LAPORAN PRAKTIKUM (5) STRUKTUR DATA

Sebuah gambar berisi lambang, logo, makanan, ilustrasi

Konten yang dihasilkan AI mungkin salah.

Oleh:

Nama : Arkan Ubaidillah Warman

NIM : 2411537001

Dosen Pengampu : Dr. Wahyudi MT .

STRUKTUR DATA

(SLL)

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERITAS ANDALAS

PADANG

1. **PENDAHULUAN**
2. **Latar Belakang**

Dalam dunia pemrograman, penyimpanan data yang efisien dan fleksibel sangat penting, terutama ketika data sering bertambah dan berkurang. Salah satu solusi dari permasalahan ini adalah struktur data linked list, khususnya single linked list. Tidak seperti array yang memiliki ukuran tetap, linked list memungkinkan manipulasi data secara dinamis. Oleh karena itu, pemahaman tentang single linked list sangat penting bagi pengembang perangkat lunak maupun ilmuwan komputer.

1. **Tujuan Percobaan**

* Memahami konsep dasar dan struktur dari **single linked list**.
* Mengimplementasikan operasi dasar single linked list seperti: **penambahan (insert), penghapusan (delete), dan penelusuran (traversal)**.
* Melatih keterampilan logika dalam menyusun node dan pointer dalam memori secara dinamis.
* Mengetahui kelebihan dan keterbatasan dari penggunaan linked list dibandingkan array.

1. **Landasan Teori**

**Single Linked List** adalah salah satu jenis struktur data linier di mana setiap elemen (disebut **node**) menyimpan dua informasi:

* **Data**: nilai yang disimpan.
* **Pointer/Link**: referensi ke node berikutnya dalam list.

Operasi dasar dalam single linked list:

* **Insert**: Menambahkan node di awal, tengah, atau akhir.
* **Delete**: Menghapus node tertentu dari list.
* **Traversal**: Menelusuri semua node dari awal hingga akhir.

**Kelebihan:**

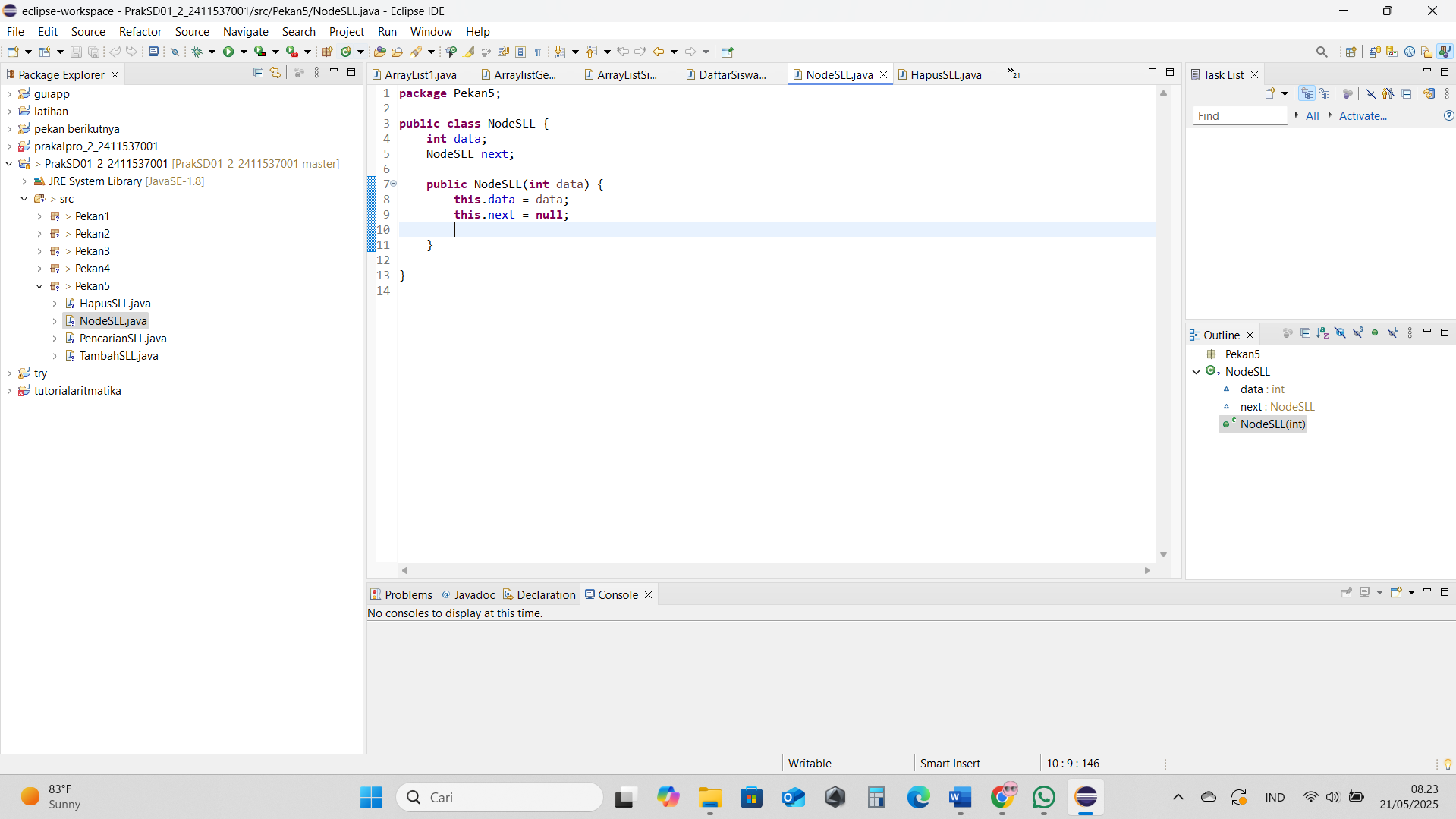
* Ukuran dinamis, efisien untuk data yang sering berubah.
* Lebih hemat memori jika dibandingkan array besar yang banyak kosong.

**Kekurangan:**

* Akses data lebih lambat karena harus dilakukan secara berurutan (tidak bisa langsung loncat ke index tertentu).
* Mengelola pointer lebih kompleks dibanding array.

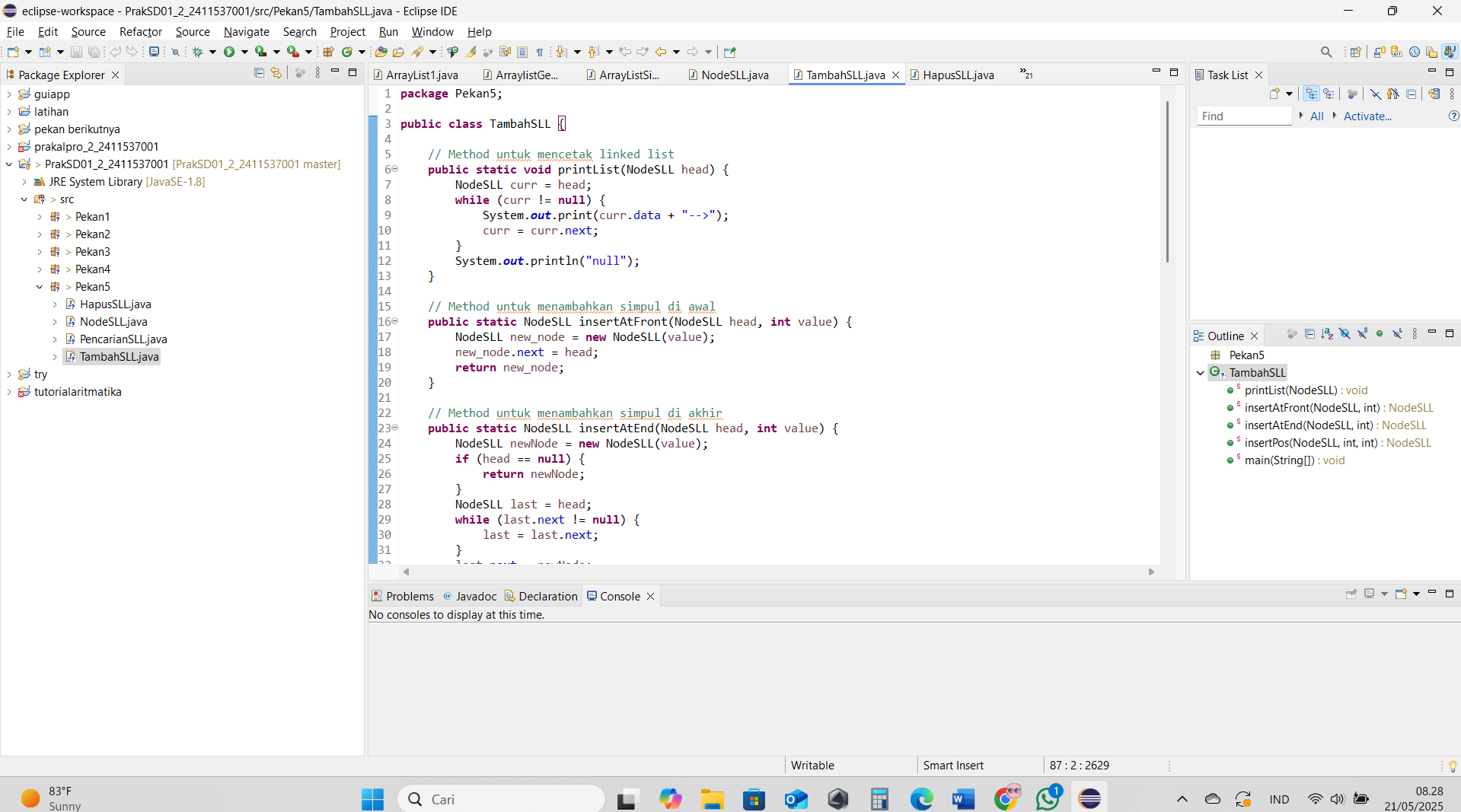
.

1. **LANGKAH-LANGKAH**
   1. -NodeSLL.java



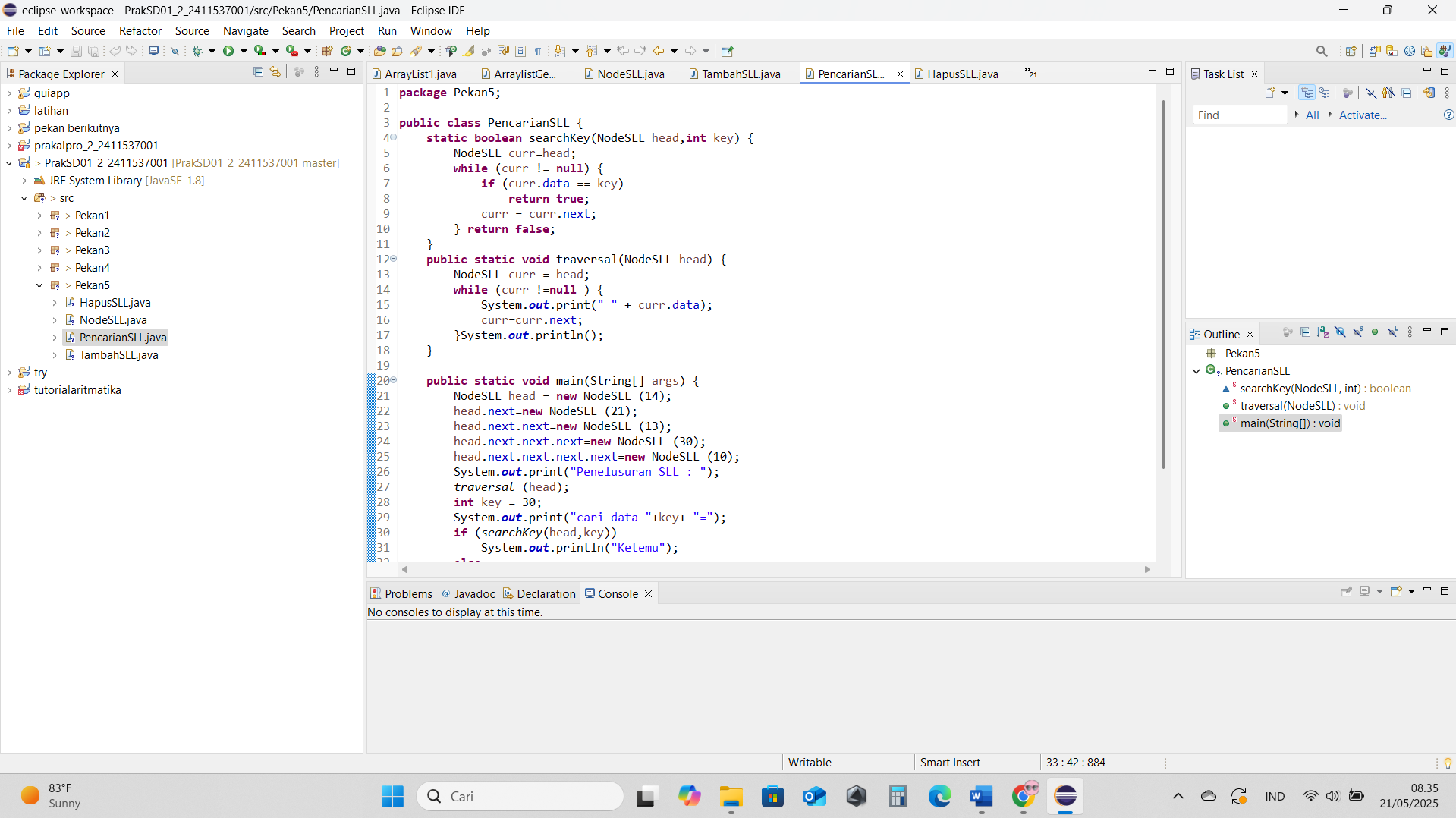
**Penjelasan:**

* **int data**: Menyimpan nilai dari node.
* **NodeSLL next**: Merupakan referensi ke node berikutnya.
* **Konstruktor NodeSLL(int data)**:
  1. Mengatur nilai awal data sesuai parameter.
  2. Inisialisasi next dengan null, artinya node ini belum menunjuk ke node lain.
  3. -TambahSLL.java



**Penjelasan Program**

* printList(): Menampilkan seluruh isi linked list dari head hingga null.
* insertAtFront(): Menambahkan simpul di awal dengan menunjuk ke head lama.
* insertAtEnd(): Menambahkan simpul di akhir dengan iterasi ke node terakhir.
* insertPos(): Menambahkan simpul di posisi yang diinginkan dengan iterasi ke posisi sebelumnya.
* main(): Inisialisasi list awal dengan 2→3→5→6, lalu menambahkan:
  1. Simpul di depan (1)
  2. Simpul di belakang (7)
  3. Simpul di posisi ke-4 (4)
  4. – PencarianSLL.java



**Penjelasan Program**

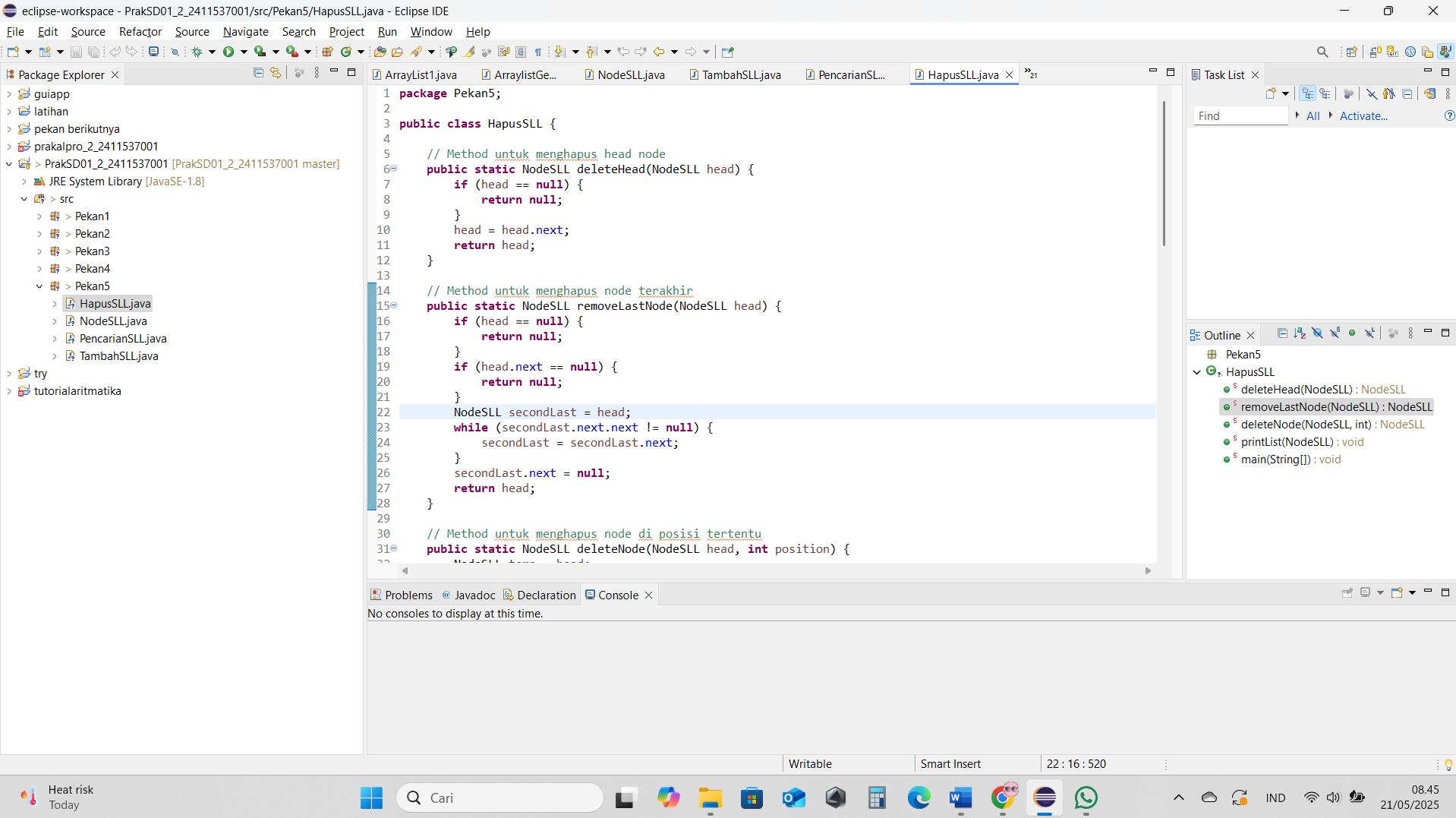
Program PencarianSLL.java merupakan implementasi operasi **pencarian data (search)** dalam struktur **Single Linked List (SLL)**.

Berikut penjelasan setiap bagian kode:

* **searchKey(NodeSLL head, int key)**  
  Method ini bertugas mencari apakah suatu nilai key terdapat dalam linked list.
  + Menggunakan **iterasi (loop while)** untuk mengecek satu per satu node.
  + Jika data dari node sama dengan key, maka return true.
  + Jika sampai akhir list tidak ditemukan, return false.

.

* **traversal(NodeSLL head)**  
  Method ini digunakan untuk menampilkan isi dari linked list.
  + Melakukan perulangan dari node pertama (head) hingga akhir (null).
  + Mencetak isi data dari setiap node.
* **main()**  
  Program utama yang:
  + 1. Membuat single linked list dengan isi: 14 → 21 → 13 → 30 → 10.
    2. Menampilkan seluruh isi list dengan traversal().
    3. Melakukan pencarian terhadap nilai 30 menggunakan searchKey().
    4. Menampilkan hasil pencarian (Ketemu atau Tidak ada).
  1. – HapusSLL.java



**Penjelasan Program**

Program HapusSLL.java digunakan untuk melakukan **penghapusan node** pada struktur data **Single Linked List (SLL)**. Program ini mendemonstrasikan tiga jenis operasi penghapusan:

* **deleteHead(NodeSLL head)**

Method ini digunakan untuk menghapus simpul pertama (head) dalam linked list.

* Jika list kosong (head == null), return null.
* Jika tidak, head akan digantikan oleh node berikutnya (head = head.next).

..

* **removeLastNode(NodeSLL head)**

Method ini menghapus node terakhir dari linked list.

* Jika list kosong atau hanya berisi satu simpul, hasilnya adalah null.
* Jika lebih dari satu node, dilakukan iterasi sampai node ke-dua terakhir (secondLast), lalu secondLast.next diset null.
* **deleteNode(NodeSLL head, int position)**

Method ini menghapus simpul di **posisi tertentu** (berdasarkan indeks ke-1).

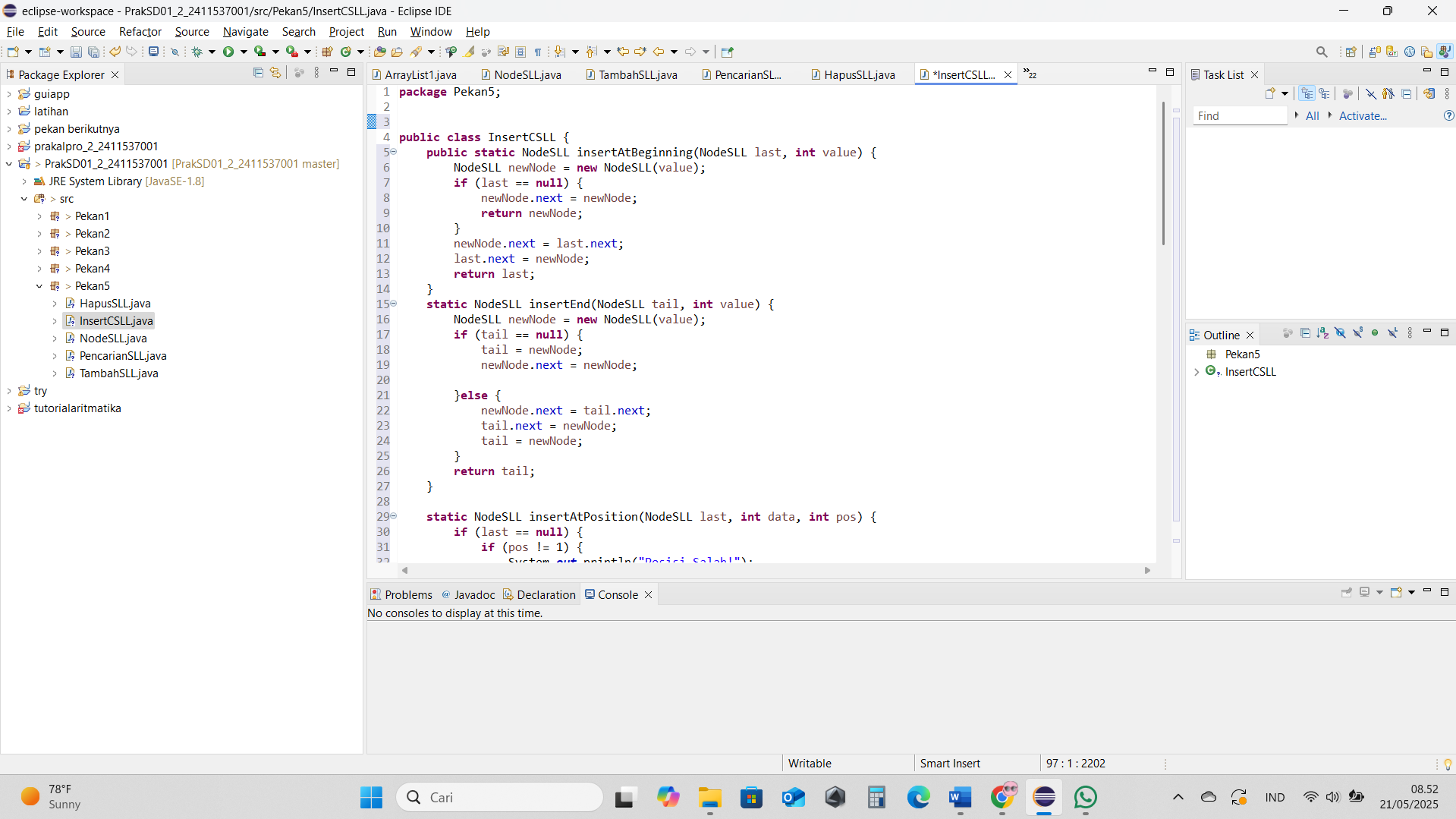
* Jika posisi adalah 1, maka sama seperti deleteHead.
* Jika posisi > 1, dilakukan perulangan menuju simpul ke-position - 1, kemudian node yang dihapus dilewati oleh node sebelumnya.
* Jika posisi tidak valid (melebihi panjang list), program menampilkan pesan “Data tidak ada”.
* **printList(NodeSLL head)**

Method ini digunakan untuk mencetak isi dari linked list dari awal hingga akhir.

* **main(String[] args)**

Program utama yang:

1. Membuat linked list awal: 1 → 2 → 3 → 4 → 5 → 6
2. Menghapus simpul pertama (1) → hasil: 2 → 3 → 4 → 5 → 6
3. Menghapus simpul terakhir (6) → hasil: 2 → 3 → 4 → 5
4. Menghapus simpul di posisi ke-2 (3) → hasil: 2 → 4 → 5
   1. – InsertCSLL.java



**Penjelasan Program**

Program InsertCSLL.java merupakan implementasi operasi **penyisipan node** dalam struktur data **Circular Single Linked List (CSLL)** atau **Linked List Sirkular**.

Pada circular linked list, node terakhir akan menunjuk kembali ke node pertama, membentuk lingkaran.

Program ini menyediakan tiga fungsi utama:

* **insertAtBeginning(NodeSLL last, int value)**
* Menambahkan simpul **di awal list sirkular**.
* Jika list kosong (last == null):
  + Simpul baru menunjuk ke dirinya sendiri.
  + Simpul tersebut dianggap sebagai node terakhir (last).
* Jika list sudah ada:
  + Simpul baru disisipkan setelah node terakhir (last) tapi sebelum head.
  + Artinya, simpul baru akan menjadi head baru.

**Return**: simpul terakhir (last) tetap tidak berubah.

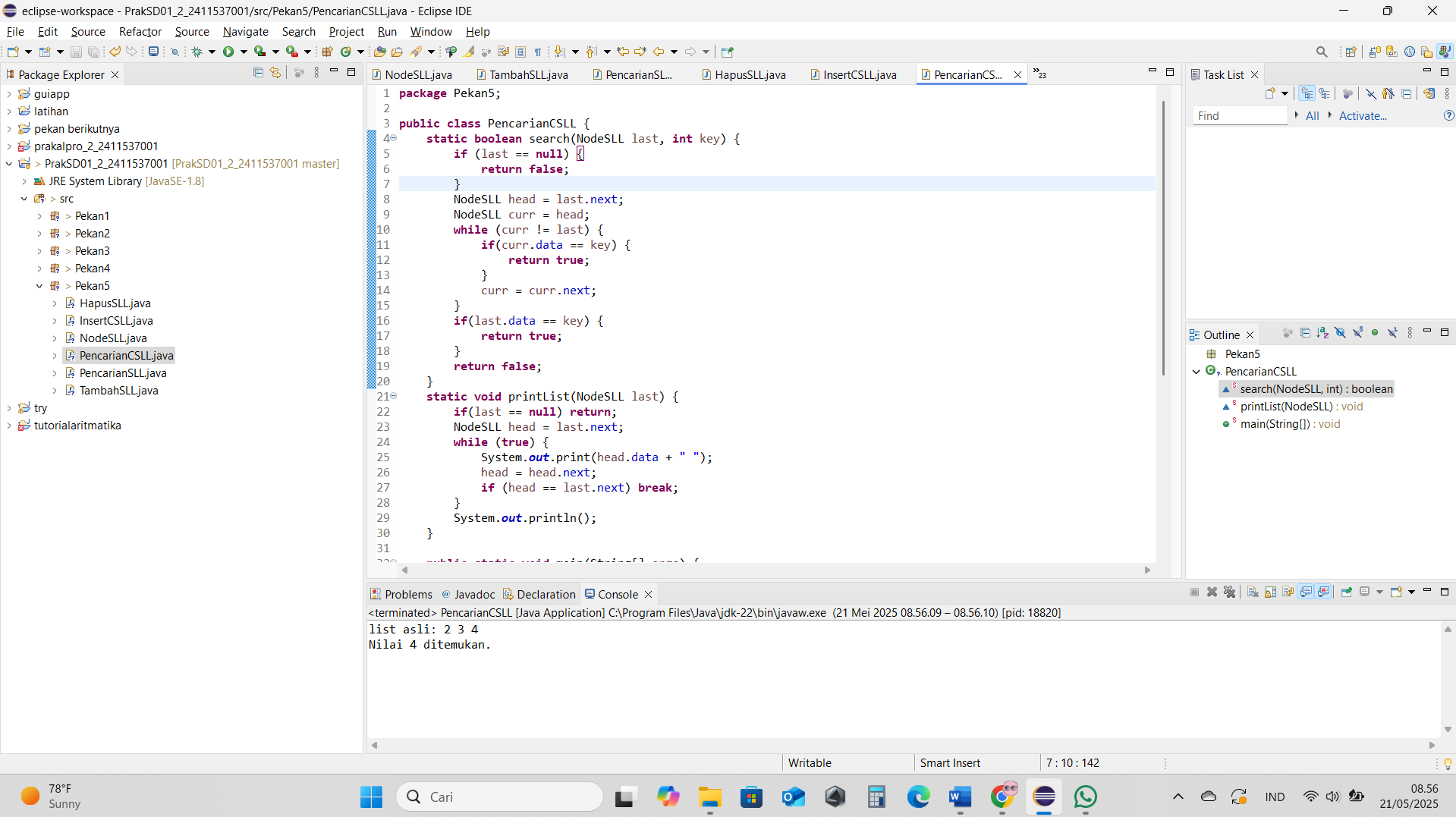
* **insertEnd(NodeSLL tail, int value)**
* Menambahkan simpul **di akhir list sirkular**, tepat setelah simpul terakhir.
* Jika list kosong:
  + Simpul pertama akan menunjuk ke dirinya sendiri.
* Jika list tidak kosong:
  + Simpul baru menunjuk ke node setelah tail (head).
  + Tail sebelumnya menunjuk ke node baru.
  + Tail di-update ke simpul baru.

**Return**: node baru sebagai tail baru (simpul terakhir).

* **insertAtPosition(NodeSLL last, int data, int pos)**
* Menambahkan simpul **pada posisi tertentu** dalam list.

..

* 1. – PencarianCSLL.java



**Penjelasan Program**

Program PencarianCSLL.java merupakan implementasi operasi **pencarian elemen** pada struktur data **Circular Single Linked List (CSLL)**, yaitu linked list di mana simpul terakhir menunjuk kembali ke simpul pertama, membentuk struktur melingkar.

* **search(NodeSLL last, int key)**

Method ini digunakan untuk **mencari sebuah nilai** (key) dalam list sirkular.

* Jika list kosong (last == null), langsung return false.
* Penelusuran dimulai dari node pertama (head = last.next).
* Loop dilakukan hingga kembali ke node pertama.
* Jika ditemukan node dengan nilai data == key, maka return true.
* Jika sampai akhir tidak ditemukan, return false.
* **printList(NodeSLL last)**

Method ini digunakan untuk **menampilkan semua elemen dalam list sirkular**.

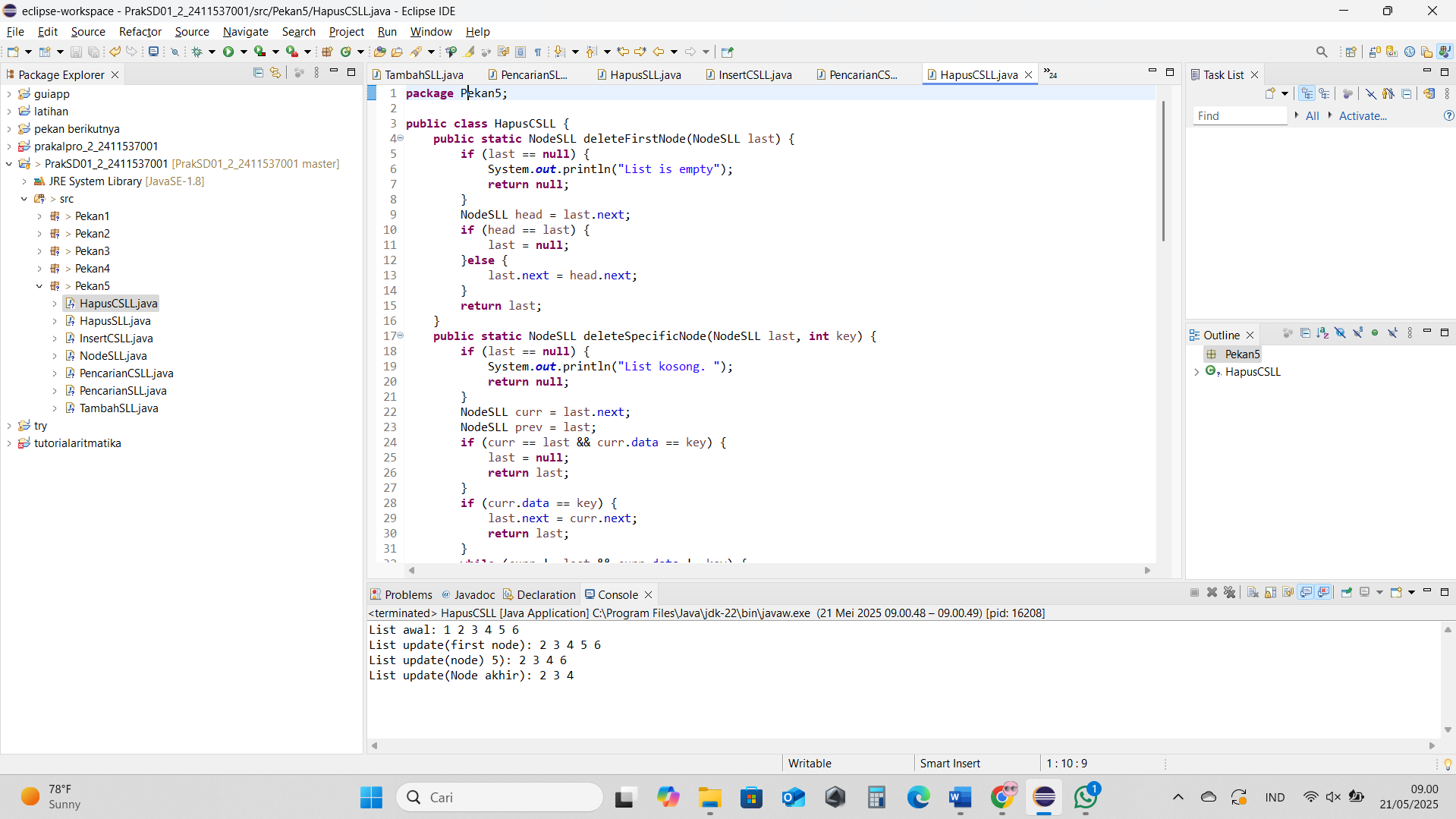
* Mulai dari node setelah last (yaitu head).
* Gunakan perulangan while (true) dan berhenti saat kembali ke node awal.
* Karena list berbentuk sirkular, tidak ada node yang bernilai null.

.

* **main(String[] args)**

Program utama berfungsi untuk:

1. Membuat circular linked list dengan isi: 2 → 3 → 4 → kembali ke 2.
   * first = 2, first.next = 3, first.next.next = 4
   * last = 4, dan last.next = first untuk membentuk lingkaran
2. Menampilkan isi list dengan printList(last).
3. Melakukan pencarian terhadap nilai 4 dengan search(last, key).
4. Menampilkan apakah nilai ditemukan atau tidak.
   1. – HapusCSLL



**Penjelasan Program**

Program HapusCSLL.java merupakan implementasi operasi **penghapusan simpul (node)** pada **Circular Single Linked List (CSLL)**, yaitu bentuk linked list di mana node terakhir menunjuk kembali ke node pertama.

Program ini terdiri dari tiga jenis penghapusan node:

* **deleteFirstNode(NodeSLL last)**

Fungsi ini digunakan untuk menghapus **simpul pertama (head)** dari circular linked list.

* Jika list kosong, maka ditampilkan pesan "List is empty".
* Jika hanya ada satu simpul (head == last), maka last diset menjadi null.
* Jika terdapat lebih dari satu simpul, maka last.next diperbarui ke head.next.

.

* **deleteSpecificNode(NodeSLL last, int key)**

Fungsi ini digunakan untuk menghapus **simpul dengan nilai tertentu** (key) dari list.

* Jika list kosong, maka ditampilkan pesan "List kosong".
* Jika hanya satu node dan itu adalah node yang dicari, last menjadi null.
* Jika node yang dihapus adalah head (last.next), maka last.next diarahkan ke node setelahnya.
* Jika node berada di tengah atau akhir:
  + Iterasi dilakukan untuk mencari node yang cocok.
  + Jika ditemukan:
    - prev.next diarahkan ke curr.next, sehingga node curr dihapus.
    - Jika node yang dihapus adalah last, maka last diperbarui menjadi prev.
  + Jika tidak ditemukan, tampilkan pesan bahwa node tidak ditemukan.
* **deleteLastNode(NodeSLL last)**

Fungsi ini digunakan untuk menghapus **simpul terakhir** dalam list.

* Jika list kosong, return null.
* Jika hanya satu simpul, list dikosongkan.
* Jika lebih dari satu simpul:
  + Iterasi dilakukan untuk menemukan simpul sebelum last.
  + curr.next (simpul kedua terakhir) akan menunjuk kembali ke head.
  + last diperbarui ke curr.
* **printList(NodeSLL last)**

Fungsi ini digunakan untuk menampilkan isi list secara melingkar.

* **main(String[] args)**

Berikut langkah-langkah yang dilakukan dalam fungsi main():

1. Membuat circular single linked list dengan isi: 1 → 2 → 3 → 4 → 5 → 6 → (kembali ke 1)
2. Menampilkan list awal.
3. Menghapus node pertama: hasilnya 2 → 3 → 4 → 5 → 6.
4. Menghapus node dengan data 5: hasilnya 2 → 3 → 4 → 6.
5. Menghapus node terakhir (6): hasilnya 2 → 3 → 4.
6. **KESIMPULAN**

Dari percobaan ini dapat disimpulkan bahwa **single linked list** adalah struktur data yang efisien untuk menyimpan dan memanipulasi data secara dinamis. Operasi-operasi seperti insert dan delete lebih fleksibel dibanding array. Namun, penggunaannya membutuhkan pemahaman tentang pointer dan manajemen memori. Single linked list sangat berguna dalam aplikasi yang membutuhkan pengolahan data yang sering berubah atau tidak diketahui jumlah elemennya di awal.