Documentation Detective:

Objective: Develop a task using Gradle that automatically scans the codebase for classes and methods annotated with @ClassDocumentaion and @MethodDocumentation.

Tasks:

1. Parse the annotations and extract the javadoc from the code with the associated descriptions.
2. Identify classes with annotations but missing documentation and vice versa.
3. Generate reports highlighting inconsistencies
4. Create a file with all the extracted documentation

Java Assignment using Gradle, Apache POI, JCharts, DBCP, and iTextPDF

Scenario:

Develop a Java application that utilizes the provided libraries to:

Read data from an Excel file:
File to be used:
https://docs.google.com/spreadsheets/d/1-E4EAizNV2ozKKMvD9ifTK83cpthKszd/edit#gid=190050555

Use Apache POI library to parse the Excel file and extract data into Java objects.

Load data records into a database:

Utilize DBCP connection to connect to the database and insert the extracted data records. (Use Parallel streams to insert the data in parallel)

Create charts:

- Figure out the Team with Maximum number of interviews for the months of October and November 2023 by querying the database
- Figure out the Team with Minimum number of interviews for the months of October and November 2023 by querying the database

- Figure out the top 3 Panels for the month of October and November 2023 using lambda streams
- Figure out the top 3 Skills for the month of October and November 2023 using a view on the database
- Figure out the top 3 Skills for which the interviews were conducted in the **Peak Time**

Use JCharts library to generate visualizations based on the loaded database data.

Attach charts in a PDF file and print the path of the PDF file generated.

Project Setup:

Create a new Gradle project and configure build dependencies for Apache POI, JCharts, DBCP, and iTextPDF.

Excel data reading:

1. Implement a class to read the Excel file using XSSFWorkbook or HSSFWorkbook depending on your Excel file format.
2. Iterate through rows and columns of the sheet(s) and extract data into Java objects representing your data model.

Database connection and data insertion:

1. Use DBCP to establish a connection pool for efficient database access.
2. Prepare SQL statements for inserting your extracted data objects into the desired database tables.
3. Execute the prepared statements within the connection pool to populate the database.
4. Use Multithreading to insert the data faster using a a connection pool

Chart generation:

1. Use JCharts to create different types of charts.
2. Customize chart properties like title, labels, colors, and legends.

PDF creation and chart embedding:

1. Use iTextPDF to create a new PDF document.
2. Define page layout and sections for document structure.
3. Utilize iTextPDF's chart embedding features to insert the generated JCharts objects into specific sections of the PDF.

Execute the application:

1. Define a main method to initiate the data reading, database insertion, chart generation, and PDF creation steps.
2. Handle any exceptions or errors during execution.