CODE REFACTORING AND PERFORMANCE OPTIMIZATION PROJECT REPORT

Open Source Project: FundingRaised.java

Summary

This report details the changes made to the FundingRaised.java class in the com.acme.interviews package. The purpose of the refactor was to improve code clarity, reduce redundancy, and enhance maintainability without compromising performance. This report outlines the specific modifications, their justifications, and their overall impact on software quality and performance.

Original Issues Identified

1. Redundant Code: Logic for reading and filtering CSV data was repeated across methods.
2. Hardcoded Field Mappings: CSV column references were manually indexed, reducing flexibility.
3. Low Scalability: Adding a new filter or field required multiple code changes.
4. Poor Error Reporting: Custom exceptions lacked informative messages.
5. Minimal Test Output: main() method offered little feedback for test runs.

Key Changes Made

1. Centralized CSV File Reading

- Before: File reading logic was duplicated.
- After: Introduced a reusable private method: readCSV().
- Impact:
    - Reduces code duplication.

    - Improves consistency in file handling.
    - Ensures easier maintenance and cleaner logic.

2. Header-Based Row Mapping

- Before: Data mapping used hardcoded array indices.
- After: Created a HEADERS array and used mapRowToData() to convert each row.
- Impact:
    - Makes the code robust against changes in column order.
    - Enhances readability and clarity.
    - Easier to update if the schema evolves.

3. Dynamic Filtering with matchesOptions()

- Before: Each filter (e.g., company_name, city) was hardcoded with nested if blocks.
- After: Created a single method to compare rows against all key-value filter pairs.
- Impact:
    - Simplifies logic.
    - Supports combining multiple filters easily.
    - Enables future extensibility (e.g., filter by new fields).

4. Improved Exception Handling

- Before: NoSuchEntryException had no context or message.
- After: Included a clear custom message: "No matching entry found."
- Impact:
    - Easier debugging.

- Improves user feedback.

5. Better Testing via main()

- Before: Minimal test logic and poor error visibility.
- After: Enhanced with informative console output.
- Impact:
  - Useful for quick manual testing.
  - Easier to validate correct behavior.

Impact on Performance

Category	Status	Notes
Runtime Performance	Neutral	Still reads full CSV into memory. No added computational complexity.
Scalability	Improved	New filters/fields can be added with minimal code changes.
Memory Usage	Slightly Optimized	Cleaner memory use due to DRY principles. No duplicated lists.
Code Maintenance	Significantly Improved	Modular design makes updates safer and faster.

Before vs After: Code Comparison

Filtering Logic (Before):

```
if(options.containsKey("company_name")) {
   if(csvData.get(i)[1].equals(options.get("company_name"))) {
      // map and add
   }
}
```

Filtering Logic (After):

```
if (matchesOptions(row, options)) {
   result.add(mapRowToData(row));
}
```

Row Mapping (Before):

```
mapped.put("company_name", csvData.get(i)[1]);
// repeated for each field manually
```

Row Mapping (After):

```
for (int i = 0; i < HEADERS.length; i++) {
   data.put(HEADERS[i], row[i]);
}
```

Benefits Gained

- Cleaner and More Modular Code: Easier to follow and modify.
- Improved Maintainability: Centralized logic ensures fewer bugs and easier debugging.
- Scalability: Ready for future enhancements.
- Better Developer Experience: Easier testing, clearer errors.

Opportunities for Further Improvement

1. Support for Large Files: Use streaming (e.g., BufferedReader) instead of reading all rows at once.
2. Unit Testing: Add JUnit tests for each method.
3. Caching: Avoid re-reading the CSV if data doesn’t change frequently.
4. Command-Line Interface: Add CLI or REST interface for user-friendly interaction.

Conclusion

The refactor of FundingRaised.java has substantially improved the structure and clarity of the codebase. While performance remains roughly the same, the overall quality, scalability, and maintainability of the code have significantly increased. These improvements will make future updates easier and reduce the likelihood of bugs or regressions.