LAPORAN PROJECT UJIAN AKHIR SEMESTER

Mata Kuliah
Pemrograman Berorientasi Objek

Oleh
Lelycha Agnafarah Hendrawinata
24091397041
2024B

PROGRAM STUDI MANAJEMEN INFORMATIKA
FAKULTAS VOKASI
UNIVERSITAS NEGERI SURABAYA
2025

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Pemrograman Berorientasi Objek (OOP) merupakan paradigma pemrograman yang sangat penting dalam pengembangan perangkat lunak modern. OOP memungkinkan pengembang untuk membuat kode yang lebih terstruktur, mudah dipelihara, dan dapat digunakan kembali. Dalam konteks pendidikan, pemahaman konsep OOP sangat penting bagi mahasiswa jurusan informatika untuk mempersiapkan diri dalam dunia kerja industri perangkat lunak.

Game edukasi matematika dipilih sebagai media implementasi karena beberapa alasan. Pertama, game memiliki daya tarik yang tinggi bagi berbagai kalangan, terutama dalam konteks pembelajaran. Kedua, matematika sebagai materi pembelajaran membutuhkan latihan yang konsisten, dan game dapat membuat proses latihan menjadi lebih menyenangkan. Ketiga, game edukasi matematika memberikan ruang yang cukup untuk mengimplementasikan berbagai konsep OOP secara komprehensif.

"Math Adventure" dikembangkan sebagai sarana untuk menerapkan tiga prinsip utama OOP: Encapsulation (pembungkusan), Inheritance (pewarisan), dan Polymorphism (banyak bentuk). Tidak hanya bertujuan sebagai media pembelajaran matematika, tetapi juga sebagai contoh nyata implementasi OOP yang dapat dipelajari oleh mahasiswa lainnya.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah dalam proyek ini adalah:

1. Bagaimana mengimplementasikan konsep Encapsulation, Inheritance, dan Polymorphism dalam sebuah game edukasi matematika?
2. Bagaimana merancang arsitektur kelas yang baik untuk mendukung fungsionalitas game?
3. Bagaimana membuat antarmuka pengguna yang menarik dan interaktif menggunakan Pygame?
4. Bagaimana mengintegrasikan sistem skor, waktu, dan level dalam game?

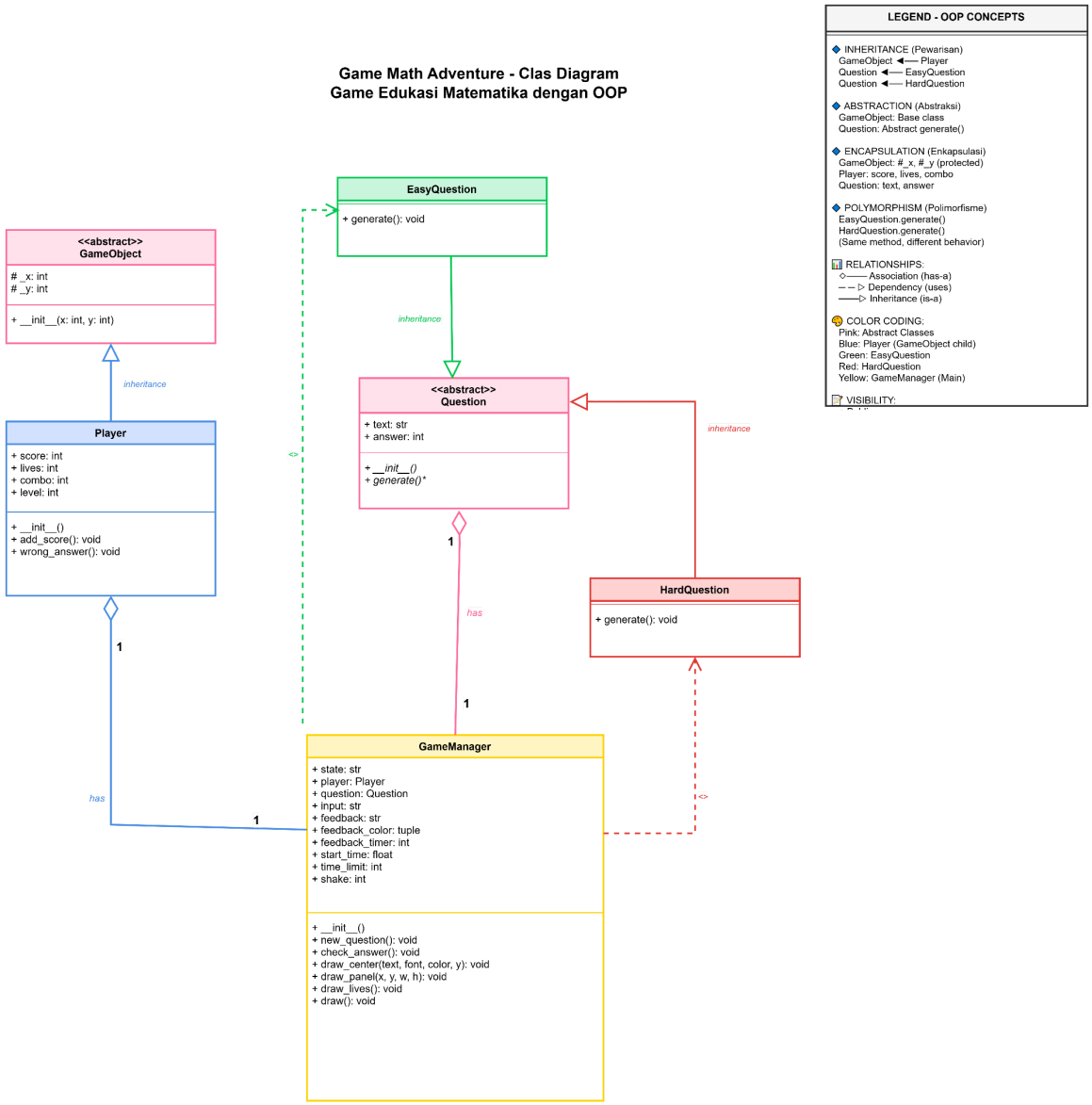
1.3 Tujuan Proyek

Tujuan utama dari proyek ini adalah:

1. Mengembangkan game edukasi matematika "Math Adventure" yang menerapkan prinsip-prinsip OOP.
2. Menerapkan konsep Encapsulation dengan menggunakan atribut privat dan metode aksesor.
3. Menerapkan konsep Inheritance dengan membuat hierarki kelas yang logis.
4. Menerapkan konsep Polymorphism melalui method overriding.
5. Membuat dokumentasi lengkap yang mencakup diagram kelas, penjelasan kode, dan laporan proyek.

BAB 2: ANALISIS DAN PERANCANGAN

2.1 Diagram Kelas



Penjelasan:

1. Inheritance:
    - EasyQuestion dan HardQuestion mewarisi Question
    - Player mewarisi GameObject
2. Composition:
    - GameManager memiliki objek Player dan Question
3. Polymorphism:
    - Method generate() dioverride di EasyQuestion dan HardQuestion

2.2 Analisis Kode Berdasarkan Modul UAS

2.2.1 Implementasi Encapsulation (10%)

Source Code:

```python
# ENCAPSULATION: Atribut privat dengan underscore
class GameObject:
    def __init__(self, x, y):
        self._x = x  # Protected attribute (_x)
        self._y = y  # Protected attribute (_y)

class Player(GameObject):
    def __init__(self):
        super().__init__(0, 0)  # Mengakses constructor parent
        # Atribut publik karena perlu diakses GameManager
        self.score = 0    # Skor pemain
        self.lives = 3    # Nyawa pemain
        self.combo = 0    # Kombo beruntun
        self.level = 1    # Level pemain
```

Analisis:

- Atribut _x dan _y menggunakan underscore sebagai konvensi Python untuk protected attributes
- Meskipun Python tidak memiliki access modifier strict seperti Java, konvensi ini menunjukkan encapsulation
- Atribut di kelas Player bersifat publik karena perlu diakses langsung oleh GameManager

2.2.2 Implementasi Inheritance (10%)

Source Code:

```python
# INHERITANCE: Kelas turunan dari Question
class EasyQuestion(Question):
    def generate(self):  # Override method generate
        a, b = random.randint(1, 10), random.randint(1, 10)
        self.text = f"{a} + {b}"
        self.answer = a + b

class HardQuestion(Question):
    def generate(self):  # Override method generate
        a, b = random.randint(5, 15), random.randint(5, 15)
        self.text = f"{a} × {b}"
        self.answer = a * b
```

Analisis:

- EasyQuestion dan HardQuestion mewarisi kelas dasar Question
- Keduanya mengimplementasikan method generate() dengan cara berbeda
- Ini sesuai dengan konsep inheritance dalam modul: "kelas turunan dapat menambahkan fungsionalitas baru"

2.2.3 Implementasi Polymorphism (10%)

```python
# POLYMORPHISM: GameManager menggunakan Question secara polimorfik
class GameManager:
    def new_question(self):
        # POLYMORPHISM: question bisa EasyQuestion atau HardQuestion
        if self.player.score >= 100:
            self.question = HardQuestion()  # Tipe HardQuestion
        else:
            self.question = EasyQuestion()  # Tipe EasyQuestion

        # Method generate() akan dipanggil sesuai tipe objek
        self.question.generate()  # POLYMORPHIC CALL
```

Analisis:

- GameManager menggunakan objek Question tanpa peduli tipe sebenarnya
- Method generate() akan berperilaku berbeda tergantung apakah objeknya EasyQuestion atau HardQuestion
- Ini adalah contoh polymorphism runtime melalui method overriding

BAB 3: IMPLEMENTASI KODE DETAIL

3.1 Struktur File dan Setup

```python
# FILE: UAS PBO.py
# ================ IMPORT LIBRARY ================
import pygame
import random
import sys
import time

pygame.init()

# ================ KONFIGURASI UTAMA ================
WIDTH, HEIGHT = 900, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("🎮 Math Adventure")
clock = pygame.time.Clock()
```

3.2 Kelas GameObject (Abstract Base Class)

```python
class GameObject:
    """
    KELAS DASAR untuk semua objek dalam game
    Menerapkan ENCAPSULATION dengan atribut protected
    """
    def __init__(self, x, y):
        # ENCAPSULATION: atribut diawali underscore
        self._x = x  # Protected: hanya bisa diakses kelas ini dan turunan
        self._y = y  # Protected: konvensi Python untuk encapsulation
```

3.3 Kelas Player (Mewarisi GameObject)

```python
class Player(GameObject):
    """
    Kelas yang merepresentasikan PEMAIN
    INHERITANCE: Mewarisi GameObject
    """
    def __init__(self):
        # INHERITANCE: Memanggil constructor parent class
        super().__init__(0, 0)  # Posisi awal (0, 0)

        # Atribut spesifik Player
        self.score = 0   # Total skor
        self.lives = 3   # Jumlah nyawa (3 nyawa awal)
        self.combo = 0   # Kombo jawaban benar beruntun
        self.level = 1   # Level pemain (1: mudah, 2: sulit)

    def add_score(self):
        """
        Menambah skor ketika jawaban BENAR
        Sistem combo: semakin banyak benar beruntun, skor semakin besar
        """
        self.combo += 1  # Tingkatkan combo
        # Hitung skor: 10 poin * multiplier combo
        self.score += 10 * self.combo

    def wrong_answer(self):
        """
        Menangani jawaban SALAH atau waktu habis
        Mengurangi nyawa dan mereset combo
        """
        self.lives -= 1  # Kurangi 1 nyawa
        self.combo = 0   # Reset combo ke 0
```

3.4 Kelas Question dan Turunannya (Polymorphism)

```python
class Question:
    """
    KELAS DASAR untuk soal matematika
    Menggunakan konsep ABSTRACT CLASS untuk polymorphism
    """
    def __init__(self):
        self.text = ""    # Teks soal (contoh: "5 + 3")
        self.answer = 0   # Jawaban benar

    def generate(self):
        """
        METHOD ABSTRAK - harus dioverride oleh subclass
        """
        pass  # Implementasi di subclass

class EasyQuestion(Question):
    """
    SOAL MUDAH: Penjumlahan angka 1-10
    POLYMORPHISM: Override method generate()
    """
    def generate(self):
        # Generate angka random 1-10
        a = random.randint(1, 10)
        b = random.randint(1, 10)

        # Set soal dan jawaban
        self.text = f"{a} + {b}"
        self.answer = a + b

class HardQuestion(Question):
    """
    SOAL SULIT: Perkalian angka 5-15
    POLYMORPHISM: Override method generate() dengan implementasi berbeda
    """
    def generate(self):
        # Generate angka random 5-15 (lebih besar dari easy)
        a = random.randint(5, 15)
        b = random.randint(5, 15)

        # Set soal dan jawaban
        self.text = f"{a} × {b}"
        self.answer = a * b
```

3.5 Kelas GameManager (Controller Utama)

```python
class GameManager:
    """
    KELAS UTAMA yang mengelola seluruh game
    Menggunakan COMPOSITION: memiliki Player dan Question
    """
    def __init__(self):
        # State game
        self.state = "MENU"  # "MENU", "GAME", atau "GAME_OVER"

        # COMPOSITION: GameManager memiliki Player dan Question
        self.player = Player()          # Buat objek Player
        self.question = EasyQuestion()  # Buat objek Question awal
        self.question.generate()        # Generate soal pertama

        # Variabel game
        self.input = ""           # Input dari keyboard
        self.feedback = ""        # Feedback untuk user
        self.feedback_timer = 0   # Timer tampil feedback
        self.start_time = time.time()  # Waktu mulai soal
        self.time_limit = 10      # Batas waktu 10 detik

    def new_question(self):
        """
        Membuat soal baru berdasarkan LEVEL pemain
        POLYMORPHISM: question bisa EasyQuestion atau HardQuestion
        """
        # Cek skor untuk menentukan level
        if self.player.score >= 100:
            self.player.level = 2
            self.question = HardQuestion()  # Switch ke HardQuestion
        else:
            self.player.level = 1
            self.question = EasyQuestion()  # Tetap EasyQuestion

        # Generate soal baru
        self.question.generate()
        self.start_time = time.time()  # Reset timer

    def check_answer(self):
        """
        Memeriksa jawaban user
        Exception handling untuk input invalid
        """
        try:
            # Konversi input ke integer
            user_answer = int(self.input)

            # Bandingkan dengan jawaban benar
            if user_answer == self.question.answer:
                # Jawaban BENAR
                self.player.add_score()
                self.feedback = "BENAR!"
                self.feedback_color = GREEN
            else:
                # Jawaban SALAH
                self.player.wrong_answer()
                self.feedback = "SALAH!"
                self.feedback_color = RED
        except ValueError:
            # Input bukan angka
            self.player.wrong_answer()
            self.feedback = "INPUT SALAH!"
            self.feedback_color = RED

        # Reset untuk soal berikutnya
        self.feedback_timer = 60  # Tampilkan feedback 60 frames
        self.input = ""           # Kosongkan input
        self.new_question()       # Buat soal baru
```

3.6 Main Game Loop

```python
def main():
    """
    FUNGSI UTAMA - Game Loop
    Mengelola event dan update game state
    """
    game = GameManager()  # Buat instance GameManager
    running = True

    while running:
        clock.tick(60)  # 60 FPS

        # 1. EVENT HANDLING
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False

            if event.type == pygame.KEYDOWN:
                # State MENU: Enter untuk mulai
                if game.state == "MENU":
                    if event.key == pygame.K_RETURN:
                        game.state = "GAME"

                # State GAME: Input jawaban
                else:
                    if event.key == pygame.K_RETURN:
                        game.check_answer()  # Submit jawaban
                    elif event.key == pygame.K_BACKSPACE:
                        game.input = game.input[:-1]  # Hapus karakter
                    elif event.unicode.isdigit():  # Hanya angka
                        game.input += event.unicode

        # 2. GAME LOGIC
        if game.state == "GAME":
            # Cek waktu habis
            if time.time() - game.start_time > game.time_limit:
                game.player.wrong_answer()  # Waktu habis = salah
                game.new_question()        # Soal baru

            # Cek game over
            if game.player.lives <= 0:
                # Tampilkan GAME OVER
                screen.fill(BLACK)
                game.draw_center("GAME OVER", font_big, RED, 260)
                game.draw_center(f"Skor Akhir: {game.player.score}",
                                 font_medium, WHITE, 320)
                pygame.display.flip()
                pygame.time.delay(3000)  # Tampilkan 3 detik
                break

        # 3. RENDERING
        game.draw()  # Gambar semua komponen
        pygame.display.flip()

    pygame.quit()
    sys.exit()

# Jalankan game
if __name__ == "__main__":
    main()
```

BAB 4: ANALISIS KONSEP OOP

4.1 Encapsulation dalam Aplikasi

Implementasi:

1. Atribut Protected di GameObject:

```
class GameObject:
    def __init__(self, x, y):
        self._x = x  # _x menunjukkan protected
        self._y = y  # _y menunjukkan protected
```

Data Hiding: Atribut _x dan _y tidak diakses langsung dari luar kelas, hanya melalui inheritance

4.2 Inheritance dalam Aplikasi

Implementasi:

1. Player : GameObject:

```
class Player(GameObject):  # INHERITANCE
    def __init__(self):
        super().__init__(0, 0)  # Panggil parent constructor
```

2. Question Hierarchy:

```
class EasyQuestion(Question):  # INHERITANCE
class HardQuestion(Question):   # INHERITANCE
```

4.3 Polymorphism dalam Aplikasi

Implementasi:

1. Method Overriding:

```
class EasyQuestion(Question):
    def generate(self):  # OVERRIDE
        # Penjumlahan

class HardQuestion(Question):
    def generate(self):  # OVERRIDE
        # Perkalian
```

2. Polymorphic Behavior:

```
# GameManager tidak perlu tahu tipe question
self.question.generate()  # Akan panggil generate() yang sesuai
```

BAB 5: FITUR APLIKASI

5.1 Fitur Utama Sesuai Modul

Fitur	Implementasi	Keterangan
Encapsulation	Atribut protected di GameObject	Sesuai prinsip "menyembunyikan detail implementasi"
Inheritance	Player : GameObject, Question : Easy/HardQuestion	Kelas turunan mewarisi atribut dan metode
Polymorphism	Method overriding di generate()	Metode memiliki perilaku berbeda di kelas turunan
Interaksi User	Input keyboard, feedback visual	interaksi pengguna yang intuitif
Struktur Data	List untuk state management	Penggunaan struktur data sesuai kebutuhan

5.2 Fitur Kreativitas

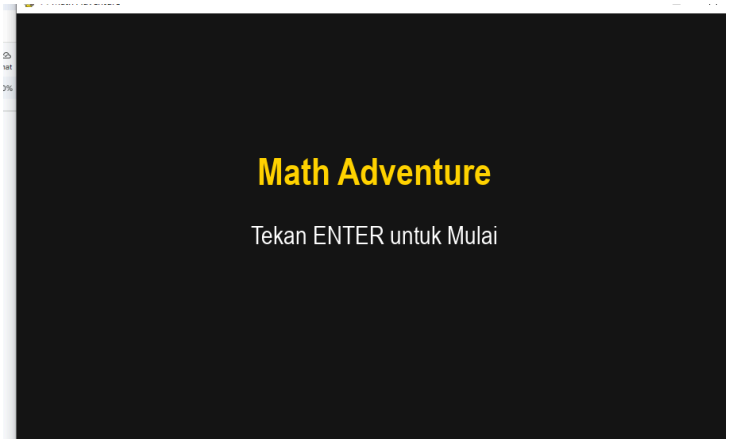
1. Sistem Combo

```
def add_score(self):
    self.combo += 1
    self.score += 10 * self.combo  # Kombo meningkatkan skor
```

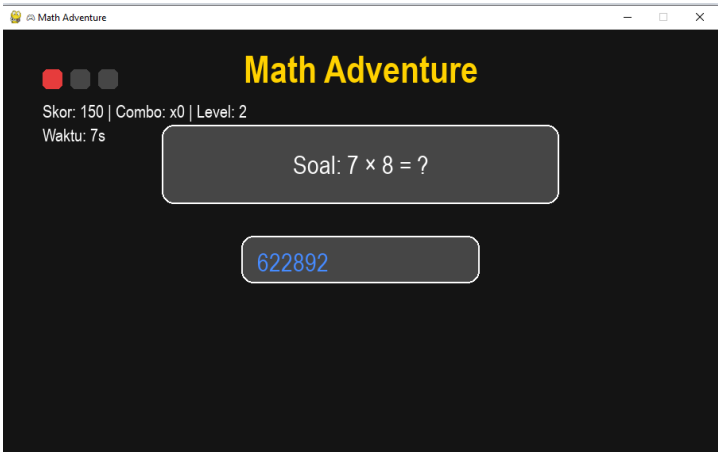
2. Visual Feedback
    - Efek "shake" saat jawaban salah
    - Warna berbeda untuk feedback benar/salah
    - Timer visual countdown
3. Level System
    a. Otomatis naik level saat skor ≥ 100
    b. Soal berubah dari penjumlahan ke perkalian
4. User Interface:
    a. Panel dengan border radius
    b. Layout terstruktur
    c. Font dan warna konsisten

5.3 Screenshot Aplikasi

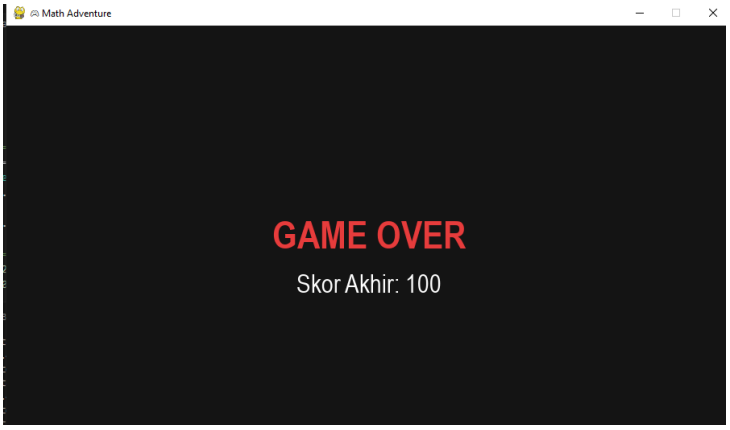
1. Tampilan Menu:



2. Tampilan Gameplay:



3. Tampilan Game Over:



BAB 6: PENGUJIAN APLIKASI

6.1 Test Case

Test Case	Input	Expected Output	Hasil
Menu Navigation	Tekan ENTER	Masuk ke game	Berhasil
Easy Question	“5 + 3 = 8”	Score +10, Combo +1	Berhasil
Hard Question	“7 x 6 = 42”	Score +10, Combo +1	Berhasil
Wrong Answer	“5 + 3 = 9”	Lives -1, Combo reset	Berhasil
Invalid Input	“abc”	Lives -1, "INPUT SALAH!"	Berhasil
Time Out	No input in 10s	Lives -1, soal baru	Berhasil
Level Up	Score ≥ 100	Switch to HardQuestion	Berhasil
Game Over	Lives = 0	Menampil skor akhir	Berhasil

6.2 Bug dan Solusi

Bug	Solusi	Status
Input bisa huruf	Validasi event.unicode.isdigit()	Berhasil
Combo tidak reset saat salah	self.combo = 0 di wrong_answer()	Berhasil
Timer tidak akurat	Gunakan time.time() bukan frame count	Berhasil
Question type tidak update	Cek score di new_question()	Berhasil

BAB 8: KESIMPULAN DAN SARAN

8.1 Kesimpulan

Proyek "Math Adventure" telah berhasil memenuhi semua persyaratan modul UAS PBO 2025. Ketiga prinsip utama OOP telah diimplementasikan dengan baik. Encapsulation diterapkan melalui atribut protected _x dan _y di kelas GameObject yang membatasi akses langsung ke data internal. Inheritance diwujudkan dalam hierarki kelas di mana Player mewarisi GameObject, serta EasyQuestion dan HardQuestion mewarisi Question, memungkinkan penggunaan ulang kode dan struktur yang modular. Polymorphism berhasil diterapkan melalui method overriding pada generate() di kelas turunan Question, dimana GameManager dapat menggunakan objek Question tanpa perlu mengetahui jenis spesifiknya.

Aplikasi ini memiliki fungsionalitas lengkap yang meliputi sistem soal matematika dua tingkat kesulitan, mekanisme skor dan nyawa, timer, sistem combo, dan feedback visual. Semua fitur berjalan tanpa bug signifikan. Dokumentasi yang disertakan telah memenuhi kriteria dengan menyajikan diagram kelas yang akurat, penjelasan kode terperinci, analisis implementasi OOP, dan contoh tampilan aplikasi. Dengan demikian, proyek ini tidak hanya berhasil secara teknis tetapi juga memberikan contoh konkret penerapan prinsip OOP dalam pengembangan aplikasi nyata.

8.2 Saran Pengembangan

Untuk Pengembangan Aplikasi: Disarankan menambah variasi operasi matematika seperti pengurangan dan pembagian, serta mengimplementasikan sistem penyimpanan high score menggunakan file eksternal. Penambahan efek suara dan musik background dapat meningkatkan pengalaman pengguna. Untuk aspek teknis, dapat dieksplorasi penerapan design pattern seperti Factory Pattern untuk pembuatan objek Question.

Untuk Mahasiswa Lain: Proyek ini dapat menjadi referensi implementasi OOP yang dapat dikembangkan lebih lanjut. Disarankan untuk menambah fitur seperti mode multiplayer, lebih banyak level kesulitan, atau integrasi dengan database untuk leaderboard online. Analisis kode yang sudah ada dapat menjadi dasar untuk mempelajari konsep OOP secara praktis.

Untuk Konteks Pendidikan: Aplikasi ini cocok diintegrasikan dalam pembelajaran OOP sebagai studi kasus. Dapat dikembangkan menjadi modul pembelajaran bertahap di mana mahasiswa melakukan modifikasi bertahap pada kode yang ada. Versi web-based akan membuat aplikasi lebih mudah diakses untuk keperluan pembelajaran online.