By Neeti Ishan Bhatt (Email id: neeti.i.bhatt@gmail.com)

Design Specifications

Problem Statement:

This Java-based deduplication tool is designed to intelligently process a variable number of JSON lead records—whether clean or messy—identify duplicates by _id or email, and output a conflict-free, up-to-date dataset with a detailed change log.
The deduplication follows a defined set of rules:

- Duplicates are identified using the _id and/or email fields (with case-insensitive comparison for email).
- When duplicates are found, the record with the most recent entryDate is retained.
- If entryDate values are identical, the record that appears last in the input list is preferred.
- All changes made during the reconciliation process are recorded in a human-readable change_log.txt, showing:
  - The source record (replaced record),
  - The output record (retained final record),
  - And a field-level diff (e.g., firstName: John → Jane).

Input:

The tool accepts a single JSON file input containing an array of Lead entries, specified as a command line argument when running the program.

There are two examples input files:

- **leads.json** *(src/main/resources/data/leads.json)*— Contains well-formed leads with no missing or null required fields.
- **leadsModified.json** *(src/main/resources/data/leadsModified.json***)** — Contains leads with some missing, null, or empty fields, including invalid leads.
  Each lead contains:
  - _id (required)
  - email (required)
  - firstName (optional)
  - lastName (optional)
  - address (optional)
  - entryDate (optional, ISO 8601 string or empty)

Important:
Leads missing or having null/empty _id or email are considered **bad entries** and will be written separately to BadJson.json.

Output:

- **dedupedleads.json** *(src/main/resources/data/dedupedLeads.json)*: The list of valid, deduplicated leads. It has JSON array of deduplicated, valid leads (after merging duplicates).
- **changeLog.txt** *(src/main/resources/data/changeLog.txt):* Field-level differences for merged leads. It has Human-readable text describing all field-level changes due to deduplication.
- **BadJson.json(***src/main/resources/data/BadJson.json):* Leads missing required fields (_id or email). It has Contains leads with some missing, null, or empty fields, including invalid leads.

By Neeti Ishan Bhatt (Email id: neeti.i.bhatt@gmail.com)

Assumptions:

leads.json :

- All records are identically structured and structurally valid JSON objects.
- Each record contains **all six expected fields**: _id, email, firstName, lastName, address, entryDate
- No fields are missing or null.
- entryDate is either a valid ISO 8601 string (e.g., "2023-11-15T11:21:18Z") or an empty string.
- Email comparison is **case-insensitive**.
- BadJson.json will be **empty** for this input because all records are valid.
- Deduplication is tested purely based on _id and/or email collisions.
- The file serves as the ideal input for testing core deduplication functionality without edge case interference.
- changeLog.txt must track:
- Field-level differences between retained and discarded leads.
- No memory constraint for JSON input data.

leadsModified.json:

- Records are **well-formed JSON**, but some may have:
  - **Missing fields** (e.g., no entryDate, email, or _id etc.)
  - **Null or empty field values** (e.g., "_id": "", "email": null)
- _id and email are mandatory:
  - If either _id or email is missing or empty (""), the entry is considered bad entry and written to BadJson.json.
- Other fields (firstName, lastName, address, entryDate) can be:
  - Missing or null → treated as "Unknown" during comparison for changeLog.txt
- For valid entries:
  - firstName, lastName, and address may be present but contain "" (empty string).
  - entryDate may be null, missing, or empty (""). In such cases:
    - ✓ The lead with a non-empty entryDate is preferred during deduplication.
    - ✓ If both are missing entryDate, a consistent fallback (e.g., first entry) is selected.
- Email comparison is **case-insensitive**.
- When resolving duplicates:
  - Leads with valid entryDate take precedence.
  - If all entryDate values are null/missing/empty, pick any lead consistently.
- changeLog.txt must track:
  - Field-level differences between retained and discarded leads.
  - Which fields were “Unknown” (null/missing) vs. actual values.
- The file is ideal for testing:
  - Bad entry filtering
  - Merging incomplete leads
  - Logging field-level differences
  - Transitive duplicate resolution

File Details & Functionality:

Packages

- ***(root)*** *(com.example.leaddedup): Contains the main application entry point to run the deduplication tool.*

- ***model (****com.example.leaddedup.model): Contains the core data models representing leads and change logs, encapsulating lead attributes and tracking differences during deduplication.*
- ***service (****com.example.leaddedup.service):: Implements the business logic and algorithms for lead comparison, deduplication strategies, and orchestration of the deduplication process.*
- ***util (****com.example.leaddedup.util): Provides helper utilities for JSON file reading, writing, and other I/O operations to support the main deduplication workflow.*
- ***test****(com.example.leaddedup.test): Contains unit tests to validate the deduplication logic, handle edge cases, verify change logging accuracy, and ensure robustness of the overall application.*

Classes

- ***model/Lead.java****: POJO representing the lead structure with attributes like _id, email, firstName, lastName, address, and entryDate; includes getters, setters, and JSON serialization annotations.*
- ***model/ChangeLog.java****: Captures and formats changes between source and deduplicated leads, including deduplication cause and detailed field-level differences for logging.*
- ***service/LeadComparisonStrategy.java****: Interface defining the contract for lead comparison and deduplication strategies.*
- ***service/LatestEntryWinsStrategy.java****: Implements the strategy to retain the lead record with the most recent entryDate when duplicates are found.*
- ***service/LeadDeduplicator.java****: Core service managing the deduplication process, applying strategies, handling invalid records, and producing change logs.*
- ***util/JsonReaderUtil.java****: Utility for reading JSON input files, parsing leads, and filtering malformed or incomplete entries.*
- ***util/JsonWriterUtil.java****: Utility for writing deduplicated leads, bad JSON entries, and change logs to output files in correct formats.*
- ***Main.java****: Main application class that serves as the entry point; it initializes input reading, runs deduplication, and writes all outputs.*
- ***test/LeadDeduplicatorTest.java****: Contains comprehensive 22-unit tests for validating deduplication rules, handling malformed input, verifying change log content, and testing edge scenarios to ensure correctness and stability.*

Algorithm:

1. **Read and parse input leads** from JSON file specified by the command-line argument (either leads.json or leadsModified.json).
2. **Validate leads:**
   Separate leads with missing or empty _id or email into badLeads and write them to BadJson.json.
3. **Group duplicates:**
   Use **Union-Find (Disjoint Set)** data structure based on keys:
   - Map from _id to lead group
   - Map from normalized (lowercase) email to lead group
     Leads sharing _id or email are unioned into the same set.
4. **Merge duplicates:**
   - For each duplicate group, select the lead with the **latest entryDate**.
   - If entryDate is empty or missing, prefer leads with a valid date.

    - If ties occur, pick any consistent lead.
    - Merge data accordingly.
5. **Generate change logs:**
    - Record all field-level changes between merged leads and the retained lead.
    - Log missing fields as "Unknown".
    - Indicate duplication cause: _id, email, or both.
6. **Write output files:**
    - Writes the final deduplicated leads to dedupedLeads.json.
    - Writes the detailed change_log.txt.
    - Writes invalid leads to BadJson.json.

Challenge:

Handling Transitive Duplicates

Approach 1 (Simple Map-Based Deduplication) could only detect direct duplicates using _id or email, but failed in cases where duplicates were linked transitively (e.g., Lead A ↔ B ↔ C). This resulted in partial deduplication and unresolved overlaps.

Solution:

Transitive Deduplication via Union-Find

To overcome this, I implemented Approach 2 using the Union-Find algorithm, which groups all related leads (even indirectly) into clusters. This ensures comprehensive deduplication by resolving all transitive links and selecting the most recent entry from each group—significantly improving data accuracy and consistency.

Unit Test (LeadDeduplicatorTest.java)

Deduplication Logic Tests:

- **testSimpleDeduplication** – Validates that the lead with the later entryDate is retained when emails match.
- **testDuplicateById** – Ensures deduplication by _id favors the latest entryDate, even with different emails.
- **testTransitiveDuplicatesByIdAndEmail** – Verifies transitive deduplication using both _id and email.
- **testMultipleDuplicates** – Confirms that among multiple email duplicates, the most recent entry is retained.
- **testNoDuplicates** – Ensures that leads with distinct _id and email are all preserved.
- **testCaseInsensitiveEmail** – Checks that email deduplication is case-insensitive.
- **testPreferNonEmptyEntryDate** – Ensures a lead with a valid entryDate is preferred over one without.
- **testLatestEntryDateWinsEvenIfMissingFields** – Confirms that the newer lead wins even if it lacks some field values.

Bad JSON Input Handling Tests

- **testBadLeadMissingId** – Skips leads with missing _id from the deduplication result.
- **testBadLeadMissingEmail** – Skips leads with missing email from the deduplication result.
- **testBadLeadEmptyId** – Excludes leads with empty _id from the final output.
- **testBadLeadEmptyEmail** – Excludes leads with empty email from the final output.

Change Log Accuracy and Edge Case Tests

- **testChangeLogSingleFieldChange** – Verifies that a single field change (e.g. lastName) is logged correctly.

- **testChangeLogMultipleFields** – Ensures multiple field changes are accurately captured in the change log.
- **testChangeLogMissing** – (Unfinished test) Expected to test logging of changes when newer lead fields are missing (use "Unknown").
- **testChangeLogMissingFields-** *Verifies that missing fields in one lead are logged as "Unknown" in the change log when compared to the retained lead.*
- **testChangeLogDifferentAndMissingFields-** *Ensures that when a lead has a combination of different and missing fields compared to the retained lead, all are logged correctly in the change log.*
- **testChangeLogDuplicateByIdOnly-** *Tests that the change log correctly identifies _id as the deduplication cause when only _id is shared between leads.*
- **testChangeLogDuplicateByEmailOnly-** *Tests that the change log correctly identifies email as the deduplication cause when only email is shared between leads.*
- **testChangeLogDuplicateByIdAndEmail-** *Confirms the change log records duplication cause as both _id and email when both are matched.*
- **testOutputFileHasCorrectLeadsOrder-** *Checks that the output dedupedLeads.json preserves field order as _id, email, firstName, lastName, address, entryDate.*
- **testOutputFileHasCorrectLeadsCount-** *Validates that the number of deduplicated leads in the output matches expected count after deduplication and invalid lead removal.*

Tech Stack & Libraries:

- **Java 17** - Primary language
- **Jackson 2.15.2** - For JSON parsing (ObjectMapper, JavaTimeModule)
- **JUnit 5** - Unit testing framework
- **Maven** - Build and dependency management

Project Setup, Installation & Running

Prerequisites

- JDK 17+
- Maven 3.8+

Steps

```
# Clone repo
$ git clone <repo-url>
$ cd lead-deduplication

# Run tests
$ mvn test


# Build project
$ mvn clean compile

# Run:  leads.json
mvn exec:java -Dexec.mainClass="com.example.leaddedup.Main" -Dexec.args="src/main/resources/data/leads.json"

# Run:  leadsModified.json
mvn exec:java -Dexec.mainClass="com.example.leaddedup.Main" -Dexec.args="src/main/resources/data/leadsModified.json”
```