Penerapan Closure dalam Simulasi Permainan dengan Decorator untuk Logging

Randa Andriana Putra[1], Uliano Wilyam Purba[2], Tria Yunanni[3], Daris Samudra[4], Nasywa Nur Afifah[5]
122450083[1], 122450098[2], 122450062[3], 122450102[4], 122450125[5]
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.id[1], uliano.122450098@student.itera.ac.id[2], tria.122450062@student.itera.ac.id[3], daris.122450102@student.itera.ac.id[4], nasywa.122450125@student.itera.ac.id[5]

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 pemain 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 leksikal (lexical scope) 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:

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 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.

```python
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 pemain sesuai dengan jumlah poin yang diterima, sementara get_score dipakai untuk memperoleh skor pemain pada saat ini. Implementasi closure terjadi pada fungsi play_game, di mana kita menerapkan variabel lokal skor yang terdeklarasi pada fungsi induk create_player. Walaupun fungsi play_game itu sendiri tidak menerima atau mengembalikkan 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 pemain) 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
```

Selanjutnya, setelah pendefinisian fungsi-fungsi ini, kita akan membuat dua pemain, pemain1 dan pemain2, dengan memanfaatkan fungsi create_player. Setiap pemaian memiliki fitur untuk memainkan permainan (play_game) dan untuk memperoleh skor saat ini (get_score). Pada tahap selanjutnya, kita akan memainkan beberapa babak permainan untuk setiap pemain dengan memanfaatkan fungsi play_game, dan mencetak skor akhir setiap pemain dengan menggunakan fungsi get_score. Output yang dihasilkan adalah sebuah hasil yang memperlihatkan aktivitas permainan dan skor akhir dari setiap pemain.

```
#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

Pemanfaatan closure dan dekorator pada pengembangan simulasi game sangat bermanfaat dalam keterbacaan kode, modularitas, dan fungsionalitas. Closure diterapkan ke dalam fungsi play_game guna meyimpan dan memperbarui skor Pemain di antara pemanggilan fungsi-fungsi lainnya. Ini bisa meningkatkan kemudahan pembacaan kode dan juga mempermudah dalam pemeliharaan. Logging_decorator dipakai untuk membuat catatan pada setiap pemanggilan fungsi dengan pesan log yang sesuai. Fungsinya adalah untuk mambantu proses pemantauan perkembangan permainan. Dengan penerapan closures dan decorators, kode game bersih, modular, dan mudah dirawat. 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