UTM

UNIVERSITI TEKNOLOGI MALAYSIA

PROJECT PROPOSAL (PHASE 1)

OBJECT ORIENTED PROGRAMMING (SECJ2154)

SEM 2 (2024/2025)

Campus Quest

LECTURER'S NAME:

DR. MUHAMMED LUQMAN

DATE:

15 May 2025

PREPARED BY (Tetrabytes) :

NO.	NAME	MATRIC NO
1.	AHMED MOHAMED KHALID ELFAKI	A23CS0286
2.	ANJUM SIDDIQUA TANVEER SIDDIQUI	A23CS0289
3.	ALWAELI AMR KHALED ABDO	A23CS4004
4.	MATHABA HASSAN MOHAMED HASSAN	A23CS4044

Table of Contents

1. Introduction

Campus Quest is a 2D platformer game inspired by Super Mario, set in a university-themed environment. The player controls a student character who navigates through various campus levels filled with enemies (e.g., assignments and quizzes), platforms, coins (merits), and obstacles. The game is built using Java Swing and AWT libraries, with custom classes for character control, enemy logic, level structure, and background music integration.

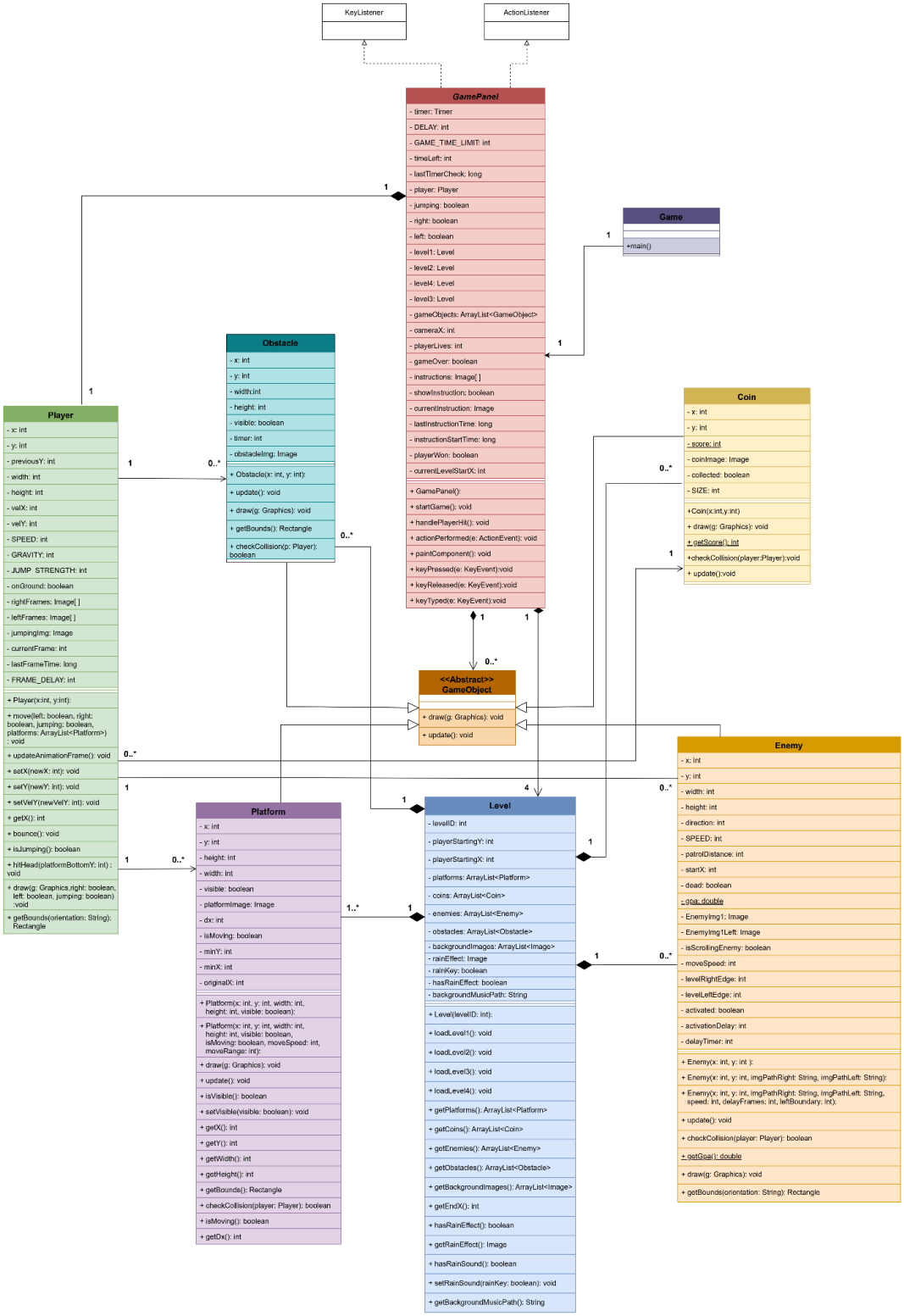
2. Problem Statement

Many educational games lack engagement and often do not relate to student experiences. There is a gap in developing relatable, gamified experiences that blend common student challenges with an interactive game environment. The goal of this project is to develop a simple, enjoyable, and educationally themed platformer game that reflects a student's campus life.

3. Objectives

- To develop a Java-based platformer game using OOP principles.
- To simulate a campus adventure through various levels and obstacles.
- To integrate multiple features like jumping, enemy collision, collectible coins, and life tracking.
- To implement background music and sound effects for an immersive experience.
- To use structured and maintainable code by organizing components into separate Java classes.

4. Project Design



4.2 Classes and Important Methods

1. GamePanel

- **Role:** The heart of the game logic and rendering.
- **Implements:** ActionListener, KeyListener
- **Key Methods:**
    - startGame(): Starts the timer and music.
    - actionPerformed(ActionEvent): Updates game state every frame.
    - paintComponent(Graphics): Draws all elements.
    - handlePlayerHit(): Handles player losing a life.
- Relationships:
    - Uses Player, Level, GameObject, and indirectly uses Platform, Coin, Enemy, Obstacle through gameObjects.

2. Player

- **Role:** Represents the main character controlled by the user.
- **Key Attributes:** x, y, dx, dy, jumping, SPEED, GRAVITY, JUMP_STRENGTH
- **Key Methods:**
    - update(AnimationFrame): Handles animation updates.
    - move(boolean, boolean, boolean, ArrayList<Platform>): Controls movement.
    - checkCollision(): Checks for interaction with platforms or coins.

3. Level

- **Role:** Defines platforms, enemies, coins, and obstacles in each level.
- **Key Attributes:** levelID, platforms, coins, enemies, obstacles
- **Key Methods:**
    - loadLevel(): Loads all objects for a level.
    - getPlatforms(), getCoins(), getEnemies(), getObstacles(): Access objects.
    - getBackground(): Gets background image for rendering.

4. GameObject (Abstract Class)

- **Role:** Base class for all objects like Enemy, Coin, Platform, Obstacle.
- **Key Methods:**
    - draw(Graphics)
    - update()
- **Subclasses**: Platform, Coin, Enemy, Obstacle

5. Enemy

- **Role:** Represents enemies (e.g., assignments or tests).
- **Key Attributes:** x, y, speed, direction, activationDelay
- **Key Methods:**
    - update(): Moves and animates enemy.
    - checkCollision(Player): Determines if it hits the player.
    - getGpa(): Used to calculate game results (GPA score).

6. Coin

- **Role:** Represents collectible coins (merits).
- **Key Attributes:** x, y, score
- **Key Methods:**
    - checkCollision(Player): Detects if collected.
    - getScore(): Returns total merits.

7. Platform

- **Role:** Represents static or moving platforms the player can walk/jump on.
- **Key Attributes:** x, y, width, height, isMoving
- **Key Methods:**
    - update(): For moving platforms.
    - checkCollision(Player)

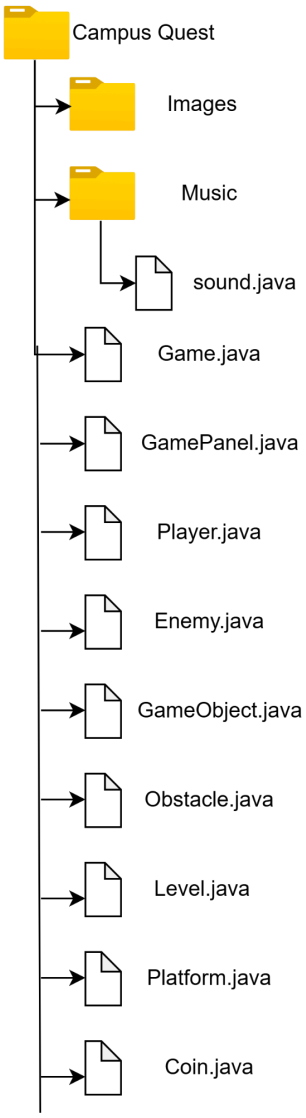
8. Obstacle

- **Role:** Represents harmful obstacles that reduce life.
- **Key Methods:**
    - checkCollision(Player)

9. Game

- Role: Contains the main() method to start the application.
- Relationship: Creates an instance of GamePanel to launch the game.

4.3 Folder Structure



5.Project Output

Figure 5.1: Level 1 starting point (Ahmed's Level)

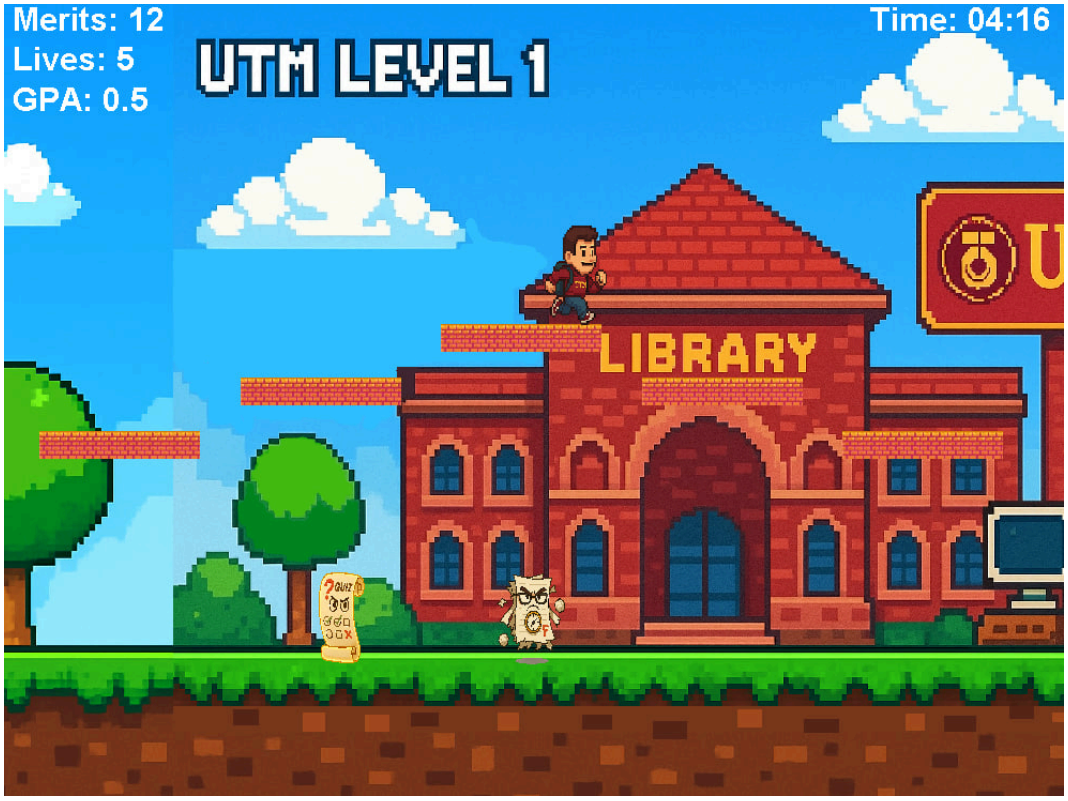


Figure 5.2: Different Backgrounds in level 1 (Ahmed's Level)

Figure 5.3: N24 Background in level3 (Mathaba's Level)

Figure 5.4: He and She Coffee Background in level 2 (Anjum's Level)

Figure 5.5: different Background in level 4 (Amr's Level)

Figure 5.6: Game over Background in level 4 (Amr’s Level)

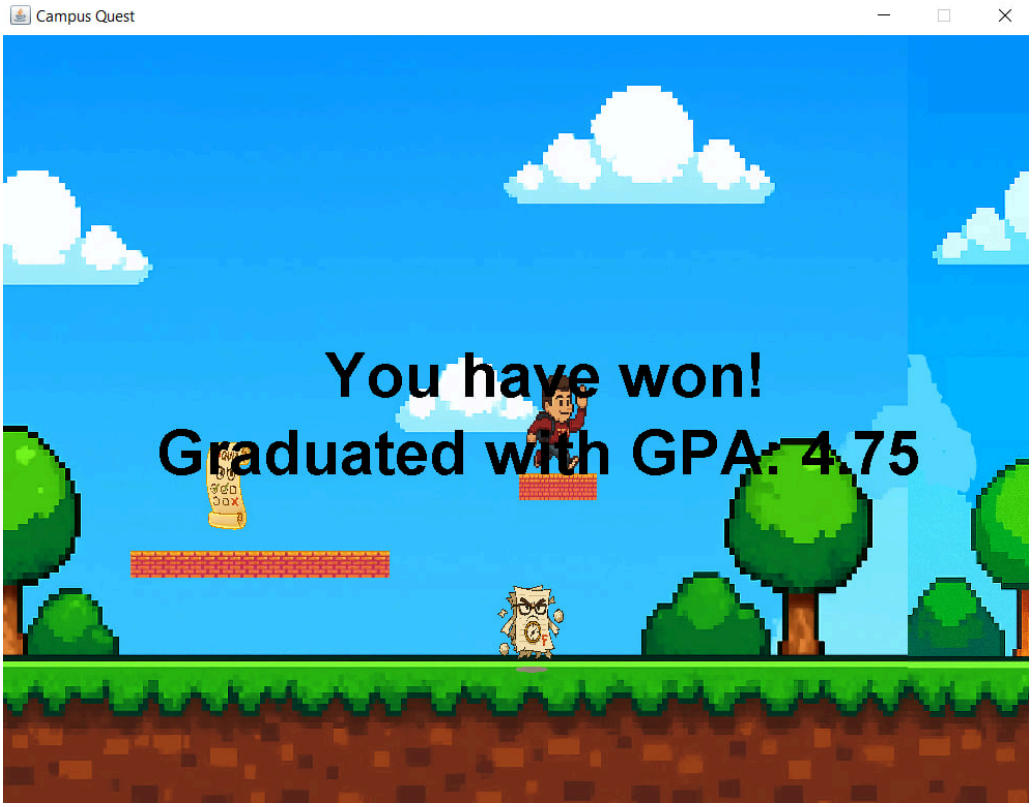


Figure 5.7: Winning announcement after finishing all levels

6.Conclusion

Campus Quest was created to give students a fun and meaningful gaming experience based on real university life at UTM. By using Java and object-oriented programming, we built a platformer game where players face challenges like assignments and quizzes while collecting merits and avoiding obstacles.

The project met its goals by combining game features such as jumping, enemies, coins, and background music, all designed with a campus theme. It also helped us improve our coding and problem-solving skills.

Overall, Campus Quest is more than just a game, it reflects the journey of a UTM student in a fun and interactive way. We hope it brings smiles to students who can relate to the ups and downs of campus life.

7.Appendices

- **Class diagram:**
  https://drive.google.com/file/d/1Br33jR4PuWK-mx8N5PJaKhYN0lomHao1/view?usp=sharing
- **Java Files (Attached):**
  - Game.java
  - GamePanel.java
  - Player.java
  - Enemy.java
  - Obstacle.java
  - Level.java
  - Coin.java
  - GameObject.java
  - Platform.java