

Penerapan Closure dalam Simulasi Permainan dengan Decorator untuk Logging

Randa Andriana Putra¹, Uliano Wilyam Purba², Tria Yunanni³, Daris Samudra⁴, Nasywa Nur Afifah⁵

122450083¹, 122450098², 122450062³, 122450102⁴, 122450125⁵

Program Studi Sains Data Institut Teknologi Sumatera

Jl. Terusan Ryacudu, Way Huwi, Kec. Jatiagung, Kabupaten Lampung Selatan, Lampung 35365

Email: randa.122450083@student.itera.ac.id¹, uliano.122450098@student.itera.ac.id², tria.122450062@student.itera.ac.id³, daris.122450102@student.itera.ac.id⁴, nasywa.122450125@student.itera.ac.id⁵

PENDAHULUAN

Seiring berkembangnya waktu game atau permainan adalah hal yang sangat lumrah dijumpai bahkan kita sendiri pun sering memainkannya, dalam permainan tersebut terkadang kita menemui log atau catatan aktivitas yang dilakukan selama bermain. Misal seperti catatan membunuh musuh, skor yang sudah didapati level yang sudah dilalui dan lain-lain. Logging tersebut sangatlah penting dalam permainan karena memudahkan player dalam bermain tanpa perlu mengingat hal yang sudah dilakukan karena sudah tercatat dalam permainan. Hubungan dengan dunia pemrograman ialah penerapan closure beserta decorator. Maka dari itu kami pada jurnal ini akan membahas penerapan closure beserta decorator untuk membuat logging atau catatan aktivitas dalam sebuah simulasi permainan sederhana.

METODE

A. Closure

Closure dalam pemrograman ialah kombinasi antara fungsi dan lexical scoping dimana fungsi tersebut dapat dinyatakan. Sederhananya, closure mampu memungkinkan fungsi untuk mengakses variabelnya diluar lingkup itu sendiri. Ini dapat terjadi karena closure menggunakan konsep lingkungan dimana struktur kode menentukan lingkup suatu variabel, bukan saat dieksekusi.

Manfaat penting closure dalam pemrograman diantara lain: Meningkatkan modularitas: Dengan pembukaan, pemrogram dapat membuat fungsi yang lebih kecil dan lebih fokus, yang meningkatkan keterbacaan kode dan modularitas. Memudahkan enkapsulasi: Dengan menggunakan closure, pemrogram dapat menyembunyikan detail implementasi fungsi, meningkatkan enkapsulasi dan keamanan kode. Memungkinkan pemrograman fungsional memerlukan pembukaan, yang memungkinkan pembuatan program yang lebih deklaratif dan mudah dipahami. (Harold, A. 1996)

B. Decorator

Decorator dalam pemrograman ialah teknik pemrograman yang memodifikasikan fungsi tanpa mengubah definisi aslinya. Cara decorator bekerja ialah membungkus fungsi yang ada dan menambahkan fungsi baru sebelum atau sesudah fungsi tersebut dilakukan.

Berikut manfaat penting dalam decorator ialah :

Meningkatkan modularitas: Decorator memungkinkan Anda untuk menambahkan fungsionalitas baru ke fungsi tanpa harus mengubah definisi aslinya, yang meningkatkan modularitas dan keterbacaan kode.

Memudahkan enkapsulasi: Decorator meningkatkan enkapsulasi dan keamanan kode dengan memungkinkan Anda untuk menyembunyikan detail implementasi fungsionalitas baru.

Mempromosikan reuse: Dekorator dapat digunakan kembali untuk menambah fitur yang sama ke berbagai fungsi, meningkatkan penggunaan kode. (Brett, S. 2019)

PEMBAHASAN

Pada pembahasan laporan kali ini kami menggunakan dekorator `logging_decorator` untuk mencatat setiap panggilan fungsi yang didekorasi dengan pesan log yang sesuai. Fungsi ini mengambil fungsi asli (`func`) sebagai argumen, dan mengembalikan fungsi wrapper yang menambahkan perilaku logging sebelum dan sesudah memanggil fungsi asli.

```
def logging_decorator(func):
    def wrapper(*args):
        result = func(*args)
        print(f"[LOG] {func.__name__} dipanggil dengan argumen {args}")
        return result
    return wrapper
```

Kemudian dalam fungsi `create_player`, kita akan mendefinisikan dua fungsi: `play_game` dan `get_score`, akan digunakan dalam simulator permainan. Fungsi `play_game` berfungsi untuk meng-update skor player sesuai dengan jumlah poin yang diterima, sementara `get_score` dipakai untuk memperoleh skor player pada saat ini. Implementasi closure terjadi pada fungsi `play_game`, di mana kita menerapkan variabel lokal skor yang terdeklarasikan pada fungsi induk `create_player`. Walaupun fungsi `play_game` itu sendiri tidak menerima atau mengembalikan variabel skor, fungsi ini memiliki kontrol langsung terhadap variabel ini karena jangkauan leksikal dari closure, yang memungkinkannya untuk memperbarui nilai skor pada setiap pemanggilan fungsi. Sehingga dengan demikian, closure mengizinkan kita untuk menyimpan dan menjaga status permainan (skor player) di antara pemanggilan fungsi lain tanpa perlu menyimpannya secara global atau memberikannya sebagai sebuah parameter. Hal ini akan meningkatkan efisiensi kode dan mempermudah pemeliharaan.

```
def create_player(name):
    score = 0

    @logging_decorator
    def play_game(points):
        nonlocal score
        score += points
        print(f"{name} mendapatkan {points} poin! Total skor: {score}")

    @logging_decorator
    def get_score():
        return score

    return play_game, get_score
```

Kemudian, setelah selesai mendefinisikan fungsi-fungsi ini, kita akan membuat dua player, player1 dan player2, dengan menggunakan fungsi create_player. Setiap pemain memiliki fitur untuk memainkan game dalam permainan (play_game) dan untuk mendapatkan skor saat ini (get_score). Pada tingkat selanjutnya, player akan memainkan beberapa tahap permainan untuk setiap player dengan menggunakan fungsi play_game, dan mencetak skor akhir setiap player dengan menggunakan fungsi get_score. Output yang dihasilkan adalah sebuah hasil yang akan menunjukan aktivitas permainan dan skor akhir dari setiap player

```
#Membuat pemain
player1_play, player1_score = create_player("Player 1")
player2_play, player2_score = create_player("Player 2")

#Memainkan permainan
player1_play(10)
player1_play(20)
player2_play(15)

#Mendapatkan skor
print("Skor akhir Player 1:", player1_score())
print("Skor akhir Player 2:", player2_score())
```

KESIMPULAN

Penggunaan closure dan dekorator pada perluasan simulasi game sangat bermanfaat dalam literasi kode, modularitas, dan fungsionalitas. Closure digunakan ke dalam fungsi play_game untuk mengarsipkan dan memperbaiki skor player di antara pemanggilan fungsi-fungsi lainnya. Ini dapat menaikan kemudahan literasi kode dan juga mempermudah dalam perawatan. Logging_decorator digunakan untuk membuat catatan pada setiap pemanggilan fungsi dengan pesan log yang sesuai. Fungsi ini diggunakan untuk mambantu proses pengontrolan peningkatan permainan. Dengan menerapkan closures dan decorators, kode game bersih, modular, dan mudah dirawat. Karena hal ini akan meningkatkan kualitas dan kemampuan perangkat lunak serta meningkatkan proses pengembangan secara menyeluruh.

REFERENSI

Harold, A. 1996. *Structure and Interpretation of Computer Programs 2nd ed.* MIT Press

Brett, S. 2019. *Effective Python: 90 Specific Ways to Write Better Python*. Addison-Wesley Professional