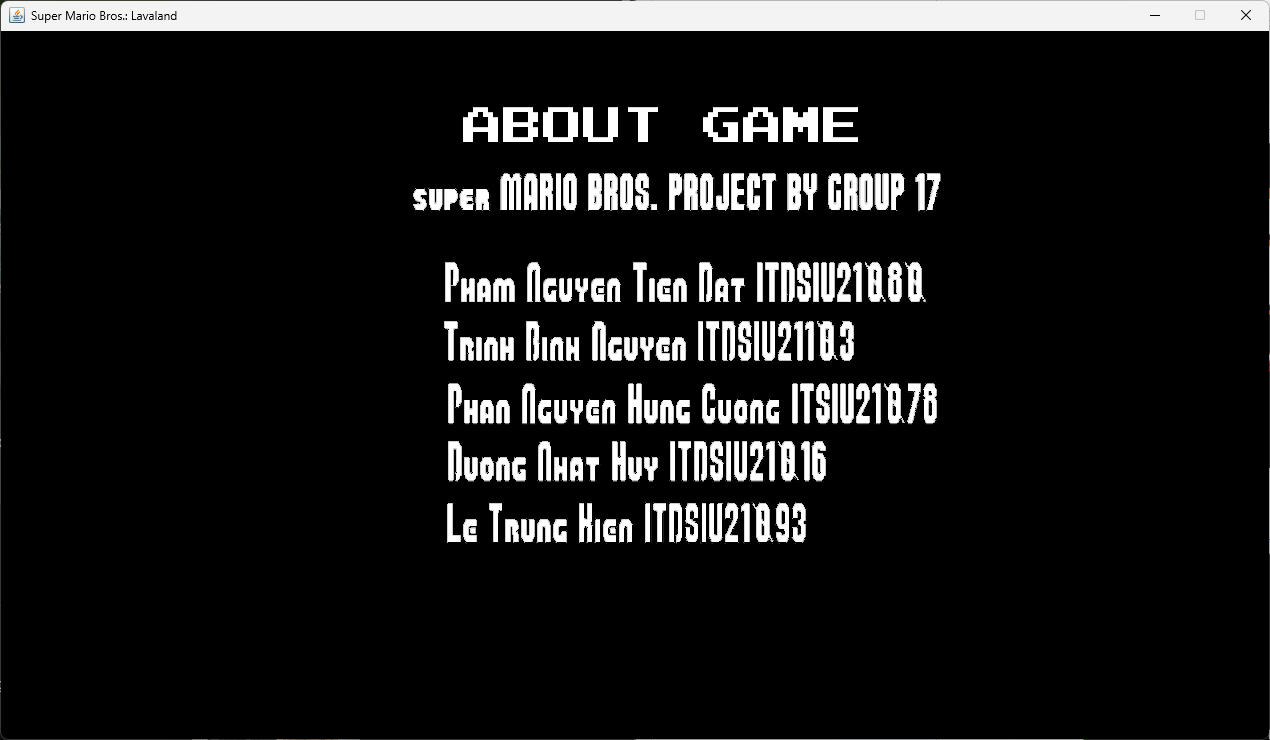
● Welcome: At the opening, the title "Super Mario Bros." is displayed. Furthermore, the title is enhanced by the use of vivid colors and a classic, two-dimensional font.

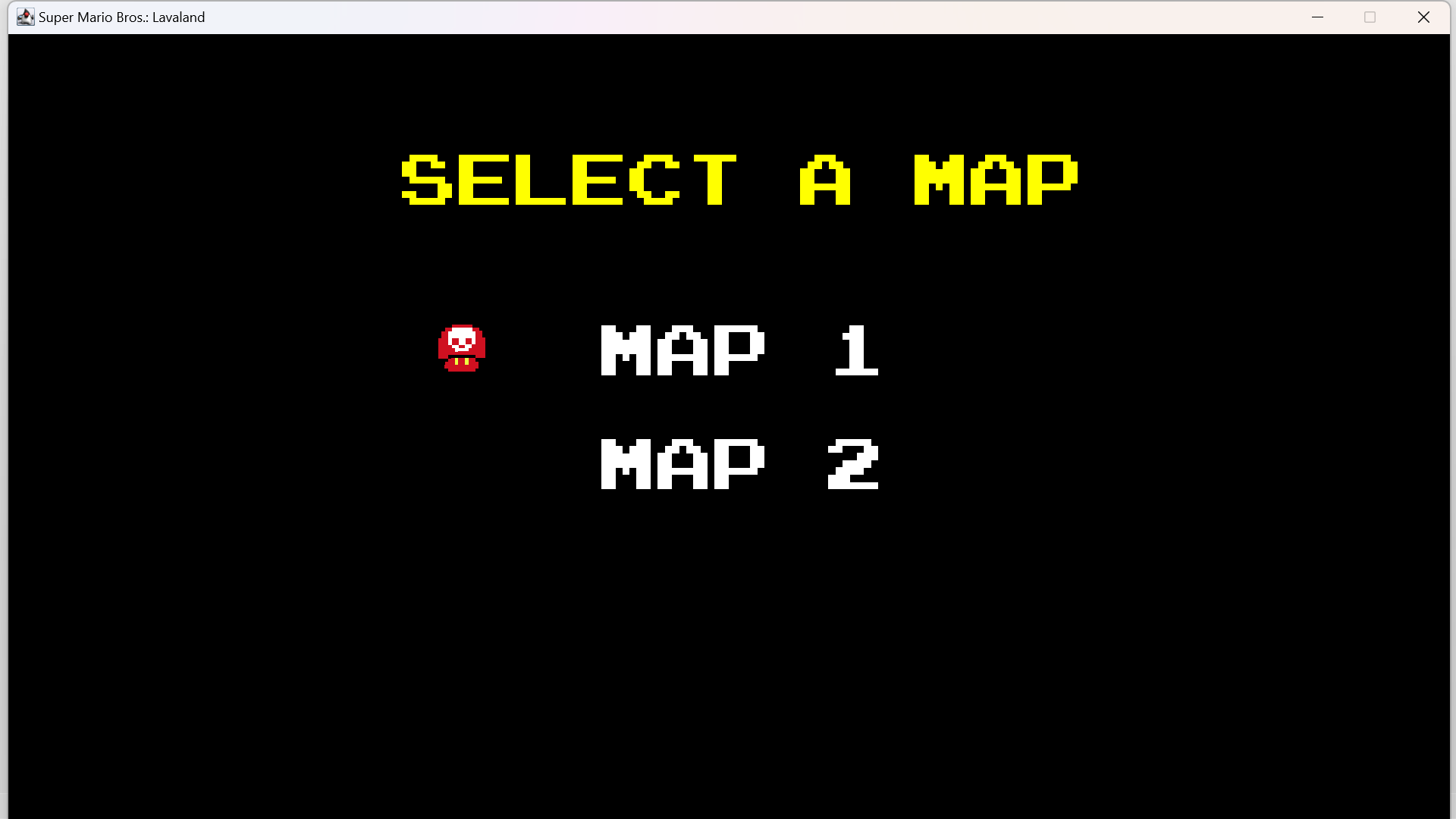
● Background: The idea for the background is to use dark colors to highlight the interface more.

We employ a basic user interface while developing our project.



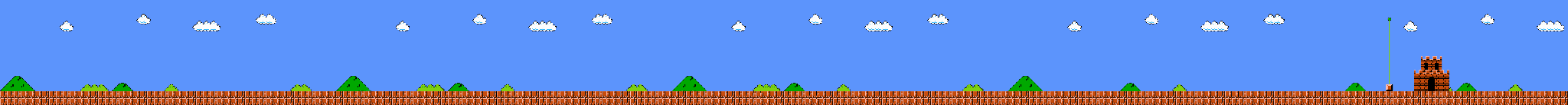






## **C. Improvements**

* **UI, Map Background and Map concept:**
* About the map background and the map concept, we changed the background into lava concept

Figure C.1. The original background

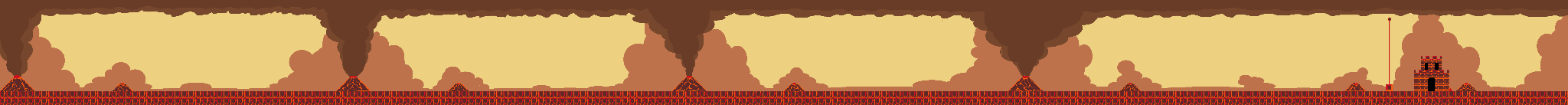
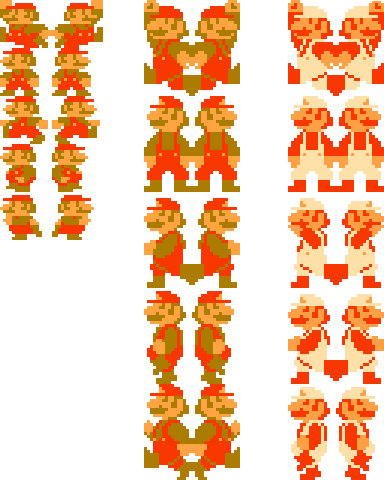


Figure C.2. The Lavaland background

* We also changed the concept of the mod to be suitable with the Lavaland concept

Figure C.3. The original and Lavaland sprite

Figure C.4. The original and Lavaland Mario Forms

* There is a user-friendly interface available for post-game and game completion situations. This redesigned UI streamlines player interactions and makes it easier to restart, continue, or return to the main screen. The objective is to improve user comfort and optimize navigation over the game's phases.



Figure C.5. The original modified UI when the game is over

Figure C.6. The modified UI when the game is over

## Figure C.7.The original UI when you finish the map

## Figure C.8. The modified UI when you finish the map

## 

## **Boss-fighting scenario**

Figure C.9. The Bower Boss in map 2

* To increase player pleasure and the level of challenge in the game, Bowser has been included as the last boss in Map 2. Bowser has 10 hearts, therefore using fireball strikes carefully is necessary for a decisive victory. His ability to shoot fireballs makes the battle more difficult and requires players to be alert. At the same time, the Flying Koopas provide another level of difficulty, requiring players to dodge Bowser's attacks as well as aerial dangers. This deliberate design decision attempts to improve the entire gaming experience by producing a complex and captivating boss encounter.
* We also add many surprise bricks in this stage to help players have more chances and power to defeat Bowser.
* **Mobs**:

## Figure C.10. The flying Koopa

Figure C.10.1 Flying Koopa

* The arrival of Flying Koopas in the air amplifies the action and gives players something more to contend with. Players must use cunning leaping techniques to defeat these opponents, which adds another level of difficulty to the entire game. This purposeful design decision ups the difficulty of the game, pushing players to hone their abilities and adjust to the ever-changing obstacles presented by the Flying Koopas in the sky.

## Figure C.11. The Spike

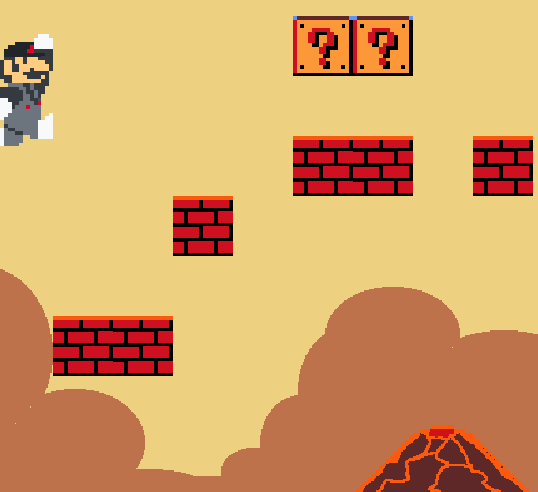
## Figure C.11.1 The Spike

## The game has also been updated with spikes included. Unlike other enemies, spikes are not immediately defeatable. The player loses a heart when they crash into a spike, so winning cannot be achieved by leaping over them. Players now face an additional difficulty as a result of this special component, which makes them more tactical in their interactions with spikes.

Figure C.12.1 The Super Mario shoot slower

Figure C.12.2 The Fire Mario shoots faster

Figure C.13.1. The Fire Mario jumps shorter

Figure C.13.2 The Super Mario jumps higher

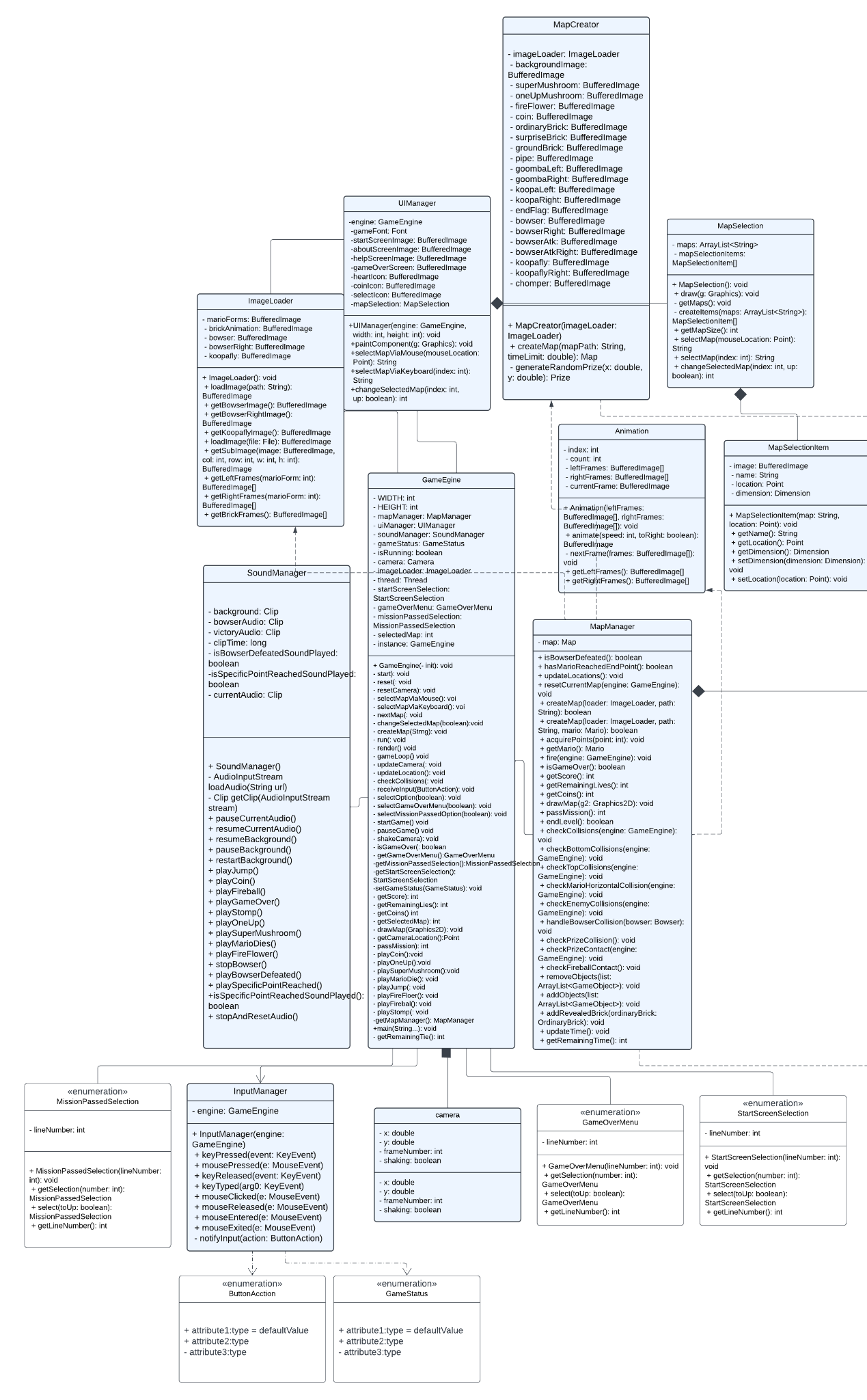
* Now, when Mario consumes super mushrooms, he can take on two forms. He will be able to jump higher and shoot slower in his first form, Super Mario form. He can shoot more quickly in the second form, Fire Mario form, but his jump height is reduced.

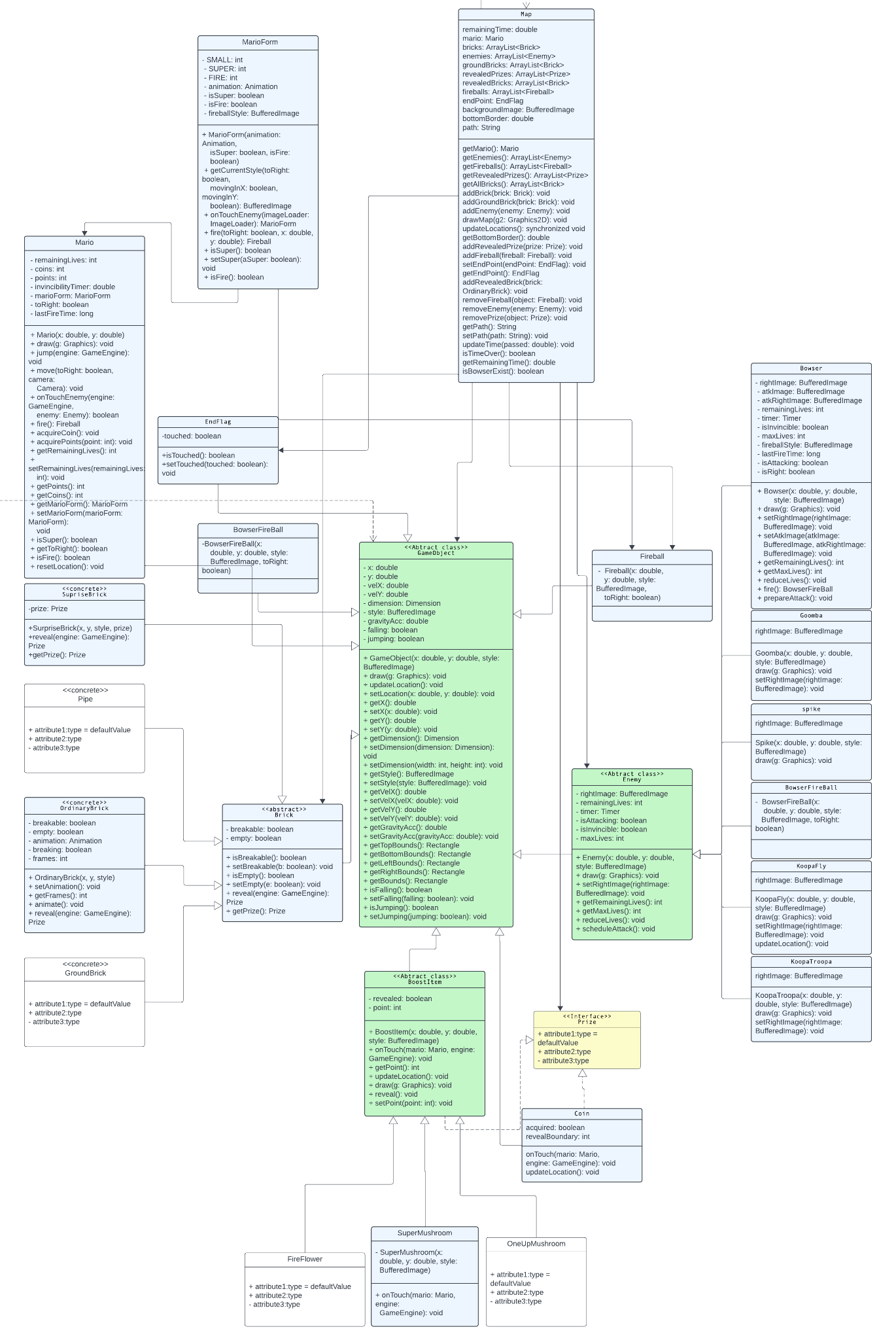
## 

## **D.** **Others**

● **UML diagram:**

* Manager package:

****

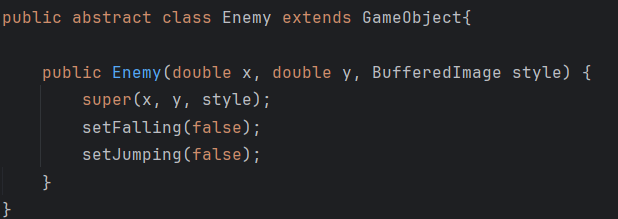
* Model package: ****

# **III. OOP CONCEPTS IMPLEMENTED**

## **A. Encapsulation**

* Encapsulation can be ensured by employing access modifiers such as private, protected, or public for class fields. By preventing unwanted access from other layers, this encapsulation improves the layer's data security and integrity. This technique is essential to object-oriented programming and improves the code's resilience and maintainability.

## **B. Polymorphism**

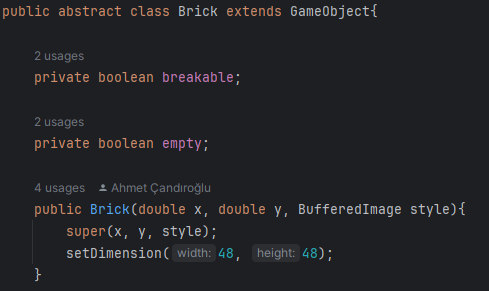


* In object-oriented programming, polymorphism is a notion that permits objects of various classes to be handled as though they were members of the same class. It lets objects behave differently according to the class to which they belong.
* The “Enemy” class’s constructor is polymorphic in nature. It invokes the “GameObject” superclass’s constructor using its parameters. This enables the “Enemy” class to be initialized with its unique properties while maintaining its status as a “GameObject”. This facilitates the inheritance of numerous variables and methods from the “GameObject” class by the “Enemy” class, simplifying the creation of other subclasses derived from the “Enemy” class.

## **C. Inheritance**

* Inheritance is a fundamental concept in object-oriented programming that allows a new class to be created based on an existing one. The Mario class's extension from the GameObject class serves as an example of this idea in the Mario video game. As a result, Bowser receives all of the GameObject class's available properties and functions. This approach encourages the reuse of code and creates a connection where Bowser is viewed as a specific type of Enemy inside the framework of the game.

## **D. Abstract**

* A class that is abstract cannot be instantiated directly; instead, it must be subclassed by another class in order for its properties to be used. Both non-abstract and abstract approaches are possible. We can have references to abstract class types, but we cannot construct instances of abstract classes.
* The picture displays a fragment of Java code that defines the abstract class {Brick}. The `breakable} and `empty} private boolean variables in this class, which extends `GameObject}, indicate the condition of a brick object in a game. The `Brick` class also has a constructor that specifies the brick object's dimensions and accepts parameters for style and position coordinates.

# **IV.** **CONCLUSION**

## **A.** **Achieved Goals**

* *Comprehensive OOP Integration:* A major focus of the game design is the efficient application of Object-Oriented Programming (OOP) concepts. Several classes were carefully designed and constructed to represent fundamental OOP principles, including Bowser, MarioForm, and Brick. Encapsulating related behaviors and data into separate objects was made possible by this strategic design, which brought to light concepts like inheritance, polymorphism, and encapsulation and abstraction. (FIX)
* *Intuitive User Interface Development:* The game's 2D visual style and interactive, user-friendly interface were expertly created utilizing Java's GUI libraries. This thoughtful design decision improves the game's aesthetic appeal and adds to the nostalgic feel, making for a more engaging gameplay experience.
* *Optimized Performance Strategies:* To guarantee a flawless gaming experience, performance optimization was the focus of a concentrated effort. Through efficient management of game objects and the application of code optimizations, the game remains fluid even in situations where the screen is cluttered with several entities, like fireballs and enemies. This methodical approach to speed optimization is essential to giving gamers a fun and uninterrupted gaming experience.( fix)

## **B.** **Limitations**

* *Limited Map Variety*: One significant drawback of our Mario game is the limited variety of maps available. The restricted number of available maps limits the overall variety and exploring potential of the game, even though the two that are currently available to players provide a satisfying but constrained experience. This restriction can make the gaming experience comparatively predictable for players looking for a wider variety of challenges.
* *Limited Game Complexity:* Our Mario game is rather simple in its current incarnation, even though it is enjoyable and functional. Some aspects that could make for a more complex and interesting gameplay experience are absent from the game. This restriction can be seen in the lack of sophisticated mechanics or other components that might raise the overall level of intricacy and depth of the gaming experience.
* *Non-Adherence to S.O.L.I.D. Principles*: In its current state, our Mario game does not fully comply with the S.O.L.I.D. design principles. This disadvantage is visible in the lack of clear separation of concerns, which might result in difficult-to-maintain-and-extend code. The absence of these principles can result in a codebase that is tightly connected and highly dependent, making it difficult to introduce new features or make changes to current ones. This limitation may have an impact on the game's general scalability and robustness, perhaps hurting the game's long-term viability and evolution. This could be a huge disadvantage for those looking for a game that adjusts and grows over time.

## **C.** **Future Enhancements**

* *Multiplayer Mode:* Enable players to collaboratively navigate the Mario world. By enabling friends or other online users to team up or compete against one another, cooperative or competitive multiplayer features could promote a vibrant social experience. In addition to increasing the excitement, this inclusion creates opportunities for cooperative gaming, cooperative accomplishments, and a more engaged gaming community.
* *New Bosses and Obstacles:* Offer an exciting new direction for future development while also introducing new difficulties and excitement. These extra opponents might have distinct attack patterns, thus players would need to adjust their tactics. Adding new barriers to the stages would increase their overall complexity and present fresh difficulties to even experienced players.
* *Shopping Items:* Allow players to convert their accrued scores into valuable items. Through the use of earned points for power-ups, character improvements, or cosmetic modifications, this system gives players the ability to personalize their gaming experiences.

# **V.** **REFERENCES**

* Candiroglu, Ahmet. “Classic Super Mario Bros. game implemented with Java for CS319 course.” GitHub, https://github.com/ahmetcandiroglu/Super-Mario-Bros. Accessed 18 Jan. 20241.
* Baker, Ben. “Super Mario’s First 14 Video Games (in Chronological Order).” TheGamer, 23 Feb. 2021, www.thegamer.com/super-mario-first-games-chronological-ranked/#mario-bros-game-amp-watch.
* Fakhroutdinov, Kirill. “Dependency in UML.” *UML Graphical Notation Overview, Examples, and Reference.*, www.uml-diagrams.org/dependency.html. Accessed 18 Jan. 2024.
* Biniasz, Kyle. “What Are OOP Concepts in Java? 4 Primary Concepts.” *Stackify*, 1 May 2023, stackify.com/oops-concepts-in-java/.
* “Code a 2D Game Engine Using Java - Full Course for Beginners.” YouTube, YouTube, 3 Jan. 2022, https://www.youtube.com/watch?v=025QFeZfeyM&feature=youtu.be. Accessed 18 Jan. 2024.

#### 