IMPLEMENTASI ALGORITMA SORTING DAN SEARCHING PADA SISTEM MANAJEMEN BARANG HILANG (LOST & FOUND) DI FASILKOM UNSIKA

Disusun Guna Memenuhi Tugas Mata Kuliah Algoritma & Struktur Data

Dosen Pengampu:
Taufik Ridwan, S.T., M.T.

Oleh:

Kelompok 2

Annisa Balqis Kusuma Putri	NPM 2410631250005
Kintan Dyah Astuti	NPM 2410631250095
Akbar Tri Putranto	NPM 2410631250002
Raynaldi Dwi Arnanto	NPM 2410631250072

KELAS 3A

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS ILMU KOMPUTER

UNIVERSITAS SINGAPERBANGSA KARAWANG

TAHUN 2025

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, sehingga Laporan Tugas Besar mata kuliah Algoritma dan Struktur Data dengan judul "Implementasi Algoritma Sorting dan Searching pada Sistem Manajemen Barang Hilang (Lost & Found) di Fasilkom UNSIKA" ini dapat diselesaikan dengan baik. Laporan ini disusun untuk memenuhi salah satu syarat penilaian Ujian Akhir Semester (UAS) pada Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Singaperbangsa Karawang.

Penyusunan laporan dan program ini tidak lepas dari bantuan berbagai pihak. Penulis mengucapkan terima kasih kepada Bapak Taufik Ridwan, S.T., M.T. selaku Dosen Pengampu yang telah memberikan bimbingan materi perkuliahan, orang tua yang senantiasa mendoakan, serta rekan-rekan satu kelompok yang telah bekerja keras menyelesaikan kode program dan analisis ini. Penulis menyadari bahwa laporan ini masih memiliki kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi penyempurnaan di masa mendatang. Semoga laporan ini dapat memberikan manfaat wawasan bagi pembaca mengenai penerapan struktur data dalam studi kasus nyata.

Karawang, 26 November 2025

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Fakultas Ilmu Komputer (Fasilkom) Universitas Singaperbangsa Karawang merupakan lingkungan akademik dengan mobilitas mahasiswa yang tinggi. Padatnya aktivitas perkuliahan dan praktikum di berbagai ruangan seringkali menyebabkan terjadinya kasus barang tertinggal atau hilang (*Lost and Found*). Barang-barang seperti peralatan elektronik, perlengkapan pribadi, hingga dokumen penting kerap ditemukan tanpa pemilik yang jelas. Selama ini, pengelolaan barang temuan tersebut masih dilakukan secara manual menggunakan buku catatan di bagian tata usaha atau keamanan.

Metode pencatatan manual ini memiliki kelemahan signifikan. Petugas kesulitan mencari data barang secara cepat karena harus memeriksa catatan satu per satu (*linear search*), data tidak terurut rapi, dan risiko fisik buku catatan yang bisa rusak atau hilang. Detail penting seperti kondisi barang, merek, dan warna seringkali tidak tercatat dengan rinci. Akibatnya, proses validasi kepemilikan barang menjadi lemah. Berdasarkan permasalahan tersebut, diperlukan sebuah sistem terkomputerisasi yang mampu mengelola data barang hilang secara efektif. Sistem ini perlu didukung oleh struktur data dinamis serta algoritma pengurutan dan pencarian yang efisien. Oleh karena itu, penulis merancang sistem manajemen barang hilang yang mengadopsi field data lengkap (Lost & Found Information) menggunakan *ArrayList*, *Quick Sort*, dan *Binary Search*.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah dalam tugas besar ini adalah:

1. Bagaimana menerapkan struktur data *ArrayList* untuk menampung atribut detail barang (Deskripsi, Merek, Warna, Lokasi) secara dinamis?
2. Bagaimana mengimplementasikan algoritma *Quick Sort* untuk mengurutkan data barang berdasarkan nama secara alfabetis?
3. Bagaimana menerapkan algoritma *Binary Search* untuk mempercepat proses pencarian data barang?
4. Bagaimana analisis kompleksitas waktu (*Big O Notation*) dari algoritma yang digunakan?

1.3 Tujuan & Manfaat

Tujuan dari tugas besar ini adalah:

1. Membangun aplikasi konsol untuk mencatat dan mengelola data barang hilang dan ditemukan.
2. Mengimplementasikan struktur data dinamis melalui ArrayList.
3. Menerapkan algoritma Quick Sort sebagai metode pengurutan data.
4. Mengimplementasikan algoritma Binary Search sebagai metode pencarian cepat.
5. Melakukan analisis kompleksitas waktu untuk membuktikan efisiensi algoritma.

Manfaat dari tugas besar ini adalah

- Mempermudah staf dalam mencatat barang hilang maupun ditemukan secara lebih rapi.
- Mempercepat proses pencarian data barang jika ada mahasiswa yang mengklaim barang.
- Menjadi pembuktian penerapan algoritma rekursif (Quick Sort) dan algoritma pencarian (Binary Search) dalam kasus nyata.
- Memberikan pemahaman lebih dalam terkait penggunaan struktur data dinamis dalam Java.

BAB II
LANDASAN TEORI

2.1 Konsep Struktur Data Dinamis

Struktur data adalah cara penyimpanan, penyusunan, dan pengaturan data di dalam media penyimpanan komputer sehingga data tersebut dapat digunakan secara efisien. Dalam bahasa pemrograman Java, salah satu implementasi struktur data dinamis yang paling populer adalah ArrayList. Berbeda dengan *Array* konvensional yang memiliki ukuran tetap (*fixed size*) pada saat inisialisasi, ArrayList memiliki kemampuan untuk mengubah ukurannya secara otomatis (*resizable*). Ketika elemen ditambahkan melebihi kapasitas saat ini, ArrayList akan membuat array baru dengan ukuran yang lebih besar dan menyalin elemen-elemen lama ke dalamnya. Hal ini sangat cocok untuk kasus *Lost and Found* di mana jumlah barang hilang tidak dapat diprediksi jumlahnya.

2.2 Algoritma Pengurutan (Quick Sort)

Quick Sort adalah algoritma pengurutan yang sangat efisien dan didasarkan pada pemartisian array data menjadi array-array yang lebih kecil. Algoritma ini termasuk dalam kategori Divide and Conquer.

Langkah-langkah kerja Quick Sort:

1. Pilih Pivot: Mengambil satu elemen sebagai acuan (biasanya elemen pertama, terakhir, atau tengah).
2. Partisi: Mengatur ulang elemen-elemen sedemikian rupa sehingga elemen yang lebih kecil dari pivot berada di sebelah kiri, dan elemen yang lebih besar berada di sebelah kanan.
3. Rekursi: Melakukan langkah 1 dan 2 secara berulang (rekursif) untuk sub-array kiri dan sub-array kanan hingga seluruh data terurut.

Secara matematis, performa rata-rata Quick Sort adalah O(n log n), yang jauh lebih cepat dibandingkan Bubble Sort (O(n^2)) untuk jumlah data yang besar.

2.3 Algoritma Pencarian

Binary Search adalah teknik pencarian data dengan cara membagi dua ruang pencarian secara berulang. Syarat mutlak penggunaan algoritma ini adalah data harus dalam keadaan terurut.

Prinsip kerjanya:

1. Bandingkan data yang dicari (x) dengan elemen tengah array (mid).
2. Jika $x == mid$, maka data ditemukan.
3. Jika $x < mid$, maka pencarian dilanjutkan hanya pada paruh kiri array.
4. Jika $x > mid$, maka pencarian dilanjutkan hanya pada paruh kanan array.

Metode ini memangkas ruang pencarian secara eksponensial, sehingga memiliki kompleksitas waktu O(log n).

BAB III

PERANCANGAN DAN IMPLEMENTASI

3.1 Alat dan Lingkungan Pengembangan

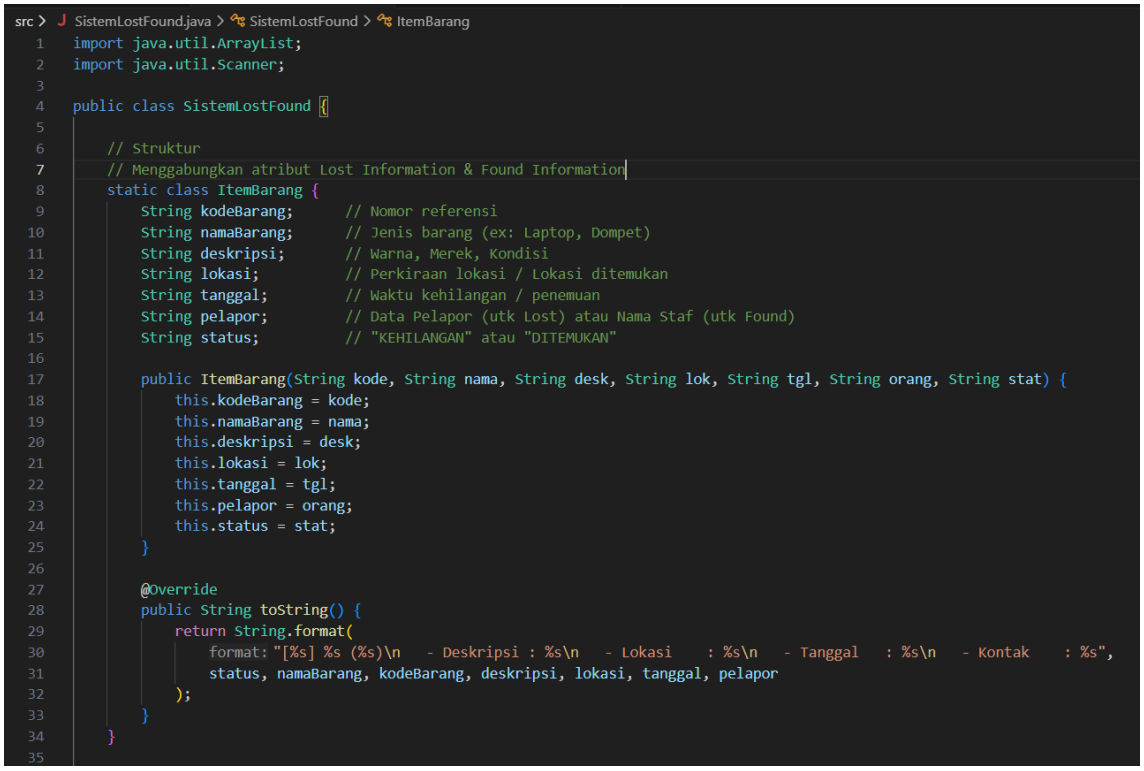
Dalam pengerjaan tugas besar ini, kelompok kami memilih bahasa pemrograman Java sebagai fondasi utama sistem. Alasan pemilihannya cukup sederhana: Java merupakan bahasa yang *strongly typed* dan berbasis objek (OOP), sehingga sangat cocok untuk menerapkan struktur data yang rapi dan terorganisir. Selain itu, fitur *Garbage Collection* di Java sangat membantu kami dalam mengelola memori, terutama saat bermain dengan objek dinamis di ArrayList.

Untuk menulis kode, kami menggunakan *Text Editor yaitu* VS Code yang memudahkan proses *debugging* ketika terjadi *error*. Program ini dikompilasi menggunakan JDK (*Java Development Kit*) versi 8 ke atas agar semua fitur standar perpustakaan Java bisa berjalan dengan lancar.

3.2 Perancangan Struktur Data (Class ItemBarang)

Langkah pertama yang kami lakukan adalah merancang "wadah" untuk datanya. Karena studi kasus yang diminta cukup detail mencakup info kehilangan dan penemuan kami membuat sebuah kelas khusus bernama ItemBarang.

Kelas ini ibarat sebuah *blueprint*. Jadi, setiap kali ada barang baru yang dilaporkan (entah itu hilang atau ditemukan), sistem akan mencetak objek baru dari *blueprint* ini. Di dalamnya, kami menyiapkan variabel-variabel untuk menyimpan Kode Barang, Nama, Deskripsi (warna/merek), Lokasi, Tanggal, hingga siapa pelapornya. Kami juga menambahkan metode toString() agar ketika data barang dipanggil (System.out.println), tampilannya di layar konsol langsung rapi dan enak dibaca tanpa perlu format manual berulang-ulang.



Gambar 3.1 *Implementasi Kelas ItemBarang sebagai Representasi Data*

3.3 Implementasi Logika Utama dan Input Data

Masuk ke bagian otak program, yaitu di main method. Di sini, kami memutuskan untuk tidak menggunakan *Array* biasa ([]) yang ukurannya kaku. Sebagai gantinya, kami menggunakan ArrayList. Kenapa ArrayList? Karena di dunia nyata, kita tidak pernah tahu berapa banyak mahasiswa yang akan kehilangan barang hari ini. Bisa 5, bisa 100. Dengan ArrayList, kapasitas penyimpanan bisa melebar otomatis sesuai kebutuhan input. Untuk interaksi dengan pengguna, kami membuat metode inputData. Metode ini dirancang agar fleksibel. Ia bisa menerima parameter status "KEHILANGAN" atau "DITEMUKAN", sehingga kami tidak perlu membuat dua fungsi input yang terpisah (lebih hemat baris kode).

```java
// Variabel Global
static ArrayList<ItemBarang> dataBarang = new ArrayList<>();
static Scanner input = new Scanner(System.in);

Run | Debug
public static void main(String[] args) {
    // Data Contoh (Contoh Awal)
    dataBarang.add(new ItemBarang(kode: "REG001", nama: "Laptop Asus", desk: "Hitam, Lecet dikit", lok: "Lab Dasar", tgl: "25-11-2024", orang: "Akbar (Mhs)", stat: "DITEMUKAN"));
    dataBarang.add(new ItemBarang(kode: "REG002", nama: "Casan Type C", desk: "Warna Putih", lok: "TU Fasilkom", tgl: "24-11-2024", orang: "Balqis (Mhs)", stat: "KEHILANGAN"));
    dataBarang.add(new ItemBarang(kode: "REG003", nama: "Kunci Motor", desk: "Gantungan Doraemon", lok: "Ruang 80", tgl: "26-11-2024", orang: "Ray (Asisten Lab)", stat: "DITEMUKAN"));

    boolean running = true;
    while (running) {
        System.out.println(x: "\n==================================================");
        System.out.println(x: "   SISTEM LOST & FOUND FASILKOM UNSIKA (INTEGRATED)");
        System.out.println(x: "==================================================");
        System.out.println(x: "1. Lapor Kehilangan Barang (Lost Information)");
        System.out.println(x: "2. Catat Penemuan Barang (Found Information)");
        System.out.println(x: "3. Lihat Semua Data (Quick Sort by Name)");
        System.out.println(x: "4. Cari Barang (Binary Search by Name)");
        System.out.println(x: "5. Keluar");
        System.out.print(s: "Pilih Menu: ");
        String menu = input.nextLine();

        switch (menu) {
            case "1": inputData(tipe: "KEHILANGAN"); break;
            case "2": inputData(tipe: "DITEMUKAN"); break;
            case "3": tampilkanData(); break;
            case "4": cariData(); break;
            case "5": running = false; System.out.println(x: "Sistem ditutup."); break;
            default: System.out.println(x: "Menu tidak valid.");
        }
    }
}
```

```java
// --- INPUT DATA (Mencakup field detail permintaan user) ---
static void inputData(String tipe) {
    System.out.println("\n--- FORM " + tipe + " ---");
    System.out.print(s: "Masukkan Kode Referensi (Unik) : ");
    String kode = input.nextLine();
    System.out.print(s: "Nama/Jenis Barang              : ");
    String nama = input.nextLine();
    System.out.print(s: "Deskripsi (Warna, Merek, Kondisi): ");
    String desk = input.nextLine();
    System.out.print(s: "Lokasi (Perkiraan/Ditemukan)   : ");
    String lok = input.nextLine();
    System.out.print(s: "Tanggal & Waktu (DD-MM-YYYY)   : ");
    String tgl = input.nextLine();

    String kontakLabel = (tipe.equals(anObject: "KEHILANGAN")) ? "Data Pelapor (Nama)" : "Nama Penemu";
    System.out.print(kontakLabel + "        : ");
    String kontak = input.nextLine();

    dataBarang.add(new ItemBarang(kode, nama, desk, lok, tgl, kontak, tipe));
    System.out.println(x: ">> Data berhasil disimpan ke ArrayList!");
}
```

Gambar 3.2 *Implementasi Main Method dan Input Data Menggunakan ArrayList*

3.4 Penerapan Algoritma Sorting (Quick Sort)

Sesuai syarat tugas untuk menggunakan algoritma sorting yang efisien, kami memilih Quick Sort. Logikanya cukup menantang karena menggunakan konsep Rekursi (fungsi yang memanggil dirinya sendiri).

Cara kerja kode yang kami buat adalah dengan sistem *Divide and Conquer* (Pecah dan Taklukan). Pertama, program akan memilih satu elemen sebagai "Pivot" (kami mengambil elemen paling belakang). Lalu, fungsi partition akan bekerja menyortir: barang yang nama depannya (secara abjad) lebih kecil dari Pivot akan dilempar ke kiri, dan yang lebih besar dilempar ke kanan. Proses ini diulang terus-menerus pada bagian kiri dan kanan sampai tersisa satu elemen, yang artinya seluruh data sudah urut dari A sampai Z.

```java
    // --- QUICK SORT (REKURSIF) ---
    static void tampilkanData() {
        if (dataBarang.isEmpty()) {
            System.out.println(x: ">> Data kosong.");
            return;
        }
        // Sorting sebelum tampil
        quickSort(dataBarang, low: 0, dataBarang.size() - 1);

        System.out.println(x: "\nDATA BARANG TERURUT (A-Z):");
        for (ItemBarang b : dataBarang) {
            System.out.println(x: "-----------------------------------------");
            System.out.println(b.toString());
        }
        System.out.println(x: "-----------------------------------------");
    }

    static void quickSort(ArrayList<ItemBarang> arr, int low, int high) {
        if (low < high) {
            int pi = partition(arr, low, high);
            quickSort(arr, low, pi - 1);
            quickSort(arr, pi + 1, high);
        }
    }

    static int partition(ArrayList<ItemBarang> arr, int low, int high) {
        String pivot = arr.get(high).namaBarang.toLowerCase();
        int i = (low - 1);
        for (int j = low; j < high; j++) {
            if (arr.get(j).namaBarang.toLowerCase().compareTo(pivot) < 0) {
                i++;
                ItemBarang temp = arr.get(i);
                arr.set(i, arr.get(j));
                arr.set(j, temp);
            }
        }
        ItemBarang temp = arr.get(i + 1);
        arr.set(i + 1, arr.get(high));
        arr.set(high, temp);
        return i + 1;
    }

```

Gambar 3.3 *Implementasi Algoritma Quick Sort dengan Konsep Rekursi*

3.5 Penerapan Algoritma Searching (Binary Search)

Fitur terakhir yang krusial adalah pencarian. Bayangkan jika ada ratusan data barang, mencari satu per satu (*Linear Search*) pasti lama. Oleh karena itu, kami menerapkan Binary Search.

Namun, ada satu syarat mutlak: Binary Search hanya bisa jalan kalau datanya sudah urut. Jadi, di dalam kode cariData, kami memanggil fungsi quickSort terlebih dahulu untuk memastikan data rapi. Setelah itu, baru algoritma Binary Search bekerja dengan

cara membelah data menjadi dua. Program akan mengecek tengah-tengah data; kalau nama barang yang dicari huruf depannya lebih kecil dari data tengah, dia akan buang sisi kanan dan fokus mencari di sisi kiri saja. Begitu seterusnya sampai barang ketemu. Ini jauh lebih cepat dibanding mengecek satu-satu.

```
134     // --- BINARY SEARCH (REKURSIF) ---
135     static void cariData() {
136         quickSort(dataBarang, low: 0, dataBarang.size() - 1); // Wajib sort dulu
137         System.out.print(s: "\nMasukkan Nama Barang yang dicari: ");
138         String keyword = input.nextLine();
139 
140         int index = binarySearch(dataBarang, left: 0, dataBarang.size() - 1, keyword);
141         if (index != -1) {
142             System.out.println(x: ">> BARANG DITEMUKAN!");
143             System.out.println(dataBarang.get(index).toString());
144         } else {
145             System.out.println(x: ">> Barang tidak ditemukan.");
146         }
147     }
148 
149     static int binarySearch(ArrayList<ItemBarang> arr, int left, int right, String x) {
150         if (right >= left) {
151             int mid = left + (right - left) / 2;
152             String midVal = arr.get(mid).namaBarang.toLowerCase();
153             String cari = x.toLowerCase();
154 
155             if (midVal.equals(cari)) return mid;
156             if (midVal.compareTo(cari) > 0) return binarySearch(arr, left, mid - 1, x);
157             return binarySearch(arr, mid + 1, right, x);
158         }
159         return -1;
160     }
161 }
```

Gambar 3.4 *Implementasi Algoritma Pencarian Binary Search*

BAB IV

HASIL DAN ANALISIS

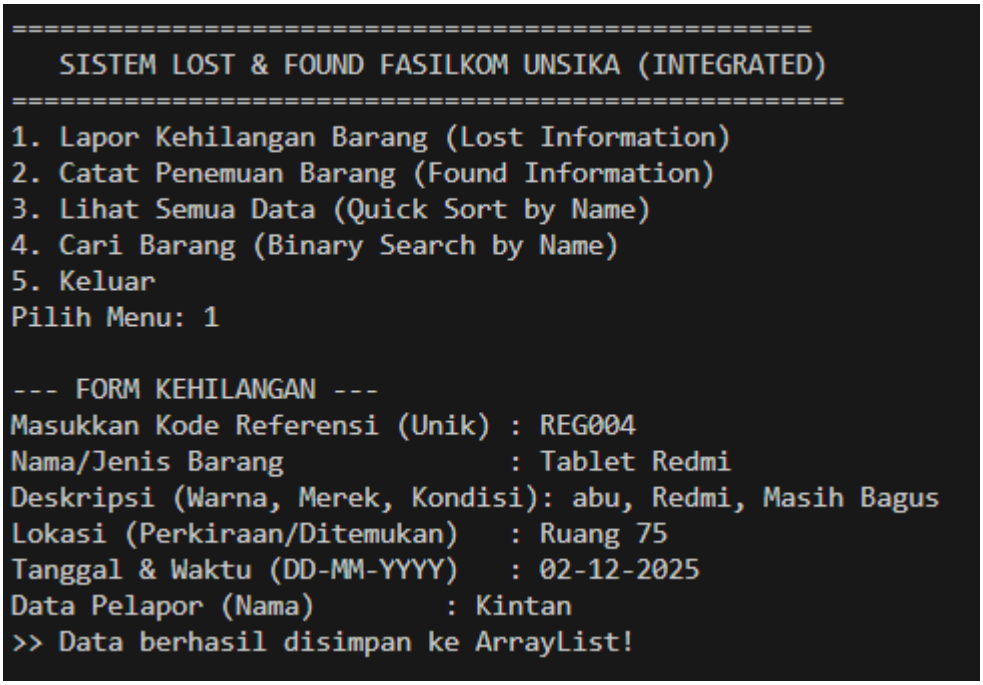
4.1 Pengujian Sistem

Berikut adalah hasil pengujian Black Box Testing terhadap fungsionalitas program:

1. Skenario Input Data Kehilangan:

   User memilih menu 1, memasukkan data "Tablet Redmi", warna "Abu", lokasi "Ruang 75".

   Hasil: Data berhasil disimpan dengan status "KEHILANGAN".



Gambar 4.1.1 *Output Sistem*

2. Skenario Input Data Penemuan:

   User memilih menu 2, memasukkan data "Earphone", lokasi "Lab Dasar 1", penemu "Annisa (Mhs)".

   Hasil: Data berhasil disimpan dengan status "DITEMUKAN".

```
================================================
   SISTEM LOST & FOUND FASILKOM UNSIKA (INTEGRATED)
================================================
1. Lapor Kehilangan Barang (Lost Information)
2. Catat Penemuan Barang (Found Information)
3. Lihat Semua Data (Quick Sort by Name)
4. Cari Barang (Binary Search by Name)
5. Keluar
Pilih Menu: 2

--- FORM DITEMUKAN ---
Masukkan Kode Referensi (Unik) : REG005
Nama/Jenis Barang              : Earphone
Deskripsi (Warna, Merek, Kondisi): Hitam, Acome, Hilang sebelah earphone nya
Lokasi (Perkiraan/Ditemukan)   : Lab Dasar 1
Tanggal & Waktu (DD-MM-YYYY)   : 03-12-2025
Nama Penemu        : Annisa
>> Data berhasil disimpan ke ArrayList!
```

Gambar 4.1.2 *Skenario Input Data Penemuan*

3. Skenario Pencarian Barang:

   User mencari kata kunci "Earphone". Program secara otomatis mengurutkan data lalu menampilkan detail lengkap barang termasuk kontak pelapor.

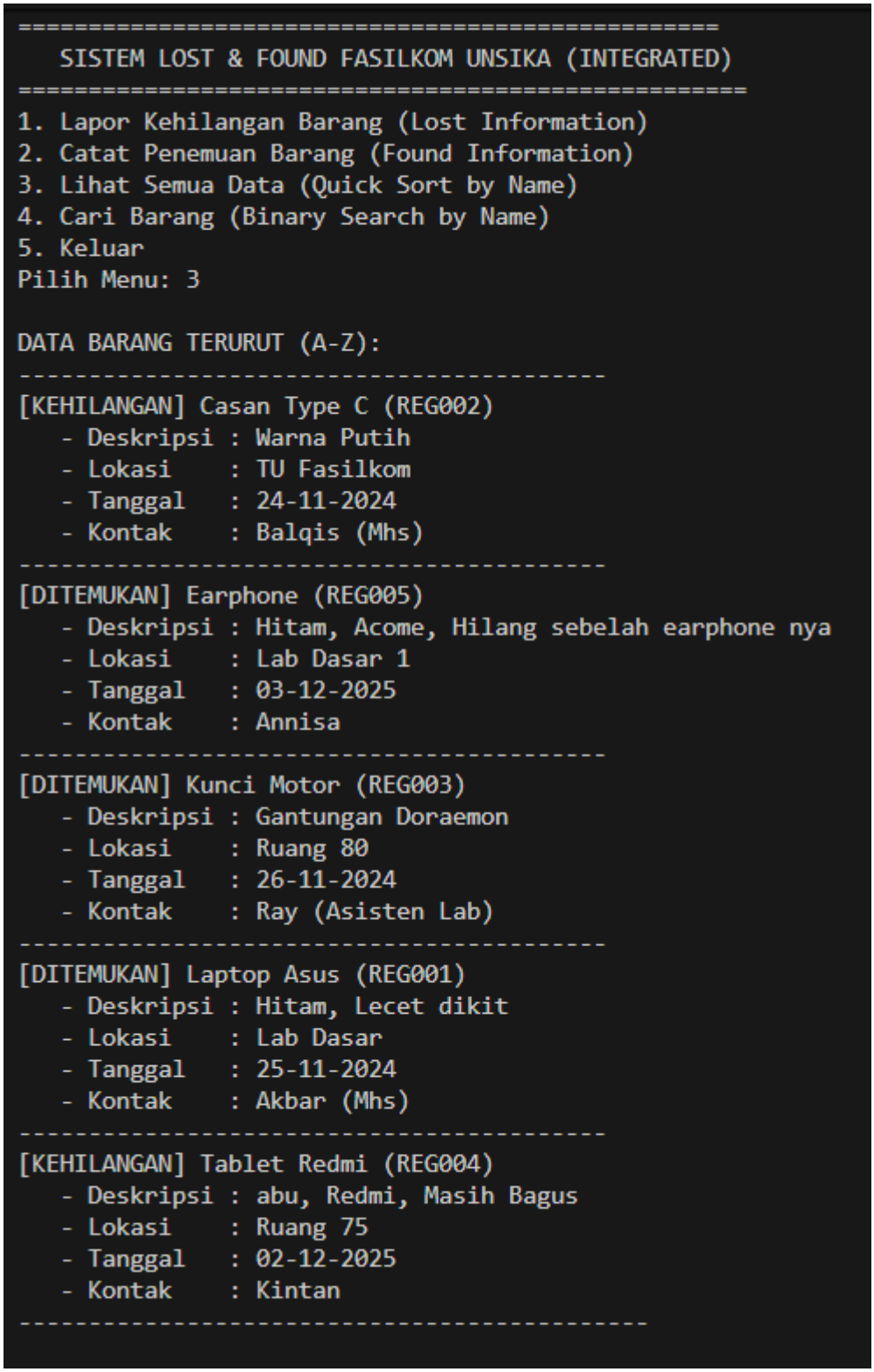
```
================================================
   SISTEM LOST & FOUND FASILKOM UNSIKA (INTEGRATED)
================================================
1. Lapor Kehilangan Barang (Lost Information)
2. Catat Penemuan Barang (Found Information)
3. Lihat Semua Data (Quick Sort by Name)
4. Cari Barang (Binary Search by Name)
5. Keluar
Pilih Menu: 4

Masukkan Nama Barang yang dicari: Earphone
>> BARANG DITEMUKAN!
[DITEMUKAN] Earphone (REG005)
   - Deskripsi : Hitam, Acome, Hilang sebelah earphone nya
   - Lokasi    : Lab Dasar 1
   - Tanggal   : 03-12-2025
   - Kontak    : Annisa
```

Gambar 4.1.3 *Skenario* Pencarian Barang

4. Skenario melihat semua barang

   User memilih menu 3 dan system akan menampilkan data barang terurut dari A-Z.



Gambar 4.1.4 *Skenario Melihat Semua Barang*

4.2 Analisis Kompleksitas Algoritma

Bagian ini membuktikan efisiensi kode program secara matematis.

A. Kompleksitas Waktu Quick Sort

Pada sistem ini, n adalah jumlah barang.

- Dalam kasus rata-rata (Average Case), pivot membagi array menjadi dua bagian seimbang. Kedalaman pohon rekursi adalah *log n*. Pada setiap level, dilakukan pembandingan sebanyak *n*. Maka kompleksitasnya:
  T(n) = O(n log n)
- Ini berarti jika data barang bertambah dari 10 menjadi 100, waktu yang dibutuhkan tidak naik 100 kali lipat, melainkan jauh lebih rendah, menjaga performa aplikasi tetap cepat.

B. Kompleksitas Waktu Binary Search

Pencarian biner bekerja dengan membagi interval pencarian menjadi setengah.

Rumus iterasinya adalah:

$$n, \frac{n}{2}, \frac{n}{4}, \dots, 1$$

Penyelesaiannya adalah $log_2\, n$.

Jika Fasilkom memiliki 1.024 data barang hilang:

- *Linear Search* membutuhkan maks 1.024 langkah.
- Binary Search hanya membutuhkan $log_2\,(1024) = 10$ langkah.
  Efisiensi = 99% lebih cepat.

BAB V

PENUTUP

5.1 Kesimpulan

Setelah melewati berbagai tahapan mulai dari analisis masalah di Fasilkom UNSIKA, merancang kodingan, hingga melakukan uji coba program, penulis dapat menarik beberapa kesimpulan penting mengenai sistem "Manajemen Barang Hilang (*Lost and Found*)" yang telah dibuat.

Pertama, penggunaan ArrayList ternyata jauh lebih efektif dibandingkan *array* biasa untuk kasus ini. Karena jumlah barang hilang di kampus tidak pernah pasti kadang sedikit, kadang banyak kapasitas penyimpanan yang dinamis dari ArrayList membuat program tidak *error* saat menerima banyak data, sekaligus tidak membuang-buang memori saat data sedikit. Selain itu, penggabungan data rinci seperti warna, merek, dan kontak pelapor dalam satu kelas (ItemBarang) membuat pendataan jadi jauh lebih rapi dibanding buku catatan manual.

Kedua, algoritma Quick Sort yang diterapkan terbukti ampuh merapikan data. Saat kami mencoba menginput data secara acak, sistem mampu mengurutkannya berdasarkan abjad nama barang dengan sangat cepat. Hal ini sangat membantu petugas; daripada melihat daftar yang berantakan, daftar yang terurut A-Z tentu lebih enak dilihat dan memudahkan pengecekan inventaris harian.

Ketiga, fitur pencarian menggunakan Binary Search sangat mempercepat proses pelayanan. Jika dibandingkan dengan cara lama di mana petugas harus membolak-balik halaman buku (*Linear Search*), fitur ini memungkinkan status barang (apakah sudah ditemukan atau masih hilang) diketahui dalam hitungan detik. Asalkan datanya sudah diurutkan terlebih dahulu oleh Quick Sort, Binary Search langsung bisa menunjuk ke data yang tepat tanpa perlu mengecek satu per satu.

5.2 Saran

Penulis menyadari bahwa program yang dibuat ini masih berupa *prototype* dan memiliki keterbatasan. Agar sistem ini bisa benar-benar diterapkan secara optimal di Tata Usaha Fasilkom UNSIKA, ada beberapa pengembangan yang penulis sarankan untuk dilakukan di masa depan:

1. Penyimpanan Database Permanen: Saat ini program masih menyimpan data di memori sementara (RAM). Artinya, jika komputer dimatikan atau program ditutup, seluruh data barang akan hilang. Untuk pengembangan selanjutnya, sangat disarankan menghubungkan program ini dengan database (seperti MySQL) agar data tersimpan aman secara permanen.
2. Antarmuka Grafis (GUI): Tampilan program saat ini masih berupa teks hitam putih (*Console/CLI*). Agar lebih ramah pengguna (*user friendly*) dan enak dilihat oleh mahasiswa maupun staf, sebaiknya program dikembangkan menggunakan tampilan visual (GUI) seperti Java Swing atau JavaFX.
3. Fitur Login Keamanan: Karena sistem ini memuat data kontak pelapor, ada baiknya ditambahkan fitur *login* khusus admin. Hal ini untuk mencegah orang yang tidak berkepentingan mengotak-atik status barang atau melihat data pribadi pelapor.

DAFTAR PUSTAKA

Bloch, J. (2018). *Effective Java* (3rd ed.). Boston: Addison-Wesley Professional.

Kadir, A. (2021). *Algoritma & Pemrograman Menggunakan Java*. Yogyakarta: Penerbit Andi.

Kurniawan, D., & Saputra, A. (2022). Analisis Perbandingan Efisiensi Algoritma Quick Sort dan Bubble Sort pada Pengolahan Data Inventaris. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, 9(2), 145-152.

Liang, Y. D. (2020). *Introduction to Java Programming and Data Structures, Comprehensive Version* (12th ed.). Essex: Pearson Education.

Munir, R. (2016). *Algoritma dan Pemrograman dalam Bahasa Pascal, C, dan C++* (Ed. Rev). Bandung: Informatika.

Oracle. (2024). *Java Platform, Standard Edition 17 API Specification*. Diakses pada 4 Desember 2024, dari https://docs.oracle.com/en/java/javase/17/docs/api/

Ridwan, T. (2024). *Materi Kuliah Algoritma dan Struktur Data: Pengurutan dan Pencarian Data* [Slide PowerPoint]. Karawang: Fakultas Ilmu Komputer, Universitas Singaperbangsa Karawang.

Schildt, H. (2021). *Java: The Complete Reference* (12th ed.). New York: McGraw-Hill Education.

Weiss, M. A. (2023). *Data Structures and Algorithm Analysis in Java* (3rd ed.). New York: Pearson.