|  |  |  |
| --- | --- | --- |
| Nama : | * Satria Andhika Indriawan | / 5223600072 |
|  | * Dzamirzi Ilham Hidayat | / 5223600062 |

Tugas :

|  |
| --- |
| Soal : 1 |
| Apa perbedaan struktur antara Observer dan Mediator di kasus ini? |
| Jawab : |
| Observer Pattern :  Player.h   * Player secara eksplisit tahu siapa saja yang mendengar event. * Semua komponen (UI, SoundSystem, ScoreSystem) **terikat langsung** ke Player melalui interface IObserver.     Player.cpp |
| Mediator Pattern:  Player.h   * Player **tidak tahu siapa yang mendengar event**. * Listener (UI, Sound, Score) **terdaftar di mediator**, dan hanya mediator yang tahu siapa saja mereka. * Lebih fleksibel dan terpisah.     Player.cpp    EventMediator.h |
|  |
| Soal : 2 |
| Dalam skala **game kecil**, mana yang lebih praktis? Kenapa? |
| Jawab : |
| **Observer Pattern lebih praktis untuk game kecil.** Karena:   * Struktur sederhana dan langsung. * Kode lebih sedikit, cukup simpan pointer observer dan panggil notify(). * Tidak perlu tambahan kelas Mediator.   Cocok saat hanya ada sedikit event dan listener. |
| Soal : 3 |
| Dalam skala **game besar** (dengan banyak event), mana yang lebih scalable? Kenapa? |
| Jawab : |
| **Mediator Pattern lebih scalable untuk game besar.** Karena:   * Event handling lebih terpusat dan fleksibel. * Player (atau objek lain) tidak harus tahu siapa yang menangani event. * Listener bisa didaftarkan ke banyak jenis event tanpa modifikasi ke sender.   Cocok saat sistem makin kompleks dengan banyak jenis event & listener. |
| Soal : 4 |
| Sebutkan satu risiko yang harus diwaspadai saat memakai masing-masing pattern. |
| Jawab : |
| |  |  | | --- | --- | | Observer : | Tight coupling: sender terlalu tahu tentang listener, bisa jadi sulit di-maintain jika jumlah listener bertambah | | Mediator : | Mediator bisa jadi terlalu kompleks (God Object) jika tidak dikelola dengan baik, apalagi jika semua event masuk ke satu tempat | |