Simulasi Belanja *Online* Sederhana Menggunakan *Higher-Order-Function*

Deva Anjani Khayyuninafsyah (122450014), Nabiilah Putri Karnaia (122450029),
Chevando Daffa Pramanda (122450095), Novelia Adinda (122450104),
Rafly Prabu Darmawan (122450140)

Program Studi Sains Data, Fakultas Sains, Institut Teknologi Sumatera
Jl. Terusan Ryacudu, Way Huwi, Kec. Jatiagung, Kabupaten Lampung Selatan, Lampung 35365

Email:
deva.122450014@student.itera.ac.id, nabiilah.122450029@student.itera.ac.id,
chevando.122450095@student.itera.ac.id, novelia.122450104@student.itera.ac.id,
rafly.122450140@student.itera.ac.id

1. Pendahuluan

Belanja *online* telah menjadi pilihan utama bagi banyak orang dalam memenuhi kebutuhan sehari-hari di era teknologi yang semakin berkembang karena kemudahan akses, beragamnya pilihan produk, dan kecepatan transaksi. Di balik begitu digemarinya perkembangan teknologi ini, tentu cara kerjanya di belakang layar menjadi hal yang patut diapresiasi. Hal itu dikarenakan terdapat suatu ide yang membantu mempermudah kompleksitas dalam pembuatannya, yaitu konsep *higher-order-function*.

Konsep *Higher-Order-Function* (HOF) membantu *programmer* membuat kode yang lebih bersih, efektif, dan efisien dengan memungkinkannya untuk menggunakan fungsi sebagai parameter atau mengembalikan fungsi sebagai nilai. HOF dapat digunakan untuk mengelola berbagai elemen data seperti *filtering*, *sorting*, dan *mapping* produk dalam pembuatan aplikasi belanja *online*.

Berdasarkan uraian di atas, maka dilakukan pembuatan simulasi belanja *online* sederhana menggunakan *higher-order-function* untuk mengetahui konsep dari suatu *platform* belanja *online* dengan sesederhana mungkin serta memahami konsep dari *higher-order-function* itu sendiri. Dari topik ini, diharapkan pembaca dapat memahami konsep tersebut dan menerapkannya dalam pengembangan suatu aplikasi mulai dari yang sesederhana mungkin.

2. Metode

Dalam permasalahan kali ini, digunakan metode *higher-order-function* dengan menerapkan beberapa fungsi. Fungsi-fungsi tersebut antara lain fungsi *reduce()*, fungsi *map()*, dan fungsi lambda yang dijelaskan sebagai berikut.

2.1. Fungsi *reduce()*

2.1.1. Pada '*tambah_ke_keranjang(keranjang, barang)*' bertanggung jawab atas penambahan barang ke dalam keranjang belanja. Dengan fungsi *reduce* tersebut, barang berhasil dimasukkan ke dalam keranjang dengan langkah-langkah yang terstruktur.

2.1.2. Pada '*total_kalkulasi(keranjang)*' akan menghitung total biaya belanjaan yang ada dalam keranjang. Dengan menggunakan fungsi *reduce*, harga barang dijumlahkan untuk menghasilkan total belanjaan.

2.1.3. Pada '*kalkulasi_total_diskon(keranjang)*' akan menghitung total diskon yang diterapkan pada semua barang di dalam keranjang belanja. Dengan pendekatan *reduce* ini, total diskon berhasil dihitung dengan efisien.

2.1.4. Pada '*reduce(tambah_ke_keranjang, iterator, [])*' akan menambahkan setiap elemen dari iterator hasil operasi *map* ke dalam keranjang belanja.

2.2. Fungsi *map()*

Pada '*map(lambda x: barang_diskon(*x), list_barang)*' digunakan untuk menerapkan fungsi '*barang_diskon*' pada setiap *tuple* dalam '*list_barang*'.

2.3. Fungsi *lambda()*

Pada '*map(lambda x: barang_diskon(*x), list_barang)*' akan membentuk fungsi ringkas yang menerima *tuple* sebagai argumen, dan kemudian menjalankan fungsi '*barang_diskon*' dengan elemen-elemen dari *tuple* tersebut sebagai argumen.

3. Pembahasan

```python
from functools import reduce

# membuat elemen barang dengan diskon
def barang_diskon(nama, harga, diskon):
  harga_diskon = harga * (1 - diskon)
  return {"nama": nama, "harga": harga, "diskon": diskon}

# menambahkan elemen barang ke keranjang belanja
def tambah_ke_keranjang(keranjang, barang):
  return keranjang + [barang]

# menghitung total harga belanjaan
def total_kalkulasi(keranjang):
  return reduce(lambda total, barang: total + barang["harga"], keranjang, 0)

# menampilkan elemen barang di keranjang
def tampilkan_keranjang(keranjang, formatter):
  print("Barang di keranjang:\n")
  for barang in keranjang:
    print(formatter(barang))

# mendapatkan harga setelah diskon dari sebuah elemen barang
def get_harga_diskon(barang):
  return barang["harga"] * (1 - barang["diskon"])
```

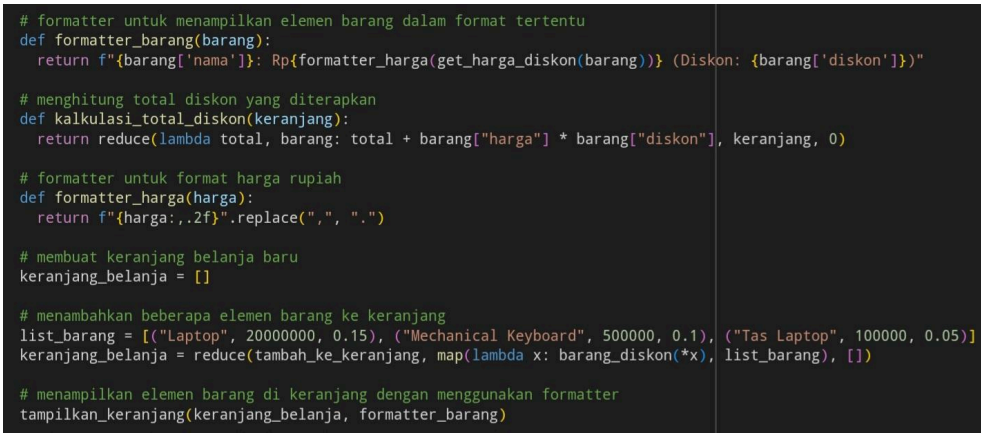
Gambar 1

Pada Gambar 1, terdapat kode dengan fungsi '*reduce*' dari modul '*functools*' serta beberapa fungsi yang didefinisikan sendiri, diantaranya sebagai berikut :

a. Fungsi '*barang_diskon*' yang menerima tiga argumen, yaitu nama, harga, dan diskon, untuk menghitung harga suatu barang setelah diskon dan mengembalikan nilai yang berisi tentang informasi barang tersebut.

b. Fungsi '*tambah_ke_keranjang*' menerima dua argumen, yaitu keranjang dan barang, yang berfungsi menambahkan item produk ke keranjang belanja dan mengembalikan daftar keranjang yang diperbarui.

c. Fungsi '*total_kalkulasi*' terdapat fitur fungsi *reduce* serta lambda untuk menjumlahkan harga barang dalam daftar keranjang belanja, kemudian mengembalikan nilai total harga barang tersebut.
d. Fungsi '*tampilkan_keranjang*' menerima dua argumen, yaitu keranjang dan *formatter*, berfungsi untuk menampilkan daftar elemen barang dalam keranjang dan mencetaknya dengan menggunakan fungsi *formatter* yang telah ditentukan.
e. Fungsi '*get_harga_diskon*' akan menghitung harga barang setelah diskon dan mengembalikan nilai dari harga tersebut.



```
# formatter untuk menampilkan elemen barang dalam format tertentu
def formatter_barang(barang):
  return f"{barang['nama']}: Rp{formatter_harga(get_harga_diskon(barang))} (Diskon: {barang['diskon']})"

# menghitung total diskon yang diterapkan
def kalkulasi_total_diskon(keranjang):
  return reduce(lambda total, barang: total + barang["harga"] * barang["diskon"], keranjang, 0)

# formatter untuk format harga rupiah
def formatter_harga(harga):
  return f"{harga:,.2f}".replace(",", ".")

# membuat keranjang belanja baru
keranjang_belanja = []

# menambahkan beberapa elemen barang ke keranjang
list_barang = [("Laptop", 20000000, 0.15), ("Mechanical Keyboard", 500000, 0.1), ("Tas Laptop", 100000, 0.05)]
keranjang_belanja = reduce(tambah_ke_keranjang, map(lambda x: barang_diskon(*x), list_barang), [])

# menampilkan elemen barang di keranjang dengan menggunakan formatter
tampilkan_keranjang(keranjang_belanja, formatter_barang)
```

Gambar 2

Kemudian pada Gambar 2 terdapat deretan kode lanjutan dari Gambar 1. Dari gambar di atas terdapat beberapa fungsi di antaranya :

a. Fungsi '*formatter_barang*' berfungsi memformat setiap elemen barang dalam format tertentu yang akan menjadi sebuah string yang mencakup nama barang, harga setelah diskon, dan persentase diskon yang diterapkan.
b. Fungsi '*kalkulasi_total_diskon*' akan menghitung total diskon yang diterapkan pada seluruh barang yang ada dalam keranjang. Lalu menggunakan fungsi *reduce* yang menerima parameter keranjang serta fungsi lambda untuk mengalikan harga dengan diskon yang nantinya akan dijumlahkan menjadi total diskon dari setiap barang di dalam keranjang belanja.
c. Fungsi '*formatter_harga*' akan memformat harga menjadi format rupiah dengan parameter harga. Parameter harga akan diformat ke dalam dua desimal lalu dengan metode replace akan menggantikan setiap koma (,) menjadi titik (.) agar hasilnya menjadi mata uang Indonesia, yaitu Rupiah (Rp).

Kemudian, dilanjutkan dengan membuat keranjang belanja baru yang *list*nya masih kosong. Lalu membuat variabel baru '*list_barang*' untuk menambahkan beberapa elemen barang ke dalam keranjang, serta membuat variabel 'keranjang_belanja' yang menggunakan fungsi *map* dengan penerapan fungsi '*barang_diskon*' ke dalam elemen '*list_barang*', yang akan menghasilkan *list* barang dengan harga setelah diskon

kemudian dan juga menggunakan fungsi *reduce* untuk menambahkan elemen barang ke dalam '*keranjang_belanja*'.

```
# menghitung total harga belanjaan
total_harga = total_kalkulasi(keranjang_belanja)
print(f"\nTotal: Rp{formatter_harga(total_harga)}")

# menghitung total diskon yang diterapkan
total_diskon = kalkulasi_total_diskon(keranjang_belanja)
print(f"Total Diskon: Rp{formatter_harga(total_diskon)} \n")

# menghitung harga akhir yang harus dibayar
harga_bayar = total_harga - total_diskon
print(f"Jumlah yang harus dibayar: Rp{formatter_harga(harga_bayar)}.")
```

```
Barang di keranjang:

Laptop: Rp17.000.000.00 (Diskon: 0.15)
Mechanical Keyboard: Rp450.000.00 (Diskon: 0.1)
Tas Laptop: Rp95.000.00 (Diskon: 0.05)

Total: Rp20.600.000.00
Total Diskon: Rp3.055.000.00

Jumlah yang harus dibayar: Rp17.545.000.00.
```

Gambar 3

Pada kode lanjutan gambar di atas menampilkan variabel '*total_harga*' untuk menghitung total harga belanjaan dari barang-barang yang ada dalam keranjang setelah diterapkannya diskon serta menampilkannya. Lalu variabel '*total_diskon*' akan menghitung total diskon yang diterapkan dari keranjang belanja serta menampilkannya. Pada kode terakhir berfungsi untuk menghitung jumlah harga yang harus dibayar setelah dikurangkannya total harga dengan total diskon.

Output dari deretan kode tersebut menampilkan daftar elemen barang di keranjang belanja, total harga belanja, total diskon harga, dan jumlah harga yang harus dibayar. Pada tampilan elemen daftar barang di keranjang, menunjukkan harga setiap barang setelah diskon serta diskon yang berlaku untuk setiap barangnya. Kemudian menampilkan juga total harga barang di keranjang sebelum dipotong diskon dan total diskon yang didapatkan dari setiap barang. Lalu yang terakhir menampilkan jumlah yang harus dibayar, menunjukkan total harga yang akan dibayar setelah total harga sebenarnya dikurangkan dengan total diskon yang didapatkan.

4. Kesimpulan

Penggunaan konsep *Higher-Order-Function* (HOF) dalam kehidupan sehari-hari salah satunya dapat kita terapkan melalui Simulasi Belanja *Online*. HOF sendiri adalah fungsi yang menerima fungsi lain menjadi parameternya dan mengembalikan fungsi lain sebagai *output*nya.

Dalam artikel ini kami menerapkan 3 jenis fungsi dasar, yaitu *reduce()*, *map()*, dan lambda() untuk pembuatan program simulasi belanja *online* sederhana. Pertama-tama program akan didefinisikan sebagai beberapa fungsi-fungsi seperti menambahkan barang, menghitung total harga dan diskon. Lalu membuat keranjang dan menambahkan ke dalamnya beberapa barang kemudian akan menampilkannya.

Output yang dihasilkan akan menampilkan daftar barang serta harga setelah didiskon dan besarnya diskon yang kemudian akan mencetak total harga belanjaan, total diskon dan jumlah yang harus dibayar.

5. Daftar Pustaka

Haverbeke, Marijn. (2014). "Eloquent JavaScript: A Modern Introduction to Programming." No Starch Press.

Lipman, Alex. (2019). "Functional Programming in JavaScript: How to improve your JavaScript programs using functional techniques." Manning Publications.

Morrison, Kyle. (2018). "Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries." O'Reilly Media.

Freeman, Eric, and Robson, Elisabeth. (2015). "Programming Elixir." Pragmatic Bookshelf.