Assignment Title: PDF Content Segmenter

Objective:

Develop a Java application that programmatically segments a system-generated PDF into distinct sections based on the whitespace between blocks of text, making exactly `X` cuts. The goal is to identify logical sections such as headings, paragraphs, and distinct blocks that are visually separated by increased whitespace, without using image processing techniques.

Requirements:

1. **Input Specifications**:
    - The application should accept a PDF file as input.
2. **Processing Details**:
    - Analyze the PDF to identify sudden changes in whitespaces along Y-axis that significantly separates text blocks.
    - Define "significant whitespace" as vertical spaces that are noticeably larger than the typical line spacing within paragraphs.
    - The application should make exactly `X` cuts based on the largest `Y` white spaces.
3. **Output Specifications**:
    - The output should be multiple PDF files, each containing one of the segments created by the cuts.
    - Ensure that the segmentation does not cut through the middle of text blocks or paragraphs.
4. **Constraints**:
    - Do not use image processing libraries or convert PDF pages to images for processing.
    - Leverage PDF manipulation libraries such as Apache PDFBox or iText to analyze and manipulate the PDF content.
5. **Technologies to Use**:
    - Java 8 or above
    - Apache PDFBox
6. **Deliverables**:
    - Source code with appropriate comments to enhance readability and maintainability.
    - A `README.md` file documenting:
        - Setup instructions
        - How to run the application
        - Examples of usage
    - Unit tests demonstrating the functionality and handling edge cases.

Evaluation Criteria:

- **Functionality**: The application meets all the requirements and correctly segments the PDF into the defined number of sections based on whitespace.
- **Code Quality**: The code should be clean, well-organized, and follow Java coding standards and best practices.
- **Error Handling**: The application gracefully handles errors such as invalid input or files that do not conform to expected formats.
- **Documentation**: The documentation is clear, concise, and sufficient for setting up and running the application.
- **Testing**: The presence of comprehensive tests that cover various scenarios including edge cases.