**LAPORAN PROYEK PRAKTIKUM**

**ALGORITMA DAN STRUKTUR DATA**

**SAFAFLOW: SISTEM PENGELOLAAN DATA JEMAAH HAJI DENGAN ALGORITMA DAN STRUKTUR DATA**

****

**Oleh Kelompok 19**

**Anggota:**

**Ayu Diana F1D02310040**

**Azizah Indriani Putri F1D02310041**

**Baiq Mutia Dewi Edelweis F1D02310107**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS MATARAM**

**2024**

**LEMBAR PENGESAHAN LAPORAN PROYEK**

**PRAKTIKUM ALGORITMA DAN STRUKTUR DATA 2024**

1. Kelompok : 19
2. Judul Proyek :SafaFlow Sistem Pengelolaan Data Jemaah Haji
3. Anggota Kelompok : Ayu Diana (F1D02310040)

Azizah Indriani Putri (F1D02310041)

Baiq Mutia Dewi Edelweis (F1D02310107)

Laporan proyek ini disusun sesuai dengan kaidah penyusunan yang telah ditentukan dan dibuat sebagai syarat mata kuliah Algoritma dan Struktur Data 2024.

Mataram, 22 Desember 2024

Telah diperiksan dan disahkan oleh:

|  |  |
| --- | --- |
| **Koordinator Asisten**  **Aditya Rahmatdiansyah**  **F1D022031** | **Asisten Pembimbing**  **Muhammad Kholiluloh**  **F1D022014** |

1. **Judul**

SafaFlow: Sistem Pengelolaan Data Jemaah Haji

1. **Latar Belakang**

Haji adalah salah satu rukun Islam yang memiliki signifikansi besar bagi umat Muslim di seluruh dunia. Proses pengelolaan calon jemaah haji melibatkan banyak aspek administratif yang kompleks, termasuk pencatatan data jemaah, pengelolaan pembayaran, antrean keberangkatan, dan pengelompokan berdasarkan hubungan keluarga atau ahli waris. Untuk mengatasi tantangan ini, diperlukan sebuah sistem manajemen yang mampu menangani data secara efisien, fleksibel, dan dapat memberikan solusi otomatisasi berbasis algoritma dan struktur data. SafaFlow bisa menjadi solusi inovatif untuk mendukung pengelolaan data jemaah haji dengan memanfaatkan konsep algoritma dan struktur data, seperti *linkedlist*, stack, *queue*, *tree*, serta teknik *searching* dan *sorting*.

Dalam SafaFlow, setiap fitur dirancang untuk mengoptimalkan proses pengelolaan data. Penggunaan *linkedlist* memungkinkan penambahan, penghapusan, dan pembaruan data jemaah secara dinamis, sehingga data calon jemaah dapat dikelola dengan lebih fleksibel. Stack diterapkan untuk menyimpan riwayat transaksi pembayaran, mempermudah pelacakan histori dan pembatalan transaksi jika diperlukan. Selain itu, antrean keberangkatan dikelola menggunakan *queue*, di mana sistem mengutamakan prinsip *first*-*in* *first*-*out* (FIFO) untuk memastikan urutan keberangkatan jemaah sesuai dengan waktu pendaftaran.

SafaFlow juga mendukung pencarian dan pengurutan data jemaah menggunakan algoritma *searching* dan *sorting* untuk menghasilkan laporan yang terstruktur dan mempermudah pengelompokan data. Selain itu, *tree* digunakan untuk memetakan hubungan keluarga atau ahli waris calon jemaah, memberikan kemudahan dalam pelacakan ahli waris terdekat jika terjadi pembatalan. Dengan kombinasi struktur data ini, SafaFlow bertujuan menciptakan sistem pengelolaan haji yang modern, efisien, dan dapat diandalkan, sehingga mendukung pelaksanaan ibadah haji yang lebih terorganisir dan lancar.

1. **Deskripsi Program**

Mengimplementasikan algoritma dan struktur data dalam proses manajemen data Jemaah haji adalah program yang dibuat dengan menggunakan bahasa pemrograman java dan mengimplementasikan bentuk linked list, stack, queue, searching, sorting, dan *tree*. Program ini ditunjukkan sebagai sistem dalam memberikan layanan sistem mengenai pengolahan data Jemaah haji. Pada dasarnya program ini merupakan program yang digunakan untuk membantu pengurus haji dalam memproses data jamaah haji yang mendaftar dan keberangkatannya. Program ini memiliki beberapa menu, yaitu:

1. Tambah Jemaah
2. Update Jemaah
3. Hapus Jemaah
4. Tampilkan Calon Jemaah
5. Tambah Transaksi Pembayaran
6. Batalkan Transaksi Pembayaran
7. Tampilkan Riwayat Pembayaran
8. Tambah ke Antrean Keberangkatan
9. Proses Keberangkatan Jemaah
10. Tampilkan Antrean

Program ini dimulai dengan pengguna dapat dimulai dengan user memilih dan mengisi data menu yang ingin di pilih. Setiap pilihan yang dilakukan oleh user memiliki fungsinya tersendiri. Melalui implementasi struktur data, proyek ini tidak hanya memberikan solusi praktis untuk pengelolaan data Jemaah haji, tetapi juga menciptakan dasar yang kuat untuk pengembangan dan perluasan fitur di masa depan. Dengan fokus pada efisiensi dan keterorganisasian, program ini diharapkan dapat meningkatkan produktivitas dan pelayanan Jemaah Haji secara keseluruhan.

1. **Algoritma**

Berikut merupakan algoritma yang digunakan untuk menyelesaikan permasalahan pada program:

* + - 1. Melakukan inisialisasi objek *tree* untuk menampung ahli waris dari Jemaah

1. **Penjelasan Code**
2. **Code Class Node**

|  |
| --- |
| // Node.java  public class Node {     String nomorPendaftaran;      String TanggalPendaftaran;      boolean StatusPembayaran;      String Nama;      String JenisKelamin;      int Umur;      Node next;      TreeNode ahliWarisRoot;      String Penyakit;      Stack riwayatPembayaran;      public Node(String nomorPendaftaran, String TanggalPendaftaran, boolean StatusPembayaran, String Nama, String JenisKelamin, int Umur, String ahliWarisPertama, String ahliWarisKedua, String penyakit){          this.nomorPendaftaran = nomorPendaftaran;          this.TanggalPendaftaran = TanggalPendaftaran;          this.StatusPembayaran = StatusPembayaran;          this.Nama = Nama;          this.JenisKelamin = JenisKelamin;          this.Penyakit = penyakit;          this.Umur = Umur;          this.next = null;          this.riwayatPembayaran = new Stack();          this.ahliWarisRoot = new TreeNode(ahliWarisPertama);          if (ahliWarisKedua != null && !ahliWarisKedua.isEmpty()) {              this.ahliWarisRoot.addSibling(new TreeNode(ahliWarisKedua));          }      } |

Script di atas merupakan script dari class “Node”. Class “Node” berfungsi untuk menginialisasikan parameter yang akan digunakan pada program. Parameter yang disimpan yaitu “nomorPendaftaran”, “TanggalPendaftaran”, “StatusPembayaran”, “Nama”, “JenisKelamin”, “Penyakit”, dan “ahliWarisRoot” dari kelas “TreeNode”. “Node” ini juga memiliki parameter “riwayatPembayaran” yang berupa stack untuk menyimpan catatan riwayat pembayaran jemaah. Pada class “Node” terdapat konstruktor yang bertugas mengatur data awal dari setiap atribut sesuai nilai yang diberikan saat pembuatan objek. Script “next” digunakan untuk menghubungkan node satu dengan node berikutnya di dalam linked list.

|  |
| --- |
| public Node(String nomorPendaftaran, String TanggalPendaftaran, boolean StatusPembayaran, String Nama, String JenisKelamin, int Umur, String penyakit, TreeNode ahliWarisRoot) {          this.nomorPendaftaran = nomorPendaftaran;          this.TanggalPendaftaran = TanggalPendaftaran;          this.StatusPembayaran = StatusPembayaran;          this.Nama = Nama;          this.JenisKelamin = JenisKelamin;          this.Umur = Umur;          this.Penyakit = penyakit;          this.ahliWarisRoot = ahliWarisRoot; // Set ahliWarisRoot          this.next = null;  } |

Script di atas adalah sebuah konstruktor dalam kelas “Node” yang digunakan untuk menginisialisasi sebuah objek “Node” dengan data jemaah tertentu. Konstruktor ini menerima parameter “nomorPendaftaran” untuk menyimpan nomor pendaftaran jemaah, “TanggalPendaftaran” untuk mencatat tanggal pendaftaran, “StatusPembayaran” yang berupa boolean untuk mencatat status pembayaran (lunas atau belum), “Nama” dan “JenisKelamin” untuk informasi pribadi jemaah, “Umur” untuk mencatat usia jemaah, serta “Penyakit” untuk mencatat riwayat penyakit yang dimiliki. Selain itu, konstruktor juga menerima parameter “ahliWarisRoot” dari “TreeNode” yang digunakan untuk merepresentasikan struktur ahli waris dalam bentuk *tree*. Atribut next diinisialisasi sebagai null untuk menunjukkan bahwa node ini belum terhubung ke node lain dalam *linked* *list*.

|  |
| --- |
| public void displayTree() {          if (ahliWarisRoot == null) {              System.out.println("  Tidak ada ahli waris.");          } else {              ahliWarisRoot.displayTree(0);          }      } |

Script di atas merupakan *Method* “displayTree()” yang berfungsi untuk menampilkan struktur pohon ahli waris dari sebuah node. Jika atribut “ahliWarisRoot” bernilai “null”, *method* ini akan mencetak pesan “Tidak ada ahli waris”, menunjukkan bahwa tidak ada data ahli waris yang tersimpan. Sebaliknya, jika “ahliWarisRoot” tidak kosong, *method* ini akan memanggil fungsi “displayTree(int level)” yang dimiliki oleh objek “ahliWarisRoot”. *Method* tersebut bertanggung jawab untuk mencetak seluruh isi *tree* ahli waris secara rekursif, dimulai dari akar dan meneruskan ke anak-anak atau saudara-saudara pada level berikutnya. Parameter “0” dalam pemanggilan menunjukkan bahwa pencetakan dimulai dari level pertama *tree*.

|  |
| --- |
| // Menambahkan ahli waris baru sebagai child atau sibling      public void tambahAhliWaris(String namaBaru, boolean sebagaiAnak){          TreeNode ahliWarisBaru = new TreeNode(namaBaru);          if (sebagaiAnak) {              this.ahliWarisRoot.addChild(ahliWarisBaru);          } else {              this.ahliWarisRoot.addSibling(ahliWarisBaru);          }      }  } |

Script di atas merupakan *method* “tambahAhliWaris” digunakan untuk menambahkan ahli waris baru ke dalam struktur pohon ahli waris yang terkait dengan sebuah node. *Method* ini menerima dua parameter yaitu “namaBaru”, yaitu nama ahli waris yang akan ditambahkan, dan sebagaiAnak dan “Boolean” yang menentukan apakah ahli waris baru akan ditambahkan sebagai anak dari “ahliWarisRoot” atau sebagai saudara “sibling”. Jika sebagai Anak bernilai “true**”**, *method* ini akan memanggil fungsi “addChild” pada “ahliWarisRoot” dan menambahkan ahli waris baru sebagai anaknya. Sebaliknya, jika sebagaiAnak bernilai “false**”** fungsi “addSibling” akan dipanggil untuk menambahkan ahli waris baru sebagai saudara dari “ahliWarisRoot”.

1. **Code Class *Tree*Node**

|  |
| --- |
| public class TreeNode {  String namaAhliWaris;  TreeNode firstChild;  TreeNode nextSibling;  public TreeNode(String namaAhliWaris) {  this.namaAhliWaris = namaAhliWaris;  this.firstChild = null;  this.nextSibling = null;  } |

Script di atas adalah class “TreeNode”, yang digunakan untuk merepresentasikan node dalam struktur data pohon ahli waris. Setiap objek dari class “TreeNode” memiliki atribut “name”, yang menyimpan nama ahli waris, serta dua atribut referensi, yaitu “firstChild” dan “nextSibling”, yang masing-masing mewakili *subtree* anak pertama dan saudara berikutnya dari node tersebut. Konstruktor “TreeNode” digunakan untuk menginisialisasi node baru dengan nama ahli waris yang diberikan.

|  |
| --- |
| public void addChild(TreeNode child) {  if (this.firstChild == null) {  this.firstChild = child;  } else {  TreeNode temp = this.firstChild;  while (temp.nextSibling != null) {  temp = temp.nextSibling;  }  temp.nextSibling = child;  }  } |

Script di atas merupakan *method* “addChild(TreeNode child)” pada kelas “TreeNode” digunakan untuk menambahkan anak (child) baru ke node dalam struktur *tree*. Jika node saat ini belum memiliki anak “firstChild” bernilai “null”, maka child baru langsung ditambahkan sebagai “firstChild”. Namun, jika node sudah memiliki anak, *method* ini akan menelusuri daftar saudara “sibling” dari anak pertama hingga mencapai anak terakhir, kemudian menambahkan child baru sebagai saudara berikutnya “nextSibling”.

|  |
| --- |
| public void addSibling(TreeNode sibling) {  TreeNode temp = this;  while (temp.nextSibling != null) {  temp = temp.nextSibling;  }  temp.nextSibling = sibling;  } |

Script di atas merupakan method “addSibling(TreeNode sibling)” pada kelas “TreeNode” digunakan untuk menambahkan saudara “sibling” baru ke node saat ini dalam struktur *tree*. *Method* ini dimulai dengan menelusuri atribut “nextSibling” hingga mencapai saudara terakhir, yaitu node dengan “nextSibling == null”. Setelah itu, “sibling” baru ditambahkan sebagai saudara berikutnya dengan menetapkannya pada atribut “nextSibling” dari saudara terakhir.

|  |
| --- |
| public void displayTree(int level) {  String indentation = " ".repeat(level);  System.out.println(indentation + "|\_" + this.namaAhliWaris);  if (this.firstChild != null) {  this.firstChild.displayTree(level + 1);  }  if (this.nextSibling != null) {  this.nextSibling.displayTree(level);  }  } |

Script di atas merupakan *method* “displayTree(int level)” pada kelas “TreeNode” digunakan untuk menampilkan struktur *tree* dari node tertentu secara hierarkis. Parameter “level” menentukan tingkat kedalaman node dalam *tree*. Jika node memiliki anak “firstChild”, *method* ini akan dipanggil secara rekursif dengan tingkat kedalaman yang bertambah satu. Jika node memiliki saudara “nextSibling”, *method* akan dipanggil dengan tingkat kedalaman yang sama.

|  |
| --- |
| public TreeNode getChild() {  return this.firstChild;  }  public TreeNode getSibling() {  return this.nextSibling;  } |

Script di atas merupakan method “getChild()” dan “getSibling()” pada kelas “TreeNode” digunakan untuk mengakses anak dan saudara dari node saat ini dalam pohon. “getChild()” mengembalikan node pertama anak “firstChild” dari node saat ini. Jika node ini tidak memiliki anak, maka *method* ini akan mengembalikan “null”. “getSibling()” mengembalikan saudara berikutnya “nextSibling” dari node saat ini. Jika node ini tidak memiliki saudara, maka *method* ini akan mengembalikan “null”.

1. **Code Class LinkedList**

|  |
| --- |
| public class LinkedList {      Node head;      public void addData(String nomorPendaftaran, String TanggalPendaftaran, boolean StatusPembayaran,      String Nama, String JenisKelamin, int Umur, String ahliWarisPertama, String ahliWarisKedua, String Penyakit) {            Node newNode = new Node(nomorPendaftaran, TanggalPendaftaran, StatusPembayaran,          Nama, JenisKelamin, Umur, ahliWarisPertama, ahliWarisKedua, Penyakit);          if (head == null) {              head = newNode;          } else {              Node temp = head;              while (temp.next != null) {                  temp = temp.next;              }              temp.next = newNode;          }      } |

Script di atas merupakan method “addData()” pada kelas “LinkedList”, yang digunakan untuk menambahkan data baru ke dalam linked list. Method ini menerima beberapa parameter seperti “nomorPendaftaran, TanggalPendaftaran, StatusPembayaran, Nama, JenisKelamin, Umur, ahliWarisPertama, ahliWarisKedua, Penyakit” yang digunakan untuk membuat node baru. Jika node ada dalam linked list, method ini akan mencari node terakhir dengan mengiterasi melalui setiap node hingga menemukan node yang “next” bernilai “null”, kemudian menambahkan node baru di akhir dengan mengubah “next” node terakhir agar menunjuk ke node baru.

|  |
| --- |
| public boolean cekNama(String Nama) {          Node temp = head;          while (temp != null) {              if (temp.Nama.equalsIgnoreCase(Nama)) {                  return true;              }              temp = temp.next;          }          return false;      } |

Script di atas merupakan method “cekNama()” pada kelas “LinkedList”, yang digunakan untuk memeriksa apakah ada node dalam linked list yang memiliki nama yang sesuai dengan parameter “Nama”. Method ini akan mulai memeriksa dari node pertama “head” dan mengiterasi setiap node yang ada. Jika ditemukan node dengan nama yang sama method ini akan mengembalikan nilai “true,” yang berarti nama tersebut ada dalam linked list. Jika tidak ditemukan node dengan nama yang cocok hingga akhir linked list “node next” bernilai “null” method ini akan mengembalikan “false”, yang berarti nama tersebut tidak ada dalam linked list.

|  |
| --- |
| public void bayar(String nomorPendaftaran, String tanggal, int jumlah, String metode) {          Node temp = head;          while (temp != null && !temp.nomorPendaftaran.equals(nomorPendaftaran)) {              temp = temp.next;          }          if (temp == null) {              System.out.println("Nomor pendaftaran tidak ditemukan.");          } else {              temp.riwayatPembayaran.push(tanggal, jumlah, metode);              if(temp.riwayatPembayaran.totalPembayaran() >= 56000000){                  temp.StatusPembayaran = true;              }              System.out.println("Pembayaran berhasil dilakukan.");          }      } |

Script di atas merupakan method “bayar()” pada kelas “LinkedList” yang digunakan untuk memproses pembayaran berdasarkan nomor pendaftaran. Jika “nomorPendaftaran” tidak ditemukan, maka akan menampilkan pesan bahwa “nomor pendaftaran tidak ditemukan”. Jika ditemukan, pembayaran akan ditambahkan ke “riwayatPembayaran” dengan memanggil *method* “push()” pada stack “riwayatPembayaran” yang mencatat tanggal, jumlah pembayaran, dan *method* pembayaran. Selanjutnya, jika “totalPembayaran” yang tercatat dalam “riwayatPembayaran” mencapai atau melebihi “56.000.000”, “StatusPembayaran” jemaah akan diperbarui menjadi “true” menandakan bahwa pembayaran sudah lunas.

|  |
| --- |
| public boolean cekNomorPendaftaran(String nomorPendaftaran) {          Node temp = head;          while (temp != null) {              if (temp.nomorPendaftaran.equals(nomorPendaftaran)) {                  return true;              }              temp = temp.next;          }          return false;      } |

Script di atas merupakan method “cekNomorPendaftaran()” pada kelas “LinkedList” yang digunakan untuk memeriksa apakah “nomorPendaftaran” yang diberikan sudah terdaftar dalam linked list. Jika “nomorPendaftaran” ditemukan method ini akan mengembalikan nilai “true”. Jika “nomorPendaftaran” tidak ditemukan di seluruh linked list, maka method ini akan mengembalikan nilai “false” menandakan bahwa nomor pendaftaran tersebut belum terdaftar.

|  |
| --- |
| public void hapusJemaah(String nomorPendaftaran, boolean hapusSeluruh) {          if (head == null) {              System.out.println("Data kosong");              return;          }          if (head.nomorPendaftaran.equals(nomorPendaftaran)) {              if (hapusSeluruh) {                  head = head.next;              } else {                  Node temp = head;                  Node parentToDelete = head;                  head = head.next;                  if (temp.ahliWarisRoot != null) {                      Node newNode = new Node(temp.ahliWarisRoot.namaAhliWaris, parentToDelete.TanggalPendaftaran, false, "", "", 0, "", parentToDelete.ahliWarisRoot.nextSibling);                      newNode.next = head;                      head = newNode;                      TreeNode child = temp.ahliWarisRoot.firstChild;                      while (child != null) {                          head.tambahAhliWaris(child.namaAhliWaris, true); // Tambahkan sebagai anak                          child = child.nextSibling;                      }                  }              }              return;          }      } |

Script di atas merupakan method “hapusJemaah” digunakan untuk menghapus data jemaah dari data berdasarkan nomor pendaftaran yang diberikan, dengan opsi untuk menghapus seluruh data ahli waris atau tidak. Jika linked list kosong “head” adalah “null”, maka *method* akan mencetak pesan “Data kosong” dan keluar. Jika “nomorPendaftaran” yang diberikan ada pada “head”, maka jemaah tersebut akan dihapus. Jika parameter “hapusSeluruh” bernilai “true”, maka jemaah akan dihapus dan posisi “head” akan dipindahkan ke node berikutnya. Jika tidak, ahli waris yang terkait dengan jemaah tersebut akan dipindahkan ke node baru yang akan menjadi “head”, dan anak-anak ahli warisnya akan ditambahkan sebagai ahli waris baru di node tersebut. Jika “nomorPndaftaran” tidak ada pada “head”, maka fungsi akan mencari node dengan nomor pendaftaran yang cocok di seluruh linked list dan memperoses data seperti penghapusan data pada “head”.

|  |
| --- |
| public void display() {          Node temp = head;          System.out.println("=====================================================");          System.out.println("                DETAIL DATA JEMAAH                   ");          System.out.println("=====================================================");          while (temp != null) {              System.out.println("Nomor Pendaftaran   : " + temp.nomorPendaftaran);              System.out.println("Nama Jemaah         : " + temp.Nama);              System.out.println("Jenis Kelamin       : " + temp.JenisKelamin);              System.out.println("Umur                : " + temp.Umur);              System.out.println("Tanggal Pendaftaran : " + temp.TanggalPendaftaran);              System.out.println("Status Pembayaran   : " + (temp.StatusPembayaran ? "Lunas" : "Belum Lunas"));              System.out.println("Penyakit            : " + temp.Penyakit );              System.out.println("-----------------------------------------------------");              temp.displayTree();              System.out.println("-----------------------------------------------------");              temp = temp.next;          }      } |

Script di atas merupakan method “display()” pada kelas “LinkedList” yang digunakan untuk menampilkan detail data jemaah yang terdaftar dalam linked list. Method ini akan menampilkan informasi terkait setiap jemaah, termasuk “nomorPendaftaran”, “nama”, “jenisKelamin”, “umur”, “tanggalPendaftaran”, “StatusPembayaran”, serta “penyakit” yang diderita. Setelah menampilkan informasi method ini juga akan memanggil method “displayTree()” dari setiap node yang akan menampilkan data ahli waris dari jemaah. Setelah semua informasi terkait jemaah ditampilkan, method ini akan berpindah ke node berikutnya hingga semua data jemaah dalam linked list ditampilkan.

|  |
| --- |
| public void showPaymentHistory(){          Node temp = head;          System.out.println("=====================================================");          System.out.println("                RIWAYAT PEMBAYARAN                   ");          System.out.println("=====================================================");          while (temp != null) {              System.out.println("Nomor Pendaftaran: " + temp.nomorPendaftaran);              System.out.println("Nama Jemaah: " + temp.Nama);              System.out.println("-----------------------------------------------------");              temp.riwayatPembayaran.tampilkanRiwayat();              System.out.println("Status Pembayaran: " + (temp.StatusPembayaran ? "Lunas" : "Belum Lunas"));              System.out.println              ("-----------------------------------------------------");              temp = temp.next;          }  } |

Script di atas merupakan method “showPaymentHistory()” pada kelas “LinkedList” yang digunakan untuk menampilkan “riwayatPembayaran” dari setiap jemaah yang terdaftar dalam linked list. Method ini dimulai dengan mencetak header untuk menandai bagian “riwayatPembayaran”. Setelah itu, method ini akan memanggil method “tampilkanRiwayat()” dari parameter “riwayatPembayaran” pada setiap node untuk menampilkan detail riwayat pembayaran jemaah. Setelah menampilkan informasi tersebut, program akan berpindah ke node berikutnya hingga seluruh data pembayaran jemaah ditampilkan.

1. **Code Class Stack**

|  |
| --- |
| public class Stack {      class Node {          String tanggal2;          double jumlah;          String Metode;          Node next;          Node(String tanggal, double jumlah, String metode) {              this.tanggal2 = tanggal;              this.jumlah = jumlah;              this.Metode = metode;              this.next = null;          }      }      Node top;      public Stack() {          top = null;      } |

Script di atas merupakan definisi kelas “Stack” yang berfungsi untuk mengelola struktur data stack yang digunakan untuk menyimpan riwayat pembayaran. Kelas “Node” yang berada di dalam kelas “Stack” berfungsi untuk merepresentasikan elemen-elemen dalam stack dengan parameter seperti “tanggal2”, “jumlah”, dan “Metode” yang menyimpan informasi pembayaran, serta “next” yang menunjuk ke node berikutnya dalam stack. Konstruktor “Node” digunakan untuk menginisialisasi elemen baru dalam stack dengan nilai-nilai yang diberikan, sedangkan konstruktor “Stack” menginisialisasi stack dengan “top” yang bernilai “null” yang berarti stack dimulai dalam keadaan kosong.

|  |
| --- |
| public void push(String tanggal, double jumlah, String metode) {          Node newNode = new Node(tanggal, jumlah, metode);          newNode.next = top;          top = newNode;  } |

Script di atas merupakan method “push()” yang digunakan untuk menambahkan elemen baru ke dalam stack pada kelas “Stack”. Method ini menerima tiga parameter yaitu “tanggal”, “jumlah”, dan “metode” yang berfungsi untuk menyimpan informasi terkait riwayat pembayaran. Sebuah node baru dibuat menggunakan konstruktor “Node” dengan nilai-nilai yang diberikan. Kemudian, node baru tersebut disisipkan di bagian atas stack dengan cara menjadikan “next” dari node baru tersebut menunjuk ke node yang sebelumnya berada di atas “top”, dan akhirnya “top’ diupdate untuk menunjuk ke node baru tersebut, menjadikannya elemen pertama dalam stack.

|  |
| --- |
| public void pop() {       if (top == null) {          System.out.println("Tidak ada transaksi untuk dibatalkan.");      } else {          System.out.println("Pembayaran tanggal " + top.tanggal2 + " dengan jumlah Rp" + top.jumlah + " dibatalkan.");              top = top.next;       }  } |

Script di atas merupakan method “pop()” yang digunakan untuk menghapus elemen paling atas dari stack pada kelas “Stack”. Method ini pertama-tama memeriksa apakah stack kosong dengan memeriksa apakah “top” bernilai “null”. Jika stack kosong, maka pesan “Tidak ada transaksi untuk dibatalkan.” akan ditampilkan. Jika stack tidak kosong, elemen paling atas “top” akan dihapus. Setelah itu, “top” diupdate untuk menunjuk ke node berikutnya, menghapus node yang lama dan menjadikannya elemen paling atas yang baru.

|  |
| --- |
| public void tampilkanRiwayat() {          if (top == null) {              System.out.println("Riwayat pembayaran kosong.");          } else {              Node temp = top;              while (temp != null) {                  System.out.println("Tanggal: " + temp.tanggal2);                  System.out.println("Jumlah : Rp" + temp.jumlah);                  System.out.println("Metode Pembayaran cash/transfer: " + temp.Metode);                  temp = temp.next;              }          }      } |

Script di atas merupakan method “tampilkanRiwayat” digunakan untuk menampilkan seluruh riwayat transaksi yang ada di dalam stack. Jika stack kosong, akan muncul pesan bahwa “Riwayat Pembayaran Kosong”. Jika ada transaksi, method ini akan mencetak informasi setiap transaksi, mulai dari yang teratas hingga yang terbawah.

1. **Code Class Queue**

|  |
| --- |
| public class Queue {      class Node {          String nama;          Node next;          Node(String nama) {              this.nama = nama;              this.next = null;          }      }      Node front, rear;      public Queue() {          front = rear = null;      } |

Script di atas mendefinisikan kelas “Queue” yang mengimplementasikan antrian menggunakan linked list. Di dalam kelas “Queue”, terdapat kelas “Node” yang digunakan untuk merepresentasikan elemen dalam antrian. Setiap node menyimpan nama dan *pointer* “next” yang mengarah ke node berikutnya dalam antrian. Kelas “Queue” memiliki dua *pointer* utama, yaitu “front” dan “rear”, yang masing-masing menunjuk ke elemen pertama dan terakhir dalam antrian. Konstruktor “Queue()” menginisialisasi kedua *pointer* tersebut sebagai “null” yang menunjukkan bahwa antrian awalnya kosong.

|  |
| --- |
| public void enqueue(String nama) {          Node newNode = new Node(nama);          if (rear == null) {              front = rear = newNode;              return;          }          rear.next = newNode;          rear = newNode;      } |

Script di atas mendefinisikan *method* “enqueue” dalam kelas “Queue” yang digunakan untuk menambahkan elemen baru ke dalam antrian. *Method* ini menerima parameter “nama” yang akan disimpan dalam node baru. Jika antrian masih kosong yaitu “rear” bernilai “null” maka elemen baru akan menjadi elemen pertama dalam antrian yaitu “front” maupun “rear” akan menunjuk ke node baru tersebut. Jika antrian tidak kosong, node baru akan ditempatkan di akhir antrian dengan menghubungkan node terakhir yang ada (yang ditunjuk oleh “rear” ke node baru dan kemudian “rear” akan diperbarui untuk menunjuk ke node baru.

|  |
| --- |
| public void dequeue() {          if (front == null) {              System.out.println("Antrean kosong.");              return;          }          System.out.println("Jemaah yang berangkat: " + front.nama);          front = front.next;          if (front == null) {              rear = null;          }      } |

Script di atas mendefinisikan *method* “dequeue” dalam kelas “Queue”, yang berfungsi untuk mengeluarkan elemen pertama (depan) dari antrian. Pertama, *method* ini memeriksa apakah antrian kosong dengan memeriksa apakah “front” bernilai “null”. Jika antrian kosong, pesan “Antrean kosong.” akan ditampilkan, dan *method* langsung berhenti. Jika antrian tidak kosong, maka nama jemaah yang berada di posisi depan “front” akan ditampilkan sebagai jemaah yang berangkat. Selanjutnya, *pointer* “front” akan dipindahkan ke elemen berikutnya “front.next”. Setelah itu, jika “front” menjadi “null” (artinya antrian sudah kosong), maka “rear” juga diset menjadi “null” untuk menandakan bahwa antrian sudah tidak memiliki elemen lagi.

|  |
| --- |
| public void tampilkanAntrean() {          if (front == null) {              System.out.println("Antrean kosong.");          } else {              Node temp = front;              while (temp != null) {                  System.out.print(temp.nama + " -> ");                  temp = temp.next;              }              System.out.println();          }      } |

Script di atas mendefinisikan *method* “tampilkanAntrean” dalam kelas “Queue”, yang berfungsi untuk menampilkan semua elemen dalam antrian. Pertama, *method* ini memeriksa apakah antrian kosong dengan memeriksa apakah “front” bernilai “null”. Jika antrian kosong, pesan “Antrean kosong.” akan ditampilkan. Jika antrian tidak kosong, maka *method* ini akan memulai iterasi dari elemen pertama “front” dan mencetak nama setiap jemaah dalam antrian. Iterasi dilakukan dengan memindahkan *pointer* “temp” ke elemen berikutnya “temp.next” hingga mencapai akhir antrian dimana “temp” menjadi “null”. Setelah semua elemen ditampilkan, sebuah baris baru dicetak untuk menandakan akhir dari *output* antrian.

1. **Code Class Sorting**

|  |
| --- |
| public class Sorting {        // Sorting berdasarkan tanggal pendaftaran menggunakan Bubble Sort      public static void sortByTanggal(LinkedList linkedList) {          if (linkedList.head == null || linkedList.head.next == null) {              return;          }          boolean swapped;          do {              swapped = false;              Node temp = linkedList.head;              while (temp != null && temp.next != null) {                  if (temp.TanggalPendaftaran.compareTo(temp.next.TanggalPendaftaran) > 0) {                      swapData(temp, temp.next);                      swapped = true;                  }                  temp = temp.next;              }          } while (swapped);          linkedList.display();      } |

Script di atas merupakan *method* “sortByTanggal” dalam kelas “Sorting” yang digunakan untuk menyortir data jemaah dalam “LinkedList” berdasarkan “tanggalPendaftaran” menggunakan algoritma Bubble Sort. Pada setiap iterasi, algoritma membandingkan tanggal pendaftaran dua node yang berdekatan. Jika tanggal pendaftaran pada node pertama lebih besar daripada “tanggalPendaftaran” pada node kedua, maka data kedua node tersebut akan dipertukarkan menggunakan *method* “swapData”. Proses ini berlanjut hingga tidak ada lagi data yang perlu dipertukarkan yang menunjukkan bahwa daftar sudah terurut. Setelah proses pengurutan selesai, daftar jemaah yang telah terurut akan ditampilkan dengan memanggil *method* “display()”.

|  |
| --- |
| // Sorting berdasarkan nama menggunakan Bubble Sort      public static void sortByNama(LinkedList linkedList) {          if (linkedList.head == null || linkedList.head.next == null) {              return;          }          boolean swapped;          do {              swapped = false;              Node temp = linkedList.head;              while (temp != null && temp.next != null) {                  if (temp.Nama.compareTo(temp.next.Nama) > 0) {                      swapData(temp, temp.next);                      swapped = true;                  }                  temp = temp.next;              }          } while (swapped);          linkedList.display();      } |

Script di atas adalah *method* “sortByNama” dalam kelas “Sorting” yang digunakan untuk menyortir data jemaah dalam “LinkedList” berdasarkan nama menggunakan algoritma Bubble Sort. Pada setiap iterasi, algoritma membandingkan nama dua node yang berdekatan. Jika nama pada node pertama lebih besar daripada nama pada node kedua, maka data kedua node tersebut dipertukarkan menggunakan *method* “swapData”. Proses ini berlanjut hingga tidak ada data yang perlu dipertukarkan lagi, yang menandakan bahwa daftar sudah terurut. Setelah pengurutan selesai, daftar jemaah yang telah terurut berdasarkan nama akan ditampilkan menggunakan *method* “display()”.

|  |
| --- |
| // Sorting berdasarkan nama menggunakan Bubble Sort      public static void sortByNoPendaftaran(LinkedList linkedList) {          if (linkedList.head == null || linkedList.head.next == null) {              return;          }          boolean swapped;          do {              swapped = false;              Node temp = linkedList.head;              while (temp != null && temp.next != null) {                  if (temp.nomorPendaftaran.compareTo(temp.next.nomorPendaftaran) > 0) {                      swapData(temp, temp.next);                      swapped = true;                  }                  temp = temp.next;              }          } while (swapped);          linkedList.display();      } |

Script di atas adalah *method* “sortByNoPendaftaran” dalam kelas “Sorting” yang digunakan untuk menyortir data jemaah dalam “LinkedList” berdasarkan nomor pendaftaran menggunakan algoritma Bubble Sort. Pada setiap iterasi, algoritma membandingkan nomor pendaftaran dua node yang berdekatan dengan menggunakan *method* “compareTo”. Jika nomor pendaftaran pada node pertama lebih besar dari node kedua, data kedua node tersebut akan dipertukarkan menggunakan *method* “swapData”. Proses ini berlanjut hingga tidak ada data yang perlu dipertukarkan lagi, yang menandakan bahwa daftar sudah terurut. Setelah pengurutan selesai, daftar jemaah yang telah terurut berdasarkan nomor pendaftaran akan ditampilkan menggunakan *method* “display()”.

|  |
| --- |
| // Ini metode untuk membantu proses penukaran data antara dua node  private static void swapData(Node node1, Node node2) {          String tempNomor = node1.nomorPendaftaran;          String tempTanggal = node1.TanggalPendaftaran;          Boolean tempStatus = node1.StatusPembayaran;          String tempNama = node1.Nama;          String tempJenis = node1.JenisKelamin;          int tempUmur = node1.Umur;          TreeNode tempAhliWaris = node1.ahliWarisRoot;          node1.nomorPendaftaran = node2.nomorPendaftaran;          node1.TanggalPendaftaran = node2.TanggalPendaftaran;          node1.StatusPembayaran = node2.StatusPembayaran;          node1.Nama = node2.Nama;          node1.JenisKelamin = node2.JenisKelamin;          node1.Umur = node2.Umur;          node1.ahliWarisRoot = node2.ahliWarisRoot;          node2.nomorPendaftaran = tempNomor;          node2.TanggalPendaftaran = tempTanggal;          node2.StatusPembayaran = tempStatus;          node2.Nama = tempNama;          node2.JenisKelamin = tempJenis;          node2.Umur = tempUmur;          node2.ahliWarisRoot = tempAhliWaris;      } |

Script di atas merupakan *method* “swapData” yang digunakan untuk menukar data antara dua node dalam “LinkedList”. *Method* ini menerima dua parameter yaitu “node1” dan “node2”, yang merupakan dua node yang datanya akan ditukar. Pertama-tama, nilai-nilai data dari “node1” disalin ke variabel sementara. Kemudian, data dari “node2” dipindahkan ke “node1”. Setelah itu, data yang telah disalin ke variabel sementara akan dipindahkan ke “node2”. Dengan cara ini, data dari kedua node tersebut akan saling dipertukarkan.

1. **Code Class Searching**

|  |
| --- |
| import java.util.Scanner;  public class Searching {      // Mencari data jemaah berdasarkan nomor pendaftaran menggunakan linear search      public Node cariJemaah(LinkedList linkedList, String nomorPendaftaran) {          Node temp = linkedList.head;          while (temp != null) {              if (temp.nomorPendaftaran.equals(nomorPendaftaran)) {                  System.out.println("Data ditemukan:");                  System.out.println("Nomor Pendaftaran: " + temp.nomorPendaftaran);                  System.out.println("Tanggal Pembayaran: " + temp.TanggalPendaftaran);                  System.out.println("Status Pembayaran: " + temp.StatusPembayaran);                  System.out.println("Nama Pendaftar: " + temp.Nama);                  System.out.println("Jenis Kelamin: " + temp.JenisKelamin);                  System.out.println("Umur: " + temp.Umur);                  return temp;              }              temp = temp.next;          }          System.out.println("Jemaah dengan nomor pendaftaran " + nomorPendaftaran + " tidak ditemukan.");          System.out.println("Silahkan cek kembali data yang di-inputkan.");          return null;      } |

Script di atas merupakan *method* “cariJemaah” pada kelas “Searching”, yang digunakan untuk mencari data jemaah dalam “LinkedList” berdasarkan “nomorPendaftaran” dengan menggunakan algoritma pencarian linier (linear search). *Method* ini menerima dua parameter “linkedList” yang berisi daftar jemaah dan “nomorPendaftaran” yang merupakan nomor pendaftaran yang ingin dicari. Proses pencarian dimulai dengan memeriksa apakah node yang sedang diperiksa “temp” memiliki nomor pendaftaran yang sesuai. Jika ditemukan, *method* ini akan menampilkan informasi terkait jemaah tersebut. Jika node dengan nomor pendaftaran yang dicari tidak ditemukan, *method* ini akan mengeluarkan pesan bahwa data tidak ditemukan dan mengembalikan “null”.

|  |
| --- |
| //buat bantu nyari parent cucu      public static TreeNode findParentByName(TreeNode node, String nama) {          if (node == null) {              return null;          }            if (node.namaAhliWaris.equalsIgnoreCase(nama)) {              return node;          }            TreeNode sibling = findParentByName(node.nextSibling, nama);          if (sibling != null) {              return sibling;          }            return findParentByName(node.firstChild, nama);      } |

Script di atas merupakan *method* “findParentByName” pada kelas “TreeNode”, yang digunakan untuk mencari parent atau induk dari sebuah node berdasarkan nama ahli waris yang diberikan. *Method* ini bekerja secara rekursif, dimulai dari node akar “root”. Jika node saat ini memiliki “nama” yang cocok dengan “nama” yang dicari, maka node ini dianggap sebagai parent dan akan dikembalikan. Jika tidak, *method* akan melanjutkan pencarian ke saudara “sibling” berikutnya dari node yang sedang diperiksa dengan memanggil *method* “findParentByName” pada “sibling”. Jika tidak ditemukan di “sibling”, pencarian dilanjutkan pada anak “child” dari node yang sedang diperiksa dengan memanggil *method* tersebut pada “firstChild”.

|  |
| --- |
| // Ini dipake buat update data jemaah, semisal ada kesalahan input atau penggantian jemaah manual oleh ahli waris      public static void updateData(LinkedList list, String nomorPendaftaran) {          Scanner scanner = new Scanner(System.in);          Searching data = new Searching();          Node temp = data.cariJemaah(list, nomorPendaftaran);          if (temp == null) return;          System.out.println("\n===========================");          System.out.println("      UPDATE DATA JEMAAH     ");          System.out.println("===========================\n");          System.out.println("Pilih data yang ingin diubah:");          System.out.println("1. Nama Jemaah");          System.out.println("2. Status Pembayaran");          System.out.println("3. Jenis Kelamin");          System.out.println("4. Umur");          System.out.println("5. Tanggal Pembayaran");          System.out.println("6. Ahli Waris");          System.out.print("\nMasukkan pilihan (1-6): ");          int pilihan = scanner.nextInt();          scanner.nextLine();          if (pilihan == 0) return;          System.out.println("\n-------------------------------------");          switch (pilihan) {              case 1:                  System.out.println("\n=============================");                  System.out.println("      UPDATE NAMA JEMAAH     ");                  System.out.println("=============================");                  System.out.print("Masukkan nama baru: ");                  temp.Nama = scanner.nextLine();                  System.out.println("\nNama berhasil diubah.");                  break;              case 2:                  System.out.println("\n=============================");                  System.out.println("  UPDATE STATUS PEMBAYARAN  ");                  System.out.println("=============================");                  System.out.print("Masukkan status pembayaran baru (true/lunas, false/belum): ");                  temp.StatusPembayaran = scanner.nextBoolean();                  System.out.println("\nStatus pembayaran berhasil diubah.");                  break;              case 3:                  System.out.println("\n=============================");                  System.out.println("     UPDATE JENIS KELAMIN    ");                  System.out.println("=============================");                  System.out.print("Masukkan jenis kelamin baru: ");                  temp.JenisKelamin = scanner.nextLine();                  System.out.println("\nJenis kelamin berhasil diubah.");                  break;              case 4:                  System.out.println("\n=============================");                  System.out.println("        UPDATE UMUR          ");                  System.out.println("=============================");                  System.out.print("Masukkan umur baru: ");                  temp.Umur = scanner.nextInt();                  System.out.println("\nUmur berhasil diubah.");                  break;              case 5:                  System.out.println("\n=============================");                  System.out.println("  UPDATE TANGGAL PEMBAYARAN  ");                  System.out.println("=============================");                  System.out.print("Masukkan tanggal pembayaran baru (yyyy-MM-dd): ");                  scanner.nextLine();                  temp.TanggalPendaftaran = scanner.nextLine();                  System.out.println("\nTanggal pembayaran berhasil diubah.");                  break;                  case 6:                      System.out.println("\n=============================");                      System.out.println("      UPDATE AHLI WARIS     ");                      System.out.println("=============================");                      System.out.print("Apakah Anda ingin menambah ahli waris baru atau hanya memperbarui ahli waris yang ada? (tambah/update): ");                      String jawaban = scanner.nextLine().trim().toLowerCase();                      if (jawaban.equals("tambah")) {                          if (temp.ahliWarisRoot == null) {                              System.out.print("Masukkan nama ahli waris pertama: ");                              String newAhliWaris = scanner.nextLine();                              temp.ahliWarisRoot = new TreeNode(newAhliWaris);                              System.out.println("Ahli waris pertama berhasil ditambahkan.");                          } else {                              System.out.println("Daftar ahli waris saat ini:");                              int index = 1;                              TreeNode currentNode = temp.ahliWarisRoot;                              while (currentNode != null) {                                  System.out.println(index + ". " + currentNode.namaAhliWaris);                                  if (currentNode.firstChild != null) {                                      TreeNode childNode = currentNode.firstChild;                                      int childIndex = 1;                                      while (childNode != null) {                                          System.out.println("|\_" + childIndex + " " + childNode.namaAhliWaris);                                          childNode = childNode.nextSibling;                                          childIndex++;                                      }                                  }                                  currentNode = currentNode.nextSibling;                                  index++;                              }                              System.out.print("\nMasukkan nama ahli waris baru: ");                              String newAhliWaris = scanner.nextLine();                                System.out.print("Apakah ahli waris baru akan menjadi anak atau cucu? (anak/cucu): ");                              String posisi = scanner.nextLine().trim().toLowerCase();                              if (posisi.equals("anak")) {                                  temp.tambahAhliWaris(newAhliWaris, false);                                  System.out.println("Ahli waris baru berhasil ditambahkan sebagai anak.");                              } else if (posisi.equals("cucu")) {                                  System.out.println("Masukkan nama parent: ");                                  String namaParent = scanner.nextLine();                                    TreeNode parentNode = findParentByName(temp.ahliWarisRoot, namaParent);                                    if (parentNode != null) {                                      System.out.println("Parent ditemukan: " + parentNode.namaAhliWaris);                                      parentNode.addChild(new TreeNode(newAhliWaris));                                      System.out.println("Cucu berhasil ditambahkan.");                                  } else {                                      System.out.println("Parent dengan nama " + namaParent + " tidak ditemukan.");                                  }                              } else {                                  System.out.println("Pilihan posisi tidak valid. Harus memilih antara 'anak' atau 'cucu'.");                              }                          }                      } else if (jawaban.equals("update")) {                          if (temp.ahliWarisRoot == null) {                              System.out.println("Belum ada ahli waris yang terdaftar.");                              System.out.print("Ingin menambah ahli waris? (y/n): ");                              char addHeir = scanner.next().charAt(0);                              scanner.nextLine();                              if (addHeir == 'y' || addHeir == 'Y') {                                  System.out.print("Masukkan nama ahli waris pertama: ");                                  String newAhliWaris = scanner.nextLine();                                  temp.ahliWarisRoot = new TreeNode(newAhliWaris);                                  System.out.println("Ahli waris pertama berhasil ditambahkan.");                              }                          } else {                              System.out.println("Daftar ahli waris saat ini:");                              int index = 1;                              TreeNode currentNode = temp.ahliWarisRoot;                              while (currentNode != null) {                                  System.out.println(index + ". " + currentNode.namaAhliWaris);                                  if (currentNode.firstChild != null) {                                      TreeNode childNode = currentNode.firstChild;                                      int childIndex = 1;                                      while (childNode != null) {                                          System.out.println("|\_" + childIndex + " " + childNode.namaAhliWaris);                                          childNode = childNode.nextSibling;                                          childIndex++;                                      }                                  }                                  currentNode = currentNode.nextSibling;                                  index++;                              }                              System.out.println("\nPilih ahli waris yang ingin diubah (masukkan nomor ahli waris): ");                              int pilihHeir = scanner.nextInt();                              scanner.nextLine();                                currentNode = temp.ahliWarisRoot;                              for (int i = 1; i < pilihHeir; i++) {                                  currentNode = currentNode.nextSibling;                              }                              if (currentNode != null) {                                  System.out.print("Masukkan nama ahli waris yang baru: ");                                  String newHeirName = scanner.nextLine();                                  currentNode.namaAhliWaris = newHeirName;                                  System.out.println("Ahli waris berhasil diubah.");                              } else {                                  System.out.println("Pilihan tidak valid.");                              }                          }                      } else {                          System.out.println("Pilihan tidak valid. Harus memilih antara 'tambah' atau 'update'.");                      }                      break;              default:                  System.out.println("\nPilihan tidak valid.");                  break;          }          System.out.println("\n-------------------------------------");          System.out.println("       Pembaruan data selesai!          ");          System.out.println("-------------------------------------\n");      }  } |

Script di atas merupak *Method* “updateData” digunakan untuk memperbarui data jemaah. Setelah mencari jemaah dengan “nomorPendaftaran” tertentu menggunakan fungsi “cariJemaah”, pengguna dapat memilih untuk mengubah data seperti “nama”, “statusPembayaran”, “jeniskelamin”, “umur”, “tanggalPembayaran”, atau “ahliWaris”.

1. **Code Class Main**

|  |
| --- |
| import java.time.LocalDate;  import java.util.Scanner;  public class Main {      public static void main(String[] args) {          Scanner scanner = new Scanner(System.in);          LinkedList link = new LinkedList();          Stack riwayatPembayaran = new Stack();          Queue antreanKeberangkatan = new Queue();          link.addData("A2897", "2018-09-11", true, "Ibrahim", "Laki-laki", 89, "Yanti", "Yanto", "Jantung");          link.addData("B3456", "2021-08-12", true, "Fatimah", "Perempuan", 75, "Yanta", "Yante", "Asma");          link.addData("C8909", "2016-07-10", true, "Ahmad", "Laki-laki", 35, "Yonge", "Yayat", "-");          link.addData("D1234", "2019-09-12", false, "Mutia", "Perempuan", 25, "Kuro", "Yaya", "-");          link.addData("E7890", "2024-06-28", true, "Indi", "Perempuan", 28, "mima", "titi", "Paru-paru basah");          link.addData("F8679", "2016-01-13", false, "Fiana", "Perempuan", 56, "Nasywa", "Nana", "Jantung");          link.addData("G0987", "2017-02-09", false, "Anis", "Perempuan", 35, "Malik", "Tya", "-");          link.addData("H4576", "2018-08-02", true, "Alysa", "Perempuan", 24, "Alya", "Ayi", "Asma");          link.addData("I9804", "2017-09-03", true, "Kaziya", "Perempuan", 38, "Adit", "Liza", "Paru-paru basah");          link.addData("J5378", "2022-05-01", true, "Rifqy", "Laki-laki", 23, "Hali", "Nayla", "-");          link.bayar("A2897", "2018-09-11", 56000000, "Cash");          link.bayar("B3456", "2021-08-12", 56000000, "Cash");          link.bayar("C8909", "2016-07-10", 56000000, "Cash");          link.bayar("D1234", "2019-09-12", 16000000, "Transfer");          link.bayar("E7890", "2024-06-28", 56000000, "Cash");          link.bayar("F8679", "2016-01-13", 38000000, "Cash");          link.bayar("G0987", "2017-02-09", 38000000, "Cash");          link.bayar("H4576", "2018-08-02", 56000000, "Cash");          link.bayar("I9804", "2017-09-03", 56000000, "Cash");          link.bayar("J5378", "2022-05-01", 56000000, "Cash");          System.out.println("  .-'''-.    \_\_\_\_     \_\_\_\_\_\_\_\_    \_\_\_\_     \_\_\_\_\_\_\_\_   .---.       ,-----.    .--.      .--. ");          System.out.println(" / \_     \\ .'  \_\_ `. |        | .'  \_\_ `. |        |  | ,\_|     .'  .-,  '.  |  |\_     |  | ");          System.out.println("(`' )/`--'/   '  \\  \\|   .----'/   '  \\  \\|   .----',-./  )    / ,-.|  \\ \_ \\ | \_( )\_   |  | ");          System.out.println("(\_ o \_).   |\_\_\_|  /  ||  \_|\_\_\_\_ |\_\_\_|  /  ||  \_|\_\_\_\_ \\  '\_ '`) ;  \\  '\_ /  | :|(\_ o \_)  |  | ");          System.out.println(" (\_,\_). '.    \_.-`   ||\_( )\_   |   \_.-`   ||\_( )\_   | > (\_)  ) |  \_`,/ \\ \_/  || (\_,\_) \\ |  | ");          System.out.println(".---.  \\  :.'   \_    |(\_ o.\_)\_\_|.'   \_    |(\_ o.\_)\_\_|(  .  .-' : (  '\\\_/ \\   ;|  |/    \\|  | ");          System.out.println("\\    `-'  ||  \_( )\_  ||(\_,\_)    |  \_( )\_  ||(\_,\_)     `-'`-'|\_\_\_\\ `\"/  \\ ) / |  '  /\\  `  | ");          System.out.println(" \\       / \\ (\_ o \_) /|   |     \\ (\_ o \_) /|   |       |        \'. \\\_/``\".'  |    /  \\    | ");          System.out.println("  `-...-'   '.(\_,\_).' '---'      '.(\_,\_).' '---'       `--------`  '-----'    `---'    `---` ");          while (true) {              System.out.println("==========================================");              System.out.println("|+|        MENU PELAYANAN JEMAAH       |+|");              System.out.println("==========================================");              System.out.println("| No |             Menu                 |");              System.out.println("==========================================");              System.out.println("| 0  | Keluar                           |");              System.out.println("==========================================");              System.out.println("| 1  | Tambah Jemaah                    |");              System.out.println("==========================================");              System.out.println("| 2  | Update Data Jemaah               |");              System.out.println("==========================================");              System.out.println("| 3  | Hapus Jemaah                     |");              System.out.println("==========================================");              System.out.println("| 4  | Tampilkan Calon Jemaah           |");              System.out.println("==========================================");              System.out.println("| 5  | Tambah Transaksi Pembayaran      |");              System.out.println("==========================================");              System.out.println("| 6  | Batalkan Transaksi Pembayaran    |");              System.out.println("==========================================");              System.out.println("| 7  | Tampilkan Riwayat Pembayaran     |");              System.out.println("==========================================");              System.out.println("| 8  | Tambah Ke Antrean Keberangkatan  |");              System.out.println("==========================================");              System.out.println("| 9  | Proses Keberangkatan Jemaah      |");              System.out.println("==========================================");              System.out.println("| 10 | Tampilkan Antrean                |");              System.out.println("==========================================");              System.out.println();              System.out.print("Pilih menu (0-10): ");              int pilihan = scanner.nextInt();              scanner.nextLine();              if (pilihan == 0) break;                switch (pilihan) {                  case 1:                  System.out.print("Nama Jemaah: ");                      String nama = scanner.nextLine();                      System.out.print("Nomor Pendaftaran: ");                      String nomor = scanner.nextLine();                      String tanggal = LocalDate.now().toString();                      System.out.print("Status Pembayaran (true/lunas, false/belum): ");                      boolean status = scanner.nextBoolean();                      scanner.nextLine();                      System.out.print("Jenis kelamin: ");                      String jenis = scanner.nextLine();                      System.out.print("Umur: ");                      int umur = scanner.nextInt();                      scanner.nextLine();                      System.out.print("Penyakit: ");                      String penyakit = scanner.nextLine();                      System.out.print("Nama Ahli Waris 1: ");                      String ahliWarisPertama = scanner.nextLine();                      System.out.print("Nama Ahli Waris 2: ");                      String ahliWarisKedua = scanner.nextLine();                      link.addData(nomor, tanggal, status, nama, jenis, umur, ahliWarisPertama, ahliWarisKedua, penyakit);                      System.out.println("Data jemaah berhasil ditambahkan!");                      break;                  case 2:                      System.out.println("=====================================================");                      System.out.println("   UPDATE DATA JEMAAH BERDASARKAN NOMOR PENDAFTARAN   ");                      System.out.println("=====================================================");                      System.out.print("Masukkan nomor pendaftaran jemaah yang ingin di-update datanya: ");                      String update = scanner.nextLine();                      if(!link.cekNomorPendaftaran(update)){                          System.out.println("Nomor pendaftaran tidak ditemukan.");                          break;                      }                      Searching.updateData(link, update);                      break;                      case 3:                      System.out.println("=====================================================");                      System.out.println("   HAPUS DATA JEMAAH BERDASARKAN NOMOR PENDAFTARAN   ");                      System.out.println("=====================================================");                      System.out.print("Masukkan nomor pendaftaran jemaah yang ingin dihapus: ");                      String hapusPendaftaran = scanner.nextLine();                      System.out.print("Apakah Anda ingin menghapus seluruh data? (y/n): ");                      String hapusSeluruhData = scanner.nextLine();                      if (hapusSeluruhData.equalsIgnoreCase("y")) {                          link.hapusJemaah(hapusPendaftaran, true); // true jika Anda ingin menghapus seluruh data                      } else {                          link.hapusJemaah(hapusPendaftaran, false);                      } // false jika Anda tidak ingin menghapus seluruh data                      break;                  case 4:                      System.out.println("==============================================");                      System.out.println("              FILTER DATA JEMAAH              ");                      System.out.println("==============================================");                      System.out.println("1. Urutkan berdasarkan tanggal pendaftaran");                      System.out.println("2. Urutkan berdasarkan nama");                      System.out.println("3. Urutkan berdasarkan nomor pendaftaran");                      System.out.println("==============================================");                      System.out.print("Pilih menu (1-3): ");                        int pilihanFilter = scanner.nextInt();                      scanner.nextLine();                      if (pilihanFilter == 1){                          Sorting.sortByTanggal(link);                      } else if (pilihanFilter == 2){                          Sorting.sortByNama(link);                      }else if (pilihanFilter == 3){                          Sorting.sortByNoPendaftaran(link);                      } else {                          System.out.println("Input tidak valid.");                      }                      break;                      case 5:                      System.out.println("==============================================");                      System.out.println("          TANGGAL DAN JUMLAH PEMBAYARAN      ");                      System.out.println("==============================================");                      System.out.print("Nomor Pendaftaran: ");                      String nomorPendaftaran = scanner.nextLine();                      if(!link.cekNomorPendaftaran(nomorPendaftaran)){                          System.out.println("Nomor pendaftaran tidak ditemukan.");                          break;                      }                      System.out.print("Tanggal Pembayaran (format: yyyy-mm-dd): ");                      String tanggal2 = scanner.nextLine();                      System.out.println("----------------------------------------------");                      System.out.print("Jumlah Pembayaran: ");                      int jumlah = scanner.nextInt();                      scanner.nextLine();                      System.out.print("Metode Pembayaran: ");                      String metode = scanner.nextLine();                      link.bayar(nomorPendaftaran, tanggal2, jumlah, metode);                      break;                  case 6:                      riwayatPembayaran.pop();                      break;                  case 7:                      link.showPaymentHistory();                      break;                  case 8:                      System.out.println("==============================================");                      System.out.println("      MASUKKAN NAMA JEMAAH KE ANTREAN        ");                      System.out.println("==============================================");                      System.out.print("Nama Jemaah yang akan dimasukkan antrean: ");                      String antreanNama = scanner.nextLine();                      if(!link.cekNama(antreanNama)){                           System.out.println("Nama Jemaah tidak ditemukan.");                          break;                      }                      antreanKeberangkatan.enqueue(antreanNama);                      break;                  case 9:                      antreanKeberangkatan.dequeue();                      break;                  case 10:                      antreanKeberangkatan.tampilkanAntrean();                      break;                  default:                      System.out.println("Pilihan tidak valid.");              }          }          scanner.close();      }  } |

Script di atas merupakan script dari kelas “Main” dimana code mengimpor berbagai kelas dan struktur data yag akan digunakan dalam program, termasuk “Scanner” untuk input pengguna, serta kelas untuk mengimplementasikan *linked* *list*, *queue*, dan *stack*. Class “Main” memiliki menu yang dapat dipilih oleh pengguna yaitu “Tambah Jemaah”, “Update Jemaah”, “Hapus Jemaah”, “Tampilkan Calon Jemaah”, “Tambah Transaksi Pembayaran”, “Batalkan Transaksi Pembayaran”, “Tampilkan Riwayat Pembayaran”, “Tambah ke Antrean Keberangkatan”, “Proses Keberangkatan Jemaah”, dan “Tampilkan Antrean”. Ketika pengguna memilih “Tambah Jemaah”, program akan meminta input detail seperti “nama”, “nomorPendaftaran”, ‘StatusPembayaran”, “jenisKelamin”, “umur”, “penyakit”, dan “ahliWaris” lalu data tersebut disimpan dalam *linked list*. Jika memilih “Update Jemaah”, pengguna dapat memperbarui data jemaah berdasarkan “nomorPendaftaran” yang dimasukkan. Pada opsi “Hapus Jemaah”, pengguna bisa menghapus data tertentu atau seluruh data dari *linked list*. Saat memilih “Tampilkan Calon Jemaah”, program akan menampilkan daftar jemaah yang dapat diurutkan berdasarkan “tanggalPendaftaran”, “nama”, atau “nomorPendaftaran”. Opsi “Tambah Transaksi Pembayaran” memungkinkan pengguna mencatat “tanggal”, “jumlah”, dan “metodePembayaran”, yang kemudian disimpan di dalam stack. Sementara “Batalkan Transaksi Pembayaran” akan menghapus transaksi terakhir dari stack. Jika pengguna memilih “Tampilkan Riwayat Pembayaran”, semua transaksi yang tercatat akan ditampilkan. Pilihan “Tambah ke Antrean Keberangkatan” menambahkan nama jemaah ke dalam queue keberangkatan, dan “Proses Keberangkatan Jemaah” menghapus nama jemaah pertama dalam antrean, menandai keberangkatan mereka. Terakhir, opsi “Tampilkan Antrean” digunakan untuk melihat daftar jemaah yang masih menunggu dalam antrean keberangkatan.

1. ***Output* Program**

|  |
| --- |
|  |

**Gambar 1.1**

Dari **Gambar 1.1** dapat diketahui bahwa gambar tersebut merupakan *output* menu utama dari program safaflow. Menu utama memiliki 10 opsi yaitu 1. Tambah Jemaah, 2. Update Jemaah, 3. Hapus Jemaah, 4. Tampilkan Calon Jemaah, 5. Tambah Transaksi Pembayaran, 6. Batalkan Transaksi Pembayaran, 7. Tampilkan Riwayat Pembayaran, 8. Tambah ke Antrean Keberangkatan, 9. Proses Keberangkatan Jemaah, dan 10. Tampilkan Antrean.

|  |
| --- |
|  |

**Gambar 1.2**

Dari **Gambar 1.2** dapat diketahui bahwa gambar tersebut merupakan hasil *output* jika pengguna memilih opsi 1. Pengguna akan diminta untuk menginputkan data dari Jemaah baru yang mendaftar yaitu nomor pendaftaran, nama, umur, jenis kelamin, status pembayaran, penyakit dan ahli waris.

|  |
| --- |
|  |

**Gambar 1.3**

Pada **Gambar 1.3** dapat diketahui bahwa gambar tersebut merupakan *output* ketika pengguna memilih 2. Pengguna akan diminta untuk menginputkan nomor pedaftaran Jemaah yang akan diupdate dan ingin mengupdate informasi apa dari Jemaah tersebut.

|  |
| --- |
|  |

**Gambar 1.4**

Pada **Gambar 1.4** dapat diketahui bahwa gambar tersebut merupakan *output* ketika pengguna memilih opsi 3 maka pengguna akan diminta untuk memasukkan nomor pedaftaran Jemaah yang ingin dihapus dan pengguna juga diberi pilihan apakah ingin menghapus seluruh data dari Jemaah yang dipilih atau hanya menghapus jemaahnya saja dan digantikan oleh ahli warisnya.

|  |
| --- |
|  |

**Gambar 1.5**

Pada **Gambar 1.5** di atas dapat diketahui bahwa gambar tersebut merupakan *output* ketika pengguna memilih opsi 4. Program akan menampilkan semua data dan informasi dari Jemaah yang telah terdaftar pada sistem.

|  |
| --- |
|  |

**Gambar 1.6**

Pada **Gambar 1.6** di atas dapat diketahui bahwa gambar tersebut merupakan *output* ketika pengguna memilih opsi 5. Pengguna akan diminta untuk memasukkan tanggal pembayaran yang sedang di proses, kemudian menambahkan jumlah pembayaran yang dilakukan dan *method* pembayaran yang dilakukan.

|  |
| --- |
| A screenshot of a menu  Description automatically generated |

**Gambar 1.7**

Pada **Gambar 1.7** di atas, dapat diketahui bahwa gambar tersebut merupakan *output* ketika penggua memilih opsi 6. Saat pengguna memilih opsi 6 maka secara otomatis program akan menghapus pembayaran terakhir yang diinputkan oleh pengguna.

|  |
| --- |
|  |

**Gambar 1.8**

Pada **Gambar 1.8** di atas, dapat diketahui bahwa gambar tersebut merupakan *output* ketika pengguna memilih opsi 7, saat pengguna memilih opsi 7 maka program akan menampilkan Riwayat pembayaran terakhir yang telah diinputkan oleh pengguna.

|  |
| --- |
| A screenshot of a menu  Description automatically generated |

**Gambar 1.9**

Pada **Gambar 1.9**, dapat diketahui bahwa gambar tersebut merupakan *output* ketika pengguna memilih opsi 8, saat pengguna memlih opsi 8 maka maka pengguna akan diminta untuk memasukkan nama Jemaah yang akan ditambahkan dalam antrian keberangkatan Haji.

|  |
| --- |
| A screenshot of a menu  Description automatically generated |

**Gambar 1.10**

Pada **Gambar 1.10**, dapat di ketahui bahwa gambar tersebut merupakan *output* ketika pengguna memilih opsi 9. Saat memilih opsi 9 program akan secara otomatis menampilkan Jemaah yang telah di proses keberangkatannya. Berdasarkan gambar tersebut, jamaah yang sedang diperoses keberangkatanya yaitu wahyu.

|  |
| --- |
|  |

**Gambar 1.11**

Pada **Gambar** **1.11** di atas, dapat diketahui bahwa gambar tersebut merupakan *output* ketika pengguna memilih opsi 10, saat memilih opsi 10 program akan menampilkan antrian Jemaah yang akan diproses.

1. **Kesimpulan**
2. **Referensi**
3. M. P. Putri, G. Barovih, R. A. Azdy, Y. Yuniansyah, A. Saputra, Y. Sriyeni, ... & F. T. Admojo, “Algoritma dan Struktur Data”, 2022.
4. Y. Arvita, “Perancangan Sistem Informasi Pelayanan Travel Haji Dan Umroh,” *Jurnal Ilmiah Media Sisfo*, vol. 15, pp. 9-18, 2021.
5. V. Jaiswal and J. A. Ray, “N-ary Treap: A novel data-structure for balanced n-ary tree,” *Authorea Preprints*, 2024.