Documentation

Reasoning behind the logic

1. Use of CloseableHttpClient:
Purpose: Manages HTTP requests efficiently and handles network communications.
Reasoning: Ensures reliable network interactions and enables robust handling of HTTP protocols, which is crucial for accessing and retrieving data from web pages securely and effectively.

2. Dynamic URL Retrieval and User Interaction:
Purpose: Allows users to select specific topics for data retrieval.
Reasoning: Enhances usability by enabling targeted data extraction, which reduces unnecessary network traffic and focuses resources on user-specified interests.

3. HTML Parsing with Jsoup:
Purpose: Extract relevant data like titles, authors, and dates from the HTML content of web pages.
Reasoning: Jsoup provides a convenient and efficient way to navigate and parse HTML documents, making it ideal for extracting structured data from potentially complex web page layouts.

4. Data Formatting and Condensation:
Purpose: Formats dates and limits the number of displayed authors.
Reasoning: Standardizes data presentation and simplifies output, making the information more accessible and easier to manage for further processing or analysis.

5. Output Generation Using PrintWriter:
Purpose: Writes formatted data to a text file.
Reasoning: Facilitates data storage and subsequent access, using a simple, delimited format that supports easy parsing and integration with other data analysis tools or databases.

6. Error Handling:
Purpose: Manages exceptions and provides feedback.
Reasoning: Enhances the application's reliability and user experience by ensuring that operational issues are handled gracefully and communicated clearly to the user.

7. Pagination Handling:
Purpose: Navigate through multiple pages of a topic to gather comprehensive data.
Reasoning: Ensures thorough data collection from website sections that spread content across several pages, crucial for complete data retrieval.

High-Level Main Application Flow

1. Initialization
HTTP Client Setup: The application uses CloseableHttpClient to manage HTTP connections. This choice is strategic and required for handling HTTP communications robustly, allowing for the management of cookies, connections, and other protocol-specific aspects.
User-Agent Configuration: Setting a custom User-agent helps prevent the crawler from being recognized as a bot, which can lead to blocking or other access restrictions.

2. Interactive Topic Selection:
Fetching and Displaying Topics: Initially, the crawler fetches the main topics list from the base URL. It then parses this HTML content to extract and display available topics. This approach allows users to interact with up-to-date information directly from the website, ensuring that the crawler adapts to any content changes on the Cochrane Library website.
User Input for Topic Selection: By incorporating user input to select topics, the crawler becomes more flexible and targeted, reducing unnecessary data processing and focusing on user-specified areas of interest.

3. Retrieve Detailed Content for Selected Topic
Dynamic URL Retrieval: After a topic is selected, the application constructs the URL for the detailed page of the chosen topic. This dynamic URL construction is crucial for navigating a website where URLs might change or where content is paginated.
Pagination Handling: The crawler detects and navigates through pagination, ensuring comprehensive data collection across multiple pages within a topic. This is vital for thorough data extraction where topics contain extensive lists of articles or studies.

4. Data Extraction and Formatting
Parse Detailed Content: Used Jsoup to parse the detailed reviews page for the specific topic. Extract relevant data such as article titles, authors, publication dates, and URLs of individual papers or studies.
Format Extracted Data: Clean and format the data for consistency. This may include trimming strings, converting date formats, and limiting the number of authors displayed. Used external package to WordWrap authors list to top 3 authors.

6. Output Generation
File Writing with PrintWriter: Extracted and formatted data is written to a text file using PrintWriter, which supports character encoding and efficient buffer management. The use of StandardOpenOption.APPEND ensures that data is appended to the existing file, preserving previous data and adding new information seamlessly.
Structured Output Format: Data is structured with delimiters for easy parsing and future processing, which can be crucial for users who might want to import this data into databases or perform further analysis.

7. Error Handling:
Comprehensive Error Management: The application handles exceptions at multiple levels, from HTTP communication errors to file writing issues, ensuring robust operation under various error conditions. Feedback is provided to the user through console messages, enhancing transparency and user experience.