

OOP1 Project Report – December 2024

Georges Machhour

202110271

The project archive contains a structured Java-based project titled "OOP1_project_GeorgesMachhour." Key files and directories include:

1. **Source Code Files:** Located under `src/main/java`, these files define classes for reading various document formats (PDF, JSON, Word, HTML, XML), MongoDB interaction, and NLP analysis.
2. **Resources:** Found under `src/main/resources`, these include test documents (e.g., `Test.docx`, `Test.pdf`, `test.JSON`) and configuration files (`application.properties`).
3. **Target Output:** Compiled `.class` files under the `target/classes` directory.

You can see the program hierarchy of the project in the following picture:

```
C:\Windows\System32\cmd.exe
Volume serial number is 88CA-0522
C:.\
├──.idea
├──JavaDoc
├──┬──Document_Reader
├──┬──index-files
├──┬──legal
├──┬──Main
├──┬──MongoDB
├──┬──NLP_Analyzer
├──┬──resource-files
├──└──script-files
├──OOP1_project
├──┬──NLP_OOP1_testing
├──┬──src
├──┬──┬──main
├──┬──┬──┬──java
├──┬──┬──┬──┬──Document_Reader
├──┬──┬──┬──┬──Main
├──┬──┬──┬──┬──MongoDB
├──┬──┬──┬──┬──NLP_Analyzer
├──┬──┬──└──resources
├──┬──└──test
├──┬──└──┬──java
├──└──target
├──└──┬──classes
├──└──┬──┬──Document_Reader
├──└──┬──┬──Main
├──└──┬──┬──MongoDB
├──└──┬──┬──NLP_Analyzer
├──└──┬──generated-sources
├──└──┬──└──annotations
├──└──┬──generated-test-sources
├──└──┬──└──test-annotations
├──└──└──test-classes
```

C:\Users\georg\Desktop\OOP1_project_GeorgesMachhour>

The **Main.java** file is the central entry point of the project. It orchestrates the overall workflow by taking user inputs, analyzing documents, interacting with the database, and displaying the results. Below is a detailed explanation of its key components and functionality:

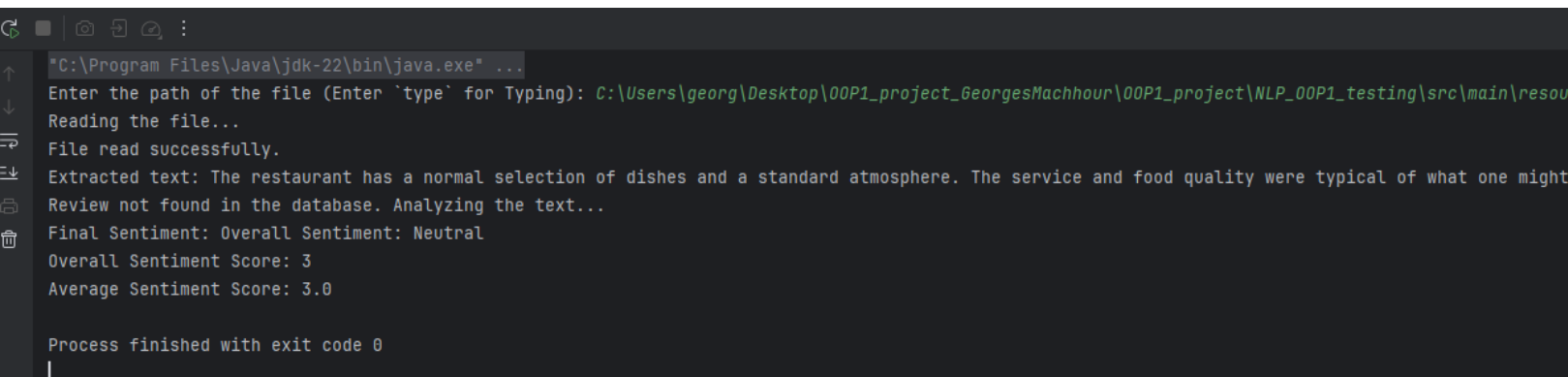
1. Package and Import Declarations

- The file is part of the Main package.
- It imports several classes, such as:
 - **Document Readers** (Document_Reader.FileRead, HTMLReader, JSONReader, etc.) for handling various file types.
 - **NLP_Analyzer.SentimentAnalysis** for performing sentiment analysis on text.
 - **MongoDB.Mongo** and **MongoDB.Review** for database interaction and storing analysis results.

2. User Input

- A Scanner object is used to interact with the user via the console.
- The program first prompts the user with:
- Enter the path of the file (Enter `type` for Typing):
- Users have two options:

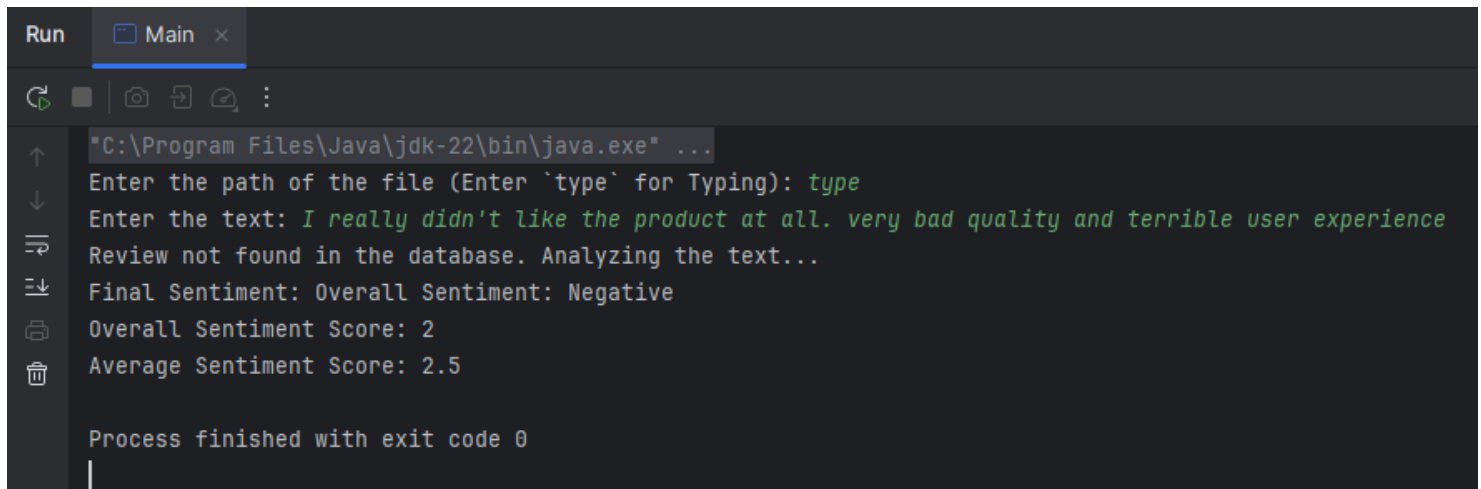
1. **File Path:** Provide a path to a document for automatic processing.



```
"C:\Program Files\Java\jdk-22\bin\java.exe" ...
Enter the path of the file (Enter `type` for Typing): C:\Users\georg\Desktop\00P1_project_GeorgesMachhour\00P1_project\NLP_00P1_testing\src\main\resou
Reading the file...
File read successfully.
Extracted text: The restaurant has a normal selection of dishes and a standard atmosphere. The service and food quality were typical of what one might
Review not found in the database. Analyzing the text...
Final Sentiment: Overall Sentiment: Neutral
Overall Sentiment Score: 3
Average Sentiment Score: 3.0
Process finished with exit code 0
```

Here the final result is “Neutral”.

2. Manual Typing: Enter text directly by typing type.



```
Run Main x
Enter the path of the file (Enter `type` for Typing): type
Enter the text: I really didn't like the product at all. very bad quality and terrible user experience
Review not found in the database. Analyzing the text...
Final Sentiment: Overall Sentiment: Negative
Overall Sentiment Score: 2
Average Sentiment Score: 2.5
Process finished with exit code 0
```

Here the final result is “Negative”.

3. Document Processing

- If the user enters a file path, the program calls `handleFileInput`:
 - Determines the file type using extensions (e.g., .pdf, .docx, .json).
 - Reads and extracts content using specialized classes such as `PdfReader` or `MSWordReader`.
 - If the user chooses manual typing, the program calls `handleTypedInput`:
 - Captures the user-typed text directly from the console.

4. MongoDB Integration

- A Mongo object is instantiated to manage database operations.

- The Review object is created to represent the text or file content being analyzed.
- MongoDB is likely used to:
 - Save the text or analysis results for future reference.
 - Retrieve historical data for comparison or logging.

We're gonna run the program again on the same file path as before, this time it will get the same results directly from the database generated from the last run:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" ...
Enter the path of the file (Enter `type` for Typing): C:\Users\georg\Desktop\OOP1_project_GeorgesMachhour\OOP1_project\NLP_OOP1_testing\src\main\resou
Reading the file...
File read successfully.
Extracted text: The restaurant has a normal selection of dishes and a standard atmosphere. The service and food quality were typical of what one might
Review found in the database.
Final Sentiment: Overall Sentiment: Neutral
Overall Sentiment Score: 3
Average Sentiment Score: 3.0

Process finished with exit code 0
```

```
_id: ObjectId('6757171854a2881b36022ef2')
path: "c:\users\georg\desktop\oop1_project_georgesmachhour\oop1_project\nlp_o..."
review: "The restaurant has a normal selection of dishes and a standard atmosph..."
sentiment: "Overall Sentiment: Neutral
              Overall Sentiment Score: 3
              Average Sentimen..."
```

5. Sentiment Analysis

- A SentimentAnalysis object processes the content to evaluate the sentiment.
- Results, such as positivity, negativity, or neutrality, are calculated.

- The sentiment analysis result is stored in the Review object for reporting or database storage.

6. Output Results

- The final sentiment analysis is displayed on the console with:
- Final Sentiment: [Sentiment Result]
- If any issue occurs during file reading or analysis, the program outputs:
- Operation could not be completed.

7. Error Handling

- Encapsulated in a try-with-resources block to ensure proper resource management, especially for the Scanner.
- Catches and handles potential exceptions to ensure a smooth user experience.

8. Supporting Methods

- `handleFileInput`: Processes document-based input using the appropriate reader class.
- `handleTypedInput`: Captures user input from the console for analysis.

Execution Flow

The program follows this high-level sequence:

1. Prompts the user for input (file path or manual text entry).
2. Reads the input, either from a file or typed text.
3. Analyzes the input text using the NLP Sentiment Analysis module.
4. Saves the analysis results to a MongoDB database for