Modularisasi Algoritma Shortest Path pada Perangkat Lunak KIRI Menggunakan Strategy Pattern

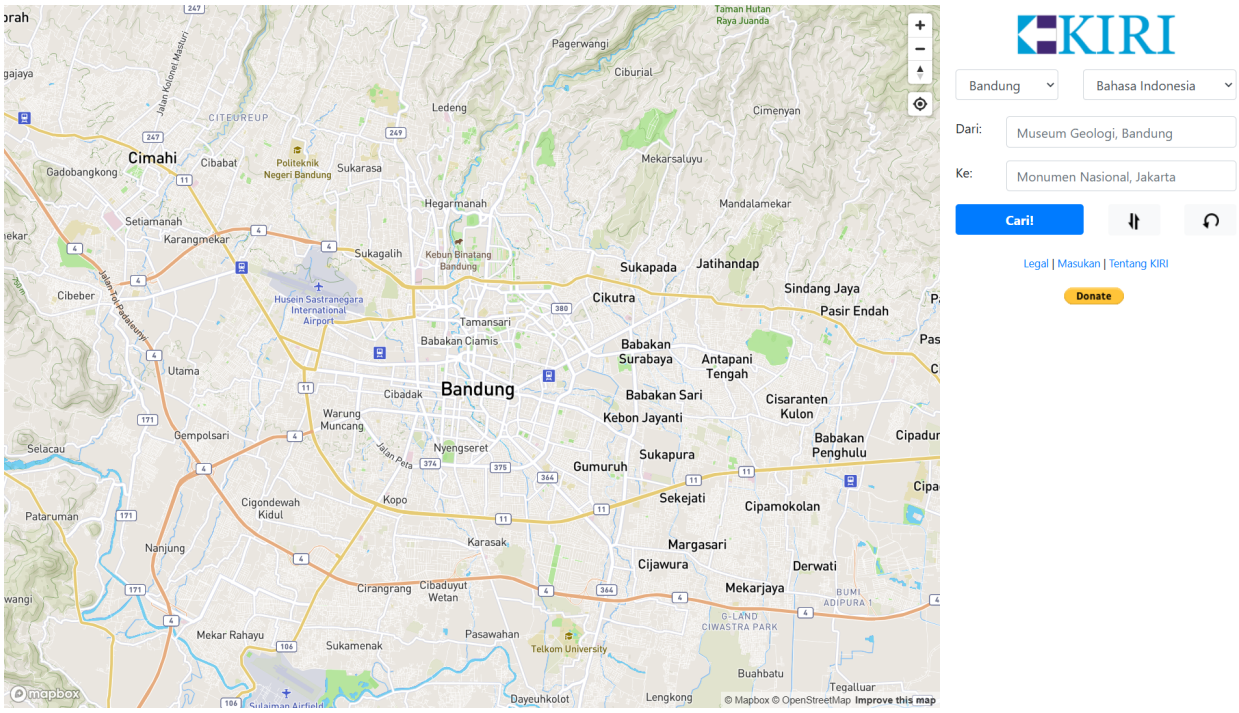
Muhammad Aldi Rivandi–6182001029

1 Data Tugas Akhir

Pembimbing utama/tunggal: **Pascal Alfadian Nugroho, M.Comp.**
Pembimbing pendamping: -
Kode Topik : **PAN5702BCS**
Topik ini sudah dikerjakan selama : **1** semester
Pengambilan pertama kali topik ini pada : Semester **57 - Ganjil 24/25**
Pengambilan pertama kali topik ini di kuliah : **Tugas Akhir 1**
Tipe Laporan : **B -** Dokumen untuk reviewer pada presentasi dan **review Tugas Akhir 1**

2 Latar Belakang

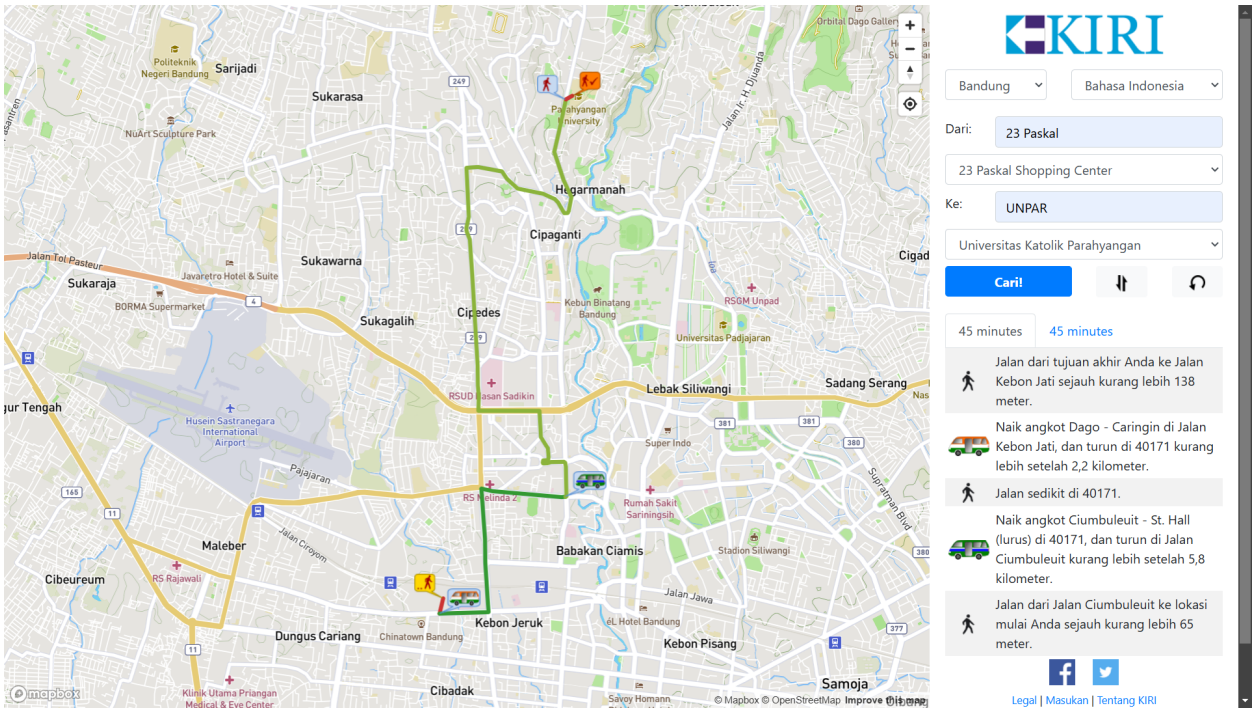
Perangkat lunak KIRI[1] (lihat Gambar 1) adalah perangkat lunak berbasis *web* yang dirancang untuk membantu pengguna menemukan rute perjalanan ketika menggunakan angkot untuk di Bandung serta TransJakarta dan Commuterline untuk di DKI Jakarta. Pada perangkat lunak KIRI, pengguna dapat memasukkan titik awal perjalanan dan titik tujuan. KIRI kemudian akan mencarikan berbagai alternatif rute yang bisa digunakan untuk mencapai tujuan tersebut.



Gambar 1: Tampilan halaman perangkat lunak KIRI

[1] https://projectkiri.id/

KIRI akan memberikan informasi mengenai langkah-langkah yang harus ditempuh oleh pengguna yang akan bepergian dari suatu tempat ke tempat tujuannya, mulai dari seberapa jauh pengguna harus berjalan untuk menaiki angkot yang bersangkutan, di mana pengguna harus naik atau turun angkot tersebut, seberapa jauh lagi pengguna harus berjalan sampai ke lokasi tujuan, dan seberapa lama estimasi waktu perjalanan yang akan ditempuh (lihat Gambar 2).



Gambar 2: Tampilan perangkat lunak KIRI, setelah menerima masukan

Arsitektur aplikasi KIRI terbagi menjadi dua bagian utama. Bagian *frontend*, yang dinamakan Tirtayasa dan dibangun menggunakan bahasa pemrograman PHP serta mengandalkan basis data MySQL untuk menyimpan serta mengelola data. Selain itu, Tirtayasa juga menggunakan framework CodeIgniter 3. Saat menerima permintaan pencarian, Tirtayasa meneruskannya ke bagian *backend*, yaitu NewMenjangan. Hasil dari NewMenjangan kemudian diformat agar dapat dibaca dengan baik oleh pengguna. Bagian ini diimplementasikan dalam bahasa pemrograman Java dan berperan penting dalam perhitungan rute optimal.[2]
NewMenjangan merupakan program *daemon* yang berjalan secara otomatis saat server dinyalakan dan terus beroperasi hingga server dimatikan. *Daemon* sendiri adalah program komputer yang berjalan dilatar belakang dan tidak berinteraksi langsung dengan pengguna[3]. Pada saat eksekusi, NewMenjangan terhubung ke basis data MySQL untuk mengambil data rute angkot yang tersimpan dalam format LineString. LineString adalah salah satu tipe data geometris dalam MySQL yang mewakili satu atau lebih segmen garis yang terhubung. LineString terdiri dari urutan titik (*point*) yang membentuk jalur atau lintasan[4]. Setiap titik pada LineString merepresentasikan lokasi potensial untuk penumpang naik atau turun. Dari data tersebut, NewMenjangan membangun *weighted graph* dalam memori (RAM) dalam bentuk *adjacency list* dan melakukan prakomputasi. Setiap titik pada LineString menjadi akan *node*, dan antara titik ke-i dan titik ke-(i+1) dihubungkan dengan *edge*. Jika ada dua titik dari rute angkot berbeda yang berdekatan (jarak di bawah konstanta tertentu), maka dibuatkan juga *edge*, yang menunjukkan kemungkinan seseorang dapat turun dari suatu angkot dan naik ke angkot lainnya untuk meneruskan perjalanan.

[2] https://ejournals.umn.ac.id/index.php/IJNMT/article/view/784
[3] https://www.ibm.com/docs/en/aix/7.1?topic=processes-
[4] https://dev.mysql.com/doc/refman/8.4/en/gis-linestring-property-functions.html

Saat NewMenjangan menerima permintaan pencarian dari titik A ke titik B, kedua titik tersebut dijadikan *node* sementara, dan dibuatkan *edge* sementara ke *node-node* yang sudah ada sebelumnya, jika jaraknya di bawah konstanta tertentu. Pencarian jarak terdekat pada graf tersebut dilakukan menggunakan algoritma Dijkstra versi teroptimasi (*priority queue* dengan struktur data *heap*). Proses ini dapat dilakukan secara paralel dengan aman (*thread-safe*) tanpa mengubah graf utama.
Pada saat ini algoritma yang digunakan KIRI masih terikat dengan algoritma Dijkstra. Oleh karena itu, pada tugas akhir ini akan diimplementasikan algoritma lainnya, yaitu algoritma A* dan Floyd-Warshall sebagai *concrete strategy*. Selain itu, akan dilakukan juga penerapan arsitektur kelas *strategy pattern* sehingga aplikasi KIRI akan menjadi lebih fleksibel dalam pemilihan algoritma *shortest path* yang akan digunakan dan juga memudahkan apabila akan dilakukan perubahan atau perbaikan pada suatu algoritma yang digunakan.

3 Rumusan Masalah

1. Bagaimana cara melakukan perubahan kode pada NewMenjangan untuk menerapkan *strategy pattern*?
2. Bagaimana mengimplementasikan algoritma A* dan Floyd-Warshall sebagai *concrete strategy*?

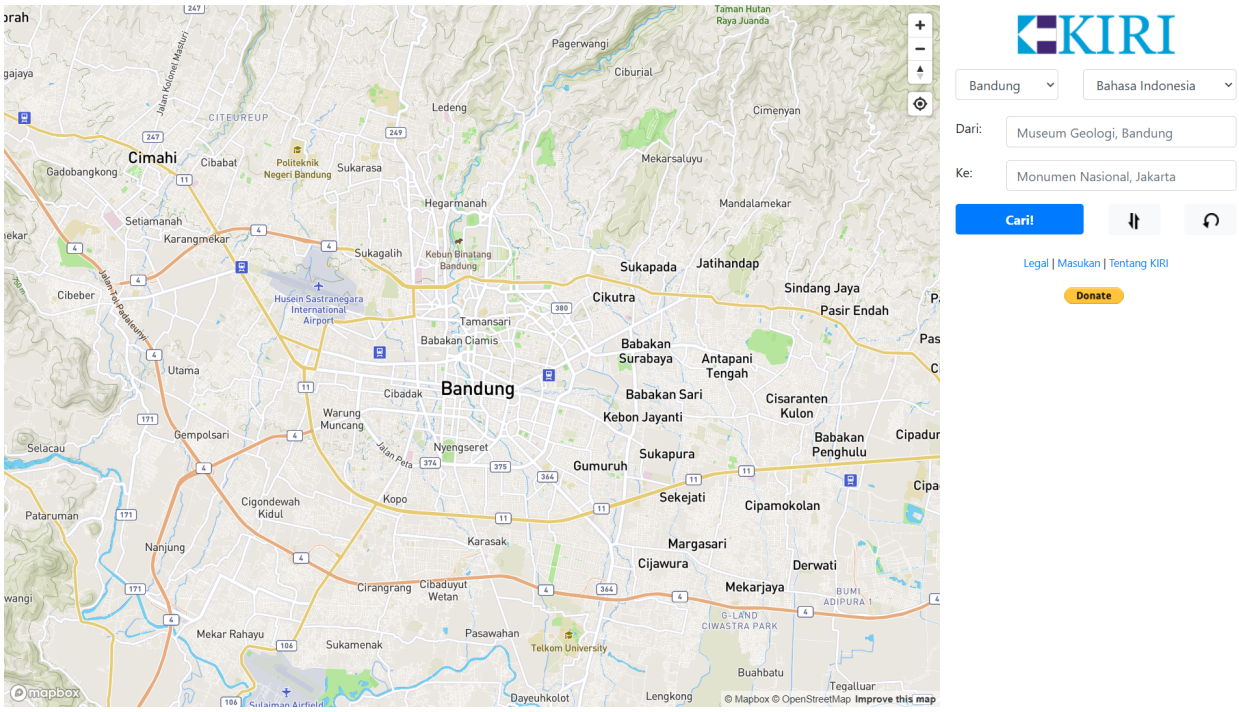
4 Tujuan

1. Melakukan perubahan arsitektur kelas dengan menerapkan *strategy pattern*.
2. Melakukan implementasi algoritma A-star dan Floyd Warshall.

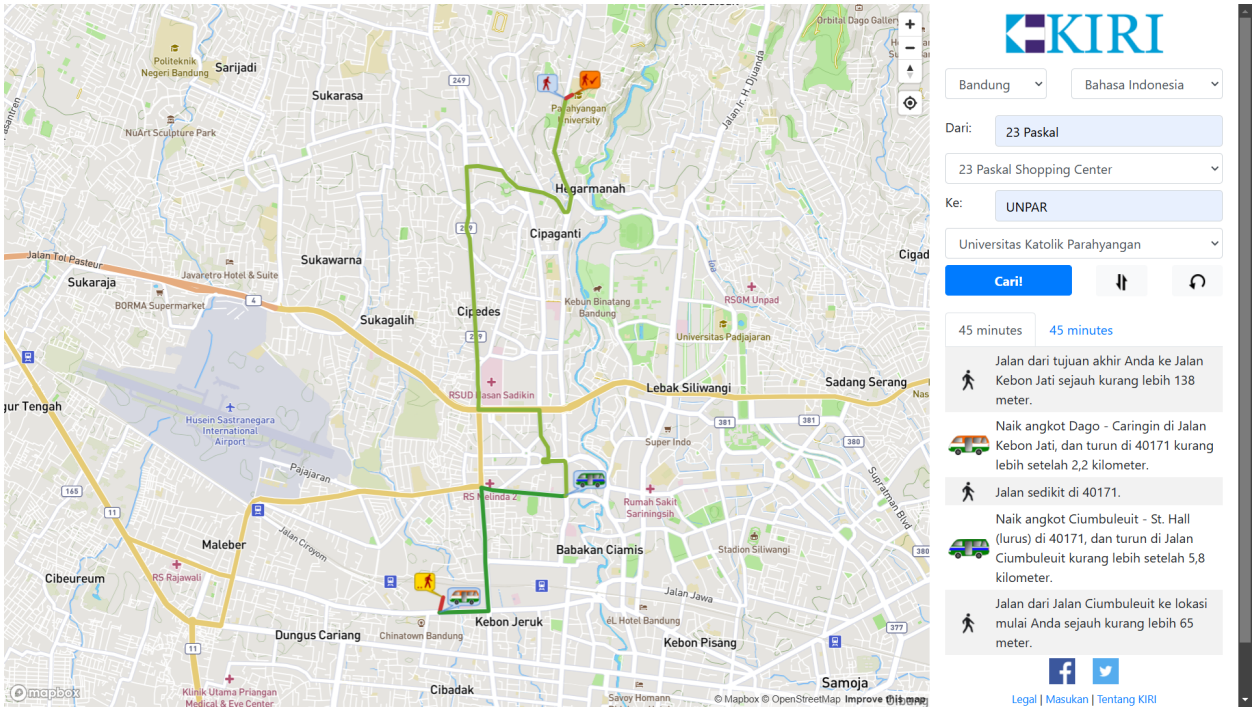
5 Detail Perkembangan Pengerjaan Tugas Akhir

Detail bagian pekerjaan skripsi sesuai dengan rencana kerja/laporan perkembangan terkahir :

1. **Melakukan eksplorasi fungsi-fungsi dan cara kerja perangkat lunak KIRI.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil : KIRI adalah sebuah aplikasi navigasi transportasi umum berbasis *web* yang menyediakan rute antara dua lokasi geografis menggunakan transportasi publik. KIRI dirancang untuk melayani kebutuhan pengguna angkot di Bandung serta TransJakarta dan Commuterline di DKI Jakarta.
   KIRI memberikan informasi detail mengenai rute perjalanan pengguna dari satu lokasi ke lokasi tujuan. Informasi yang disediakan meliputi seberapa jauh pengguna harus berjalan untuk mencapai titik naik angkot, lokasi tepat untuk naik atau turun angkot, jarak yang masih harus ditempuh dengan berjalan kaki hingga mencapai tujuan, serta estimasi waktu perjalanan yang akan dihabiskan.
   Pada aplikasi kiri pengguna dapat memanfaatkan peta dunia yang tersedia untuk melihat lokasi secara keseluruhan, dengan opsi *zoom in* dan *zoom out* untuk memperbesar atau memperkecil tampilan peta, sehingga memudahkan navigasi. Dengan fitur lokasi atau GPS, KIRI juga dapat mendeteksi lokasi pengguna secara otomatis yang mempermudah penentuan titik awal perjalanan. Selain itu, pengguna dapat memilih kota tertentu seperti Jakarta atau Bandung untuk memfokuskan pencarian rute perjalanan di wilayah yang relevan.
   KIRI juga menyediakan pilihan bahasa, yaitu bahasa Inggris atau bahasa Indonesia, sesuai dengan preferensi pengguna. Pengguna dapat memasukkan titik awal dan titik tujuan dengan dua cara, yaitu dengan mengetikkannya di kolom *input* atau dengan mengeklik langsung lokasi yang diinginkan pada peta. Setelah itu, aplikasi memungkinkan pencarian rute perjalanan berdasarkan masukan tersebut, memberikan rekomendasi rute terbaik yang mencakup informasi jarak, waktu, dan petunjuk langkah demi langkah. Tampilan awal halaman KIRI dan juga tampilan setelah menerima masukan berupa lokasi awal dan lokasi tujuan dapat dilihat pada gambar 3 dan gambar 4.

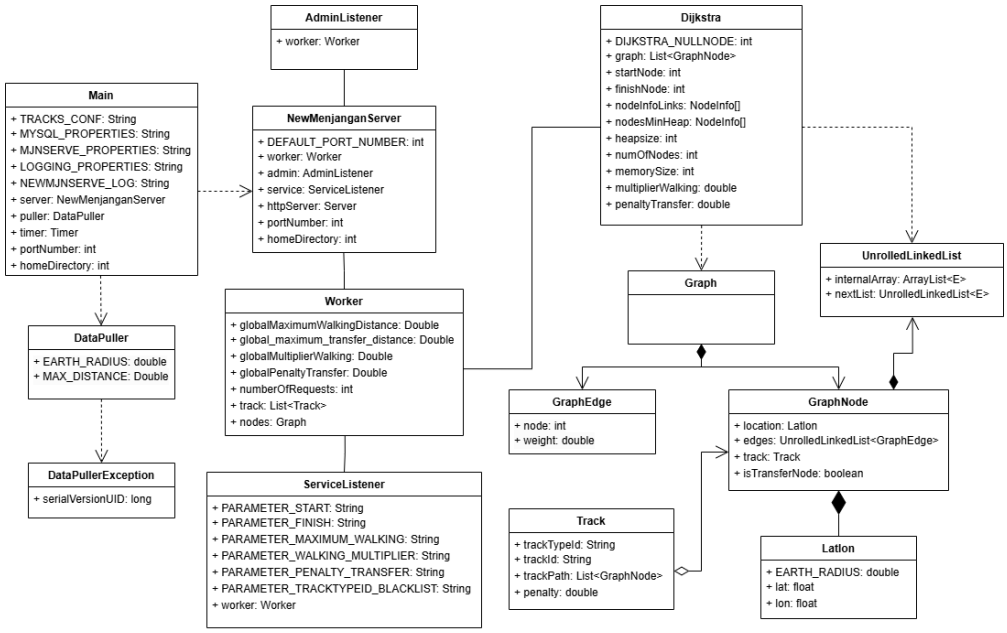


Gambar 3: Tampilan awal KIRI



Gambar 4: Tampilan KIRI setelah menerima masukan

2. **Mempelajari modul-modul yang terdapat pada Tirtayasa dan NewMenjangan.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil : Tirtayasa dan NewMenjangan merupakan 2 bagian utama dari KIRI. Tirtayasa merupakan bagian *frontend* yang dibangun menggunakan bahasa pemrograman PHP dan mengandalkan basis data MySQL untuk menyimpan serta mengelola data. Selain itu, Tirtayasa juga menggunakan framework CodeIgniter 3. Selanjutnya, yaitu NewMenjangan yang merupakan bagian *backend* dari KIRI dan dibangun dengan bahasa pemrogrman Java serta digunakan untuk memproses permintaan navigasi. Komponen ini memuat semua jalur transportasi umum dalam bentuk graf dan menggunakan algoritma Dijkstra untuk menghitung rute optimal. Berikut struktur kelas dari NewMenjangan (Gambar 5).

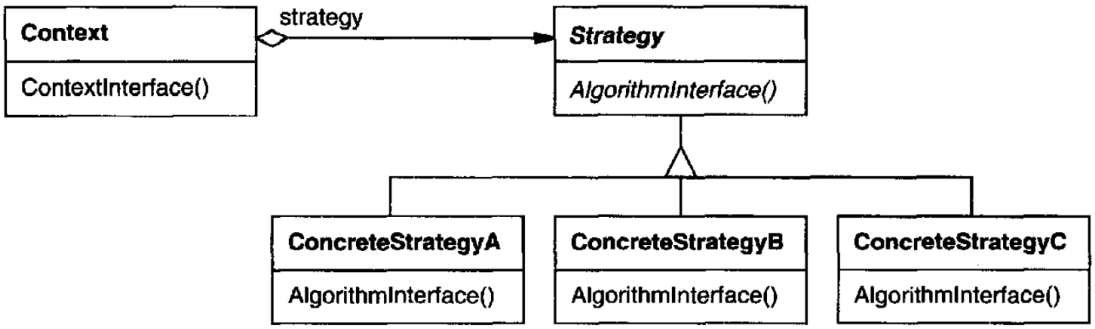


Gambar 5: Struktur Kelas NewMenjangan

3. **Mempelajari bahasa pemrograman PHP dan framework CodeIgniter 3.**
   Status : Dihapuskan/tidak dikerjakan.
   Hasil : Berdasarkan analisis singkat, bahasa pemrograman PHP dan framework CodeIgniter 3 merupakan bagian dari *frontend* KIRI, yaitu Tirtayasa. Pada tugas akhir ini, penelitian lebih difokuskan ke bagian *backend* KIRI (NewMenjangan) sehingga bahasa pemrograman PHP dan framework CodeIgniter 3 tidak didalami lebih jauh lagi.

4. **Melakukan studi literatur mengenai penerapan arsitektur kelas *straregy pattern*.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil : Strategy Pattern merupakan salah satu pola desain perilaku yang dirancang untuk mendefinisikan serangkaian algoritma, mengenkapsulasi setiap algoritma, dan memungkinkan algoritma-algoritma tersebut untuk saling dipertukarkan. Pola ini memungkinkan algoritma untuk bervariasi. Dengan demikian, klien tidak perlu mengetahui detail implementasi dari algoritma yang digunakan, melainkan cukup berinteraksi melalui antarmuka umum yang disediakan oleh objek strategi.
   Pola ini sangat berguna ketika terdapat kebutuhan untuk mendukung berbagai varian algoritma dalam menyelesaikan tugas yang sama. Strategy Pattern memindahkan setiap algoritma ke dalam kelas terpisah, yang disebut sebagai *concrete strategy*. Klien dapat memilih dan menentukan strategi yang sesuai ke dalam konteks pada waktu eksekusi, sehingga memberikan fleksibilitas yang tinggi dalam proses pengembangan perangkat lunak.

Manfaat utama dari strategy pattern adalah kemampuannya untuk menghilangkan kompleksitas yang diakibatkan oleh penggunaan pernyataan kondisional yang rumit dalam kode, serta kemudahan dalam menambahkan atau mengganti algoritma tanpa perlu memodifikasi kode klien atau konteks. Namun, penerapan pola ini juga memiliki kelemahan, seperti meningkatnya jumlah kelas dalam sistem dan potensi timbulnya *overhead* komunikasi antara konteks dan strategi. Oleh karena itu, penerapan strategy pattern sebaiknya dipertimbangkan dengan cermat, terutama dalam situasi di mana variasi algoritma memang diperlukan untuk memenuhi kebutuhan sistem.
Berikut merupakan struktur dari strategy pattern (Gambar 6).



Gambar 6: Struktur Strategy Pattern

5. **Melakukan studi literatur mengenai Graf.**
   Status : Baru ditambahkan ketika pengerjaan TA 1.
   Hasil : Graf (G) adalah struktur yang terdiri dari simpul (*vertex*) dan sisi (*edge*), dimana sisi menghubungkan pasangan simpul. Sebuah graf direpresentasikan sebagai pasangan $G = (V, E)$, dengan V sebagai himpunan simpul dan E sebagai himpunan sisi. Sisi diwakili oleh pasangan simpul yang terhubung. Graf dapat bersifat terarah atau tidak terarah.
   Simpul-simpul dalam graf dapat memiliki derajat tertentu, yaitu jumlah sisi yang menghubunginya. Sebuah graf disebut terhubung jika terdapat jalur antara setiap pasangan simpul. Jalur ini adalah urutan simpul yang dihubungkan oleh sisi. Selain itu, siklus adalah jalur tertutup di mana simpul awal dan akhir adalah sama. Teori graf juga mencakup konsep seperti pohon (graf terhubung tanpa siklus), graf bipartit (simpul dibagi menjadi dua himpunan yang saling bebas sisi), dan subgraf (bagian dari graf yang tetap mempertahankan struktur graf).
   Graf digunakan dalam berbagai hal, seperti jaringan komputer, rute transportasi, dan analisis hubungan sosial. Teori graf menyediakan dasar matematis untuk mempelajari struktur ini dan memberikan cara untuk memodelkan dan menyelesaikan masalah kompleks di berbagai bidang.

6. **Mempelajari cara kerja algoritma Dijkstra, A-star, dan Floyd Warshall.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil : Algoritma Dijkstra, Floyd-Warshall, dan A* merupakan sebuah algoritma *shortest path* yang dirancang untuk menemukan lintasan terpendek antara dua titik dalam sebuah graf. Ketiga algoritma tersebut merupakan algoritma yang akan diimplementasikan pada perangkat lunak KIRI sebagai *concrete strategy*. Berikut pembahasan cara kerja algoritma yaitu Dijkstra, Floyd-Warshall, dan A*.

- **Algoritma Dijkstra**
  Algoritma Dijkstra merupakan sebuah algoritma untuk menyelesaikan masalah *single-source shortest path*, yaitu menemukan jalur terpendek dari satu titik asal ke semua titik lainnya dalam sebuah graf berarah dengan bobot tepi non-negatif. Algoritma ini menggunakan sebuah struktur *min-priority queue* (antrean prioritas minimum) yang menyimpan titik-titik dengan prioritas sesuai dengan perkiraan jarak terpendek dari titik asal.

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

Gambar 7: Pseudocode Algoritma Dijkstra

Gambar 7 merupakan pseudocode dari algoritma Dijkstra. Pada pseudocode terdapat beberapa atribut diantarnya, yaitu G yang merepresentasikan graf, w merupakan bobot yang menyatakan jarak atau biaya antar dan s merepresentasikan simpul sumber yang merupakan titik awal pencarian. Selain itu, S merepresentasikan kumpulan simpul yang sudah diproses yang diawal diinisialisasikan kosong, sedangkan Q merepresentasikan kumpulan simpul yang belum diproses, kemudian u merepresentasikan simpul yang sedang diproses dan v merepresentasikan simpul tetangga dari u. Algoritma Dijkstra dimulai dengan menginisialisasi perkiraan jarak terpendek dari titik asal s ke semua titik lain, kecuali s itu sendiri yang diinisialisasi dengan jarak 0 dan juga semua simpul dimasukkan ke dalam *min-priority queue* (Q), di mana prioritas ditentukan berdasarkan jarak terpendek yang diketahui. Selanjutnya, algoritma memproses simpul-simpul satu per satu dengan memilih simpul u dari Q yang memiliki jarak terpendek. Simpul tersebut kemudian ditambahkan ke dalam himpunan S.
Setelah simpul u diproses, algoritma akan memeriksa semua tetangga v dari u. Untuk setiap tetangga, algoritma melakukan proses *relaksasi*, yaitu membandingkan jarak saat ini ke v dengan jarak yang melewati u. Jika jalur melalui u memberikan jarak yang lebih pendek, jarak ke v diperbarui dengan jarak baru tersebut, dan simpul pendahulu v diatur menjadi u. Proses ini memastikan bahwa jalur terpendek ditemukan secara bertahap melalui iterasi. Algoritma akan terus berjalan hingga semua simpul telah diproses atau antrean Q kosong. Hasil akhir berupa jarak terpendek dari simpul sumber s ke setiap simpul lain dalam graf.

- **Algoritma Floyd-Warshall**
  Algoritma Floyd-Warshall merupakan sebuah algoritma untuk menyelesaikan masalah jalur terpendek untuk semua pasangan titik dalam graf berarah dengan menggunakan pendekatan pemrograman dinamis. Algoritma ini sangat berguna untuk graf yang memiliki bobot sisi negatif, selama tidak terdapat siklus dengan bobot negatif dalam graf tersebut. Pendekatan ini menghitung jalur terpendek antara semua pasangan titik dengan menggunakan tabel bobot antar titik dan mengulanginya secara bertahap untuk mencapai solusi optimal.

```
FLOYD-WARSHALL(W)
1  n = W.rows
2  D^(0) = W
3  for k = 1 to n
4      let D^(k) = (d_ij^(k)) be a new n × n matrix
5      for i = 1 to n
6          for j = 1 to n
7              d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
8  return D^(n)
```

Gambar 8: Pseudocode Algoritma Floyd-Warshall

Gambar 8 merupakan pseudocode dari algoritma Floyd-Warshall. Pada pseudocode terdapat beberapa atribut diantarnya, yaitu W yang merupakan sebuah matriks berbobot berukuran $n * n$ dan mewakili bobot dari setiap sisi pada graf, $D^{(0)}$ atau $D^{(k)}$ merupakan Matriks berukuran $n * n$ pada iterasi ke-k, n merepresentasikan banyaknya simpul dalam graf, diperoleh dari jumlah baris dari matriks W. Selain itu, terdapat $d_{ij}^{(k)}$ yang merupakan elemen dari matriks $D^{(k)}$ yang menunjukkan jarak terpendek antara simpul i dan j pada iterasi ke-k. Algoritma Floyd-Warshall dimulai dengan menginisialisasi n yang diinisialisasi dengan nilai baris pada matriks W dan juga $D^{(0)}$ yang diinisialisasi dengan matriks W.

Selama n iterasi, algoritma memperbarui matriks jarak terpendek dengan mempertimbangkan kemungkinan jalan melalui simpul antara $\{1, 2, ..., k\}$. Pada setiap langkah, algoritma memeriksa apakah jarak dari i ke j dapat diperpendek dengan melalui simpul k, dibandingkan dengan jarak langsung antara i dan j. Proses ini menghasilkan solusi optimal, di mana $D^{(n)}$ mencakup semua jarak terpendek yang memungkinkan.Algoritma Floyd-Warshall memiliki kompleksitas waktu $O(n^3)$ karena terdiri dari tiga *looping* untuk semua titik dalam graf.

- **Algoritma A***

  Algoritma A* adalah metode pencarian yang meminimalkan estimasi total biaya solusi dengan menggabungkan dua fungsi, yaitu $g(n)$ dan $h(n)$ Fungsi $g(n)$ menghitung biaya aktual dari titik awal hingga simpul n, sedangkan $h(n)$ memperkirakan biaya tersisa dari n ke tujuan. Kombinasi ini menghasilkan $f(n) = g(n) + h(n)$, yang memberikan perkiraan total biaya solusi jika rute melalui simpul n. Algoritma ini biasanya dipilih karena dapat mencapai solusi yang optimal dan lengkap, terutama jika fungsi heuristik $h(n)$ memenuhi kriteria tertentu.

  Kondisi utama yang diperlukan agar algortima A* memberikan solusi optimal adalah heuristik $h(n)$ yang bersifat *admissible*, yaitu tidak pernah melebih-lebihkan biaya ke tujuan, dan *consistent* atau *monotonic*, di mana nilai h tidak menurun di sepanjang jalur. Dengan adanya heuristik yang memenuhi syarat ini, algoritma A* dapat menghindari eksplorasi simpul-simpul yang tidak relevan, mengurangi waktu dan memori yang dibutuhkan.

  Terdapat kendala utama dari algoritma A*, yaitu penggunaan memori yang besar karena algoritma ini perlu menyimpan semua simpul yang telah dihasilkan. Untuk mengatasi hal ini, terdapat varian A* seperti *Iterative-Deepening A** (IDA*) yang mengurangi kebutuhan memori tanpa mengorbankan optimalitas solusi, dengan biaya eksekusi yang sedikit lebih tinggi.

7. **Mengubah implementasi algoritma Dijkstra yang sudah ada ke dalam *strategy pattern.***
   Status : Ada sejak rencana kerja skripsi.
   Hasil : Implementasi belum terselesaikan karena pengerjaan difokuskan terlebih dahulu ke dokumen tugas akhir.

8. **Menulis sebagian dokumen skripsi, yaitu Bab 1, 2, dan 3.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil : Dokumen skripsi bab 1 berisikan latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan. Untuk saat ini batasan masalah masih belum diisi dan akan dilakukan ketika TA 2. Bab 2 berisikan penjelasan teori-teori yang diperlukan untuk penelitian, diantranya, yaitu KIRI, Design Pattern dan Strategy Pattern, MySQL dan *LineString*, dan Algoritma *Shortest Path* yang terdiri dari algoritma Dijkstra, algoritma Floyd-Warshall, dan algoritma A*. Kemudian, Bab 3 berisikan analisis sistem kini yang menjelaskan mengenai struktur kelas pada NewMenjangan dan juga analisis untuk sistem usulan, yang dimana akan diimplementasikan strategy pattern dengan algoritma Dijkstra, algoritma Floyd-Warshall, dan algoritma A* akan menjadi *concrete strategy*. Selain itu, akan diimplementasikan juga algoritma Floyd-Warshall, dan algoritma A* karena untuk saat ini hanya algoritma Dijkstra saja yang baru diimplementasikan.

6 Pencapaian Rencana Kerja

Langkah-langkah kerja yang berhasil diselesaikan dalam Tugas Akhir 1 ini adalah sebagai berikut:

1. Melakukan eksplorasi fungsi-fungsi dan cara kerja perangkat lunak KIRI.
2. Mempelajari modul-modul yang terdapat pada Tirtayasa dan NewMenjangan.
3. Melakukan studi literatur mengenai penerapan arsitektur kelas straregy pattern.
4. Melakukan studi literatur mengenai graf.
5. Mempelajari cara kerja algoritma Dijkstra, A*, dan Floyd Warshall.
6. Menulis sebagian dokumen skripsi, yaitu Bab 1, 2, dan 3.

Bandung, 06/12/2024

Muhammad Aldi Rivandi

Menyetujui,

Nama: Pascal Alfadian Nugroho, M.Comp.
Pembimbing Tunggal