Student Priority Group Tool

Description

This is a Java application designed to identify students who fall into specific "priority groups" based on various academic and administrative criteria. It features a graphical user interface (GUI) built with Swing for ease of use, allowing users to load student data, select criteria, and generate reports. It also includes helper scripts for preparing raw Excel data.

Features

- **Graphical User Interface:** Provides an intuitive interface (`src.Main`) for selecting data folders, programmes, and priority criteria.
- **Multiple Priority Criteria:** Supports identifying students based on:
  - Low Attendance (`src.DataPipeline.isLowAttendance`)
  - New Registration Status (`src.DataPipeline.isNewRegistration`)
  - Trailing Modules (`src.DataPipeline.hasTrailingModules`)
  - Failed Components (`src.DataPipeline.hasFailedComponents`)
  - Programme Registration Status Not 'RE' (`src.DataPipeline.isProgrammeRegStatusNotRE`)
  - Module Enrollment Status Not 'RE' (`src.DataPipeline.hasModuleEnrollmentNotRE`)
  - Overdue Components (`src.DataPipeline.hasOverdueComponents`)
  - All Reasons combined (`src.DataPipeline.calculatePriorityGroup`)
- **Configurable Thresholds:** Allows setting a custom attendance percentage threshold (`src.Main.attendanceRateField`).
- **Data Loading:** Fetches student data based on selected programme code from a specified root folder (using `src.StEP.fetchStudents` - *Note: `StEP.java` details are assumed*).
- **Logging:** Generates detailed logs for data processing and priority group identification, saved to separate CSV files per priority type (`src.DataPipeline.logPriorityStudent` , `src.DataPipeline.getLogFolderPath`).
- **CSV Export:** Allows exporting the list of identified priority students to a CSV file (`src.Main.handleExportToCsv`).
- **Data Preparation Scripts:** Includes VBScripts to help preprocess raw Excel files.

Prerequisites

- Java Development Kit (JDK) 8 or later.
- Microsoft Excel installed (for running the VBScript helper scripts).
- The Apache POI libraries included in the `lib/` directory (for the Java application).

Setup

1. Ensure you have a compatible JDK installed and configured.
2. Ensure Microsoft Excel is installed.
3. Compile the Java source code. You can typically do this in an IDE or via the command line:

```
# Navigate to the project's root directory
# Adjust classpath separator (';' for Windows, ':' for Linux/macOS) if needed
javac -d bin -cp "lib/*" src/*.java
```

Data Preparation (Using Helper Scripts)

The `rawData` folder contains VBScripts to help convert and organize raw Excel reports into the CSV format expected by the Java application.

Important: These scripts require Microsoft Excel to be installed on your Windows machine.

1. **`RenameProgrammeReportWB.vbs`：**
   - **Purpose:** Renames Excel files located specifically in the `rawData/SBMT/EBR` folder based on content within the first sheet (Programme Code, Term Year, Level Year). It standardizes filenames to `ProgrammeReport-<ProgrammeCode>-<TermYear>-<LevelYear>.xlsx`.
   - **Placement:** Place this script inside the `rawData` folder. It is hardcoded to look for files within its `SBMT/EBR` subdirectory.
   - **Usage:** Navigate to the `rawData` folder and double-click the script or run it from the command line:

     ```
     cscript RenameProgrammeReportWB.vbs
     ```

   - **Note:** Review the script's output for any skipped files or errors.
2. **`ProcessExcelToCSV.vbs`：**
   - **Purpose:** Recursively scans the folder it's placed in (and its subfolders) for `.xlsx` files. It converts each worksheet into a separate CSV file, performing data cleaning (removing quotes, commas, line breaks). It handles "ProgrammeReport" files specially by renaming

subsequent sheets to "Module". Skips temporary files and sheets containing errors like "Report could not be retrieved".

- **Placement:** Place this script inside the `rawData` folder (or the specific parent folder containing the Excel files you want to process).
- **Output:** Creates a `Data` folder at the same level as the script's parent folder (e.g., if the script is in `rawData`, it creates `../Data`). The CSV files are saved within this `Data` folder, mirroring the subfolder structure of the source Excel files.
- **Usage:** Navigate to the folder containing the script and double-click it or run from the command line:

```
cscript ProcessExcelToCSV.vbs
```

- **Note:** This script will overwrite existing CSV files if run multiple times without clearing the `Data` folder first. It forcefully closes any running Excel instances upon completion.

Usage (Java Application)

1. **Prepare Data:**
   - Use the helper scripts (`RenameProgrammeReportWB.vbs`, `ProcessExcelToCSV.vbs`) to convert your raw Excel files into the required CSV format within the `Data` folder.
   - Ensure the `Data` folder contains subdirectories named after the programme codes (e.g., `Data/SBMT/`, `Data/LBL/`) containing the generated CSV files.
2. **Run the Application:**

```
# Adjust classpath separator (';' for Windows, ':' for Linux/macOS) if needed
java -cp "bin;lib/*" src.Main
```

3. **Using the GUI (`src.Main`):**
   - **Choose Root Folder:** Select the `Data` directory created by the `ProcessExcelToCSV.vbs` script.
   - **Choose Target Folder:** Select the directory where log files and exported CSVs should be saved (defaults towards `result/`).
   - **Programme Code:** Select the specific programme (e.g., `SBMT`) to process from the dropdown (`src.Main.programmeCombo`). The application expects CSV files within the corresponding subfolder in the Root Folder (e.g., `Data/SBMT/`).
   - **Priority Reason:** Select the desired criteria from the dropdown (`src.Main.reasonCombo`).
   - **Attendance Threshold (%):** If "Low Attendance" or "All Reasons" is selected, this field (`src.Main.attendanceRateField`) becomes active. Enter the threshold percentage (default is 30).

- **Load Data:** Click to load student data for the selected programme from the CSV files using `src.StEP.fetchStudents`. Status messages appear in the log area.
- **Generate Priority Group:** After loading data, click this to process students based on the selected criteria using `src.DataPipeline.fetchPriorityStudents`. Results are shown in the log area.
- **Export to CSV:** If priority students were found, click this button (`src.Main.exportBtn`) to save the list to a CSV file.
- **Reset:** Clears selections, data, and log messages.

4. **Output:**
   - Log files are generated in the specified target folder under `<Target Folder>/<ProgrammeCode>/log/` (e.g., `result/SBMT/log/`). Separate logs like `priority_student_list_LowAttendance.csv` are created.
   - Exported CSV files are saved to the location chosen via the "Save" dialog.

Project Structure

```
.
├── rawData/            # Folder for raw Excel files and helper scripts
│   ├── SBMT/
│   │   └── EBR/        # Location for files processed by RenameProgrammeReportWB.vbs
│   ├── LBL/
│   ├── ProcessExcelToCSV.vbs
│   └── RenameProgrammeReportWB.vbs
├── Data/               # Processed CSV data folder (output of ProcessExcelToCSV.vbs)
│   ├── LBL/
│   └── SBMT/
│       └── EBR/
├── lib/                # Required JAR libraries (Apache POI)
├── result/             # Default output folder for logs and exports
│   └── SBMT/
│       └── log/
├── src/                # Java source files (Main.java, DataPipeline.java, etc.)
├── bin/                # Compiled .class files (output of compilation)
├── pom.xml             # Maven Project Object Model (if used)
└── README.md           # This file
```

Dependencies

- **Java Application:**

    - Apache POI: Used for handling Microsoft Office formats (likely Excel for data input/output, based on library presence). Found in the `lib/` directory.
- **Helper Scripts:**
    - Microsoft Windows
    - Microsoft Excel

Description

This is a Java application designed to identify students who fall into specific "priority groups

Features

*   **Graphical User Interface:** Provides an intuitive interface ([`src.Main`](src/Main.java))
*   **Multiple Priority Criteria:** Supports identifying students based on:
    *   Low Attendance ([`src.DataPipeline.isLowAttendance`](src/DataPipeline.java))
    *   New Registration Status ([`src.DataPipeline.isNewRegistration`](src/DataPipeline.java))
    *   Trailing Modules ([`src.DataPipeline.hasTrailingModules`](src/DataPipeline.java))
    *   Failed Components ([`src.DataPipeline.hasFailedComponents`](src/DataPipeline.java))
    *   Programme Registration Status Not 'RE' ([`src.DataPipeline.isProgrammeRegStatusNotRE`](s
    *   Module Enrollment Status Not 'RE' ([`src.DataPipeline.hasModuleEnrollmentNotRE`](src/Dat
    *   Overdue Components ([`src.DataPipeline.hasOverdueComponents`](src/DataPipeline.java))
    *   All Reasons combined ([`src.DataPipeline.calculatePriorityGroup`](src/DataPipeline.java)
*   **Configurable Thresholds:** Allows setting a custom attendance percentage threshold ([`src.
*   **Data Loading:** Fetches student data based on selected programme code from a specified roo
*   **Logging:** Generates detailed logs for data processing and priority group identification,
*   **CSV Export:** Allows exporting the list of identified priority students to a CSV file ([`s
*   **Data Preparation Scripts:** Includes VBScripts to help preprocess raw Excel files.

Prerequisites

*   Java Development Kit (JDK) 8 or later.
*   Microsoft Excel installed (for running the VBScript helper scripts).
*   The Apache POI libraries included in the `lib/` directory (for the Java application).

Setup

1. Ensure you have a compatible JDK installed and configured.
2. Ensure Microsoft Excel is installed.
3. Compile the Java source code. You can typically do this in an IDE or via the command line:
    ```sh
    # Navigate to the project's root directory
    # Adjust classpath separator (';' for Windows, ':' for Linux/macOS) if needed
    javac -d bin -cp "lib/*" src/*.java
    ```

Data Preparation (Using Helper Scripts)

The `rawData` folder contains VBScripts to help convert and organize raw Excel reports into the

Important: These scripts require Microsoft Excel to be installed on your Windows machine.

1. **`RenameProgrammeReportWB.vbs`:**
    *   **Purpose:** Renames Excel files located specifically in the `rawData/SBMT/EBR` folder b
    *   **Placement:** Place this script inside the `rawData` folder. It is hardcoded to look fo
    *   **Usage:** Navigate to the `rawData` folder and double-click the script or run it from t
        ```cmd
        cscript RenameProgrammeReportWB.vbs
        ```
    *   **Note:** Review the script's output for any skipped files or errors.

2. **`ProcessExcelToCSV.vbs`:**
    *   **Purpose:** Recursively scans the folder it's placed in (and its subfolders) for `.xlsx
    *   **Placement:** Place this script inside the `rawData` folder (or the specific parent fol
    *   **Output:** Creates a `Data` folder at the same level as the script's parent folder (e.g
    *   **Usage:** Navigate to the folder containing the script and double-click it or run from
        ```cmd
        cscript ProcessExcelToCSV.vbs
        ```
    *   **Note:** This script will overwrite existing CSV files if run multiple times without cl

Usage (Java Application)

1. **Prepare Data:**
    *   Use the helper scripts (`RenameProgrammeReportWB.vbs`, `ProcessExcelToCSV.vbs`) to conve
    *   Ensure the `Data` folder contains subdirectories named after the programme codes (e.g.,
2. **Run the Application:**
    ```sh
    # Adjust classpath separator (';' for Windows, ':' for Linux/macOS) if needed
    java -cp "bin;lib/*" src.Main
    ```
3. **Using the GUI ([`src.Main`](src/Main.java)):**
    *   **Choose Root Folder:** Select the `Data` directory created by the `ProcessExcelToCSV.vb
    *   **Choose Target Folder:** Select the directory where log files and exported CSVs should
    *   **Programme Code:** Select the specific programme (e.g., `SBMT`) to process from the dro
    *   **Priority Reason:** Select the desired criteria from the dropdown ([`src.Main.reasonCon
    *   **Attendance Threshold (%):** If "Low Attendance" or "All Reasons" is selected, this fie
    *   **Load Data:** Click to load student data for the selected programme from the CSV files
    *   **Generate Priority Group:** After loading data, click this to process students based on
    *   **Export to CSV:** If priority students were found, click this button ([`src.Main.export
    *   **Reset:** Clears selections, data, and log messages.

4. **Output:**

```
    *   Log files are generated in the specified target folder under `<Target Folder>/<Programme
    *   Exported CSV files are saved to the location chosen via the "Save" dialog.

## Project Structure
```

.

```
├── rawData/ # Folder for raw Excel files and helper scripts
│   ├── SBMT/
│   │   └── EBR/ # Location for files processed by RenameProgrammeReportWB.vbs
│   ├── LBL/
│   ├── ProcessExcelToCSV.vbs
│   └── RenameProgrammeReportWB.vbs
├── Data/ # Processed CSV data folder (output of ProcessExcelToCSV.vbs)
│   ├── LBL/
│   └── SBMT/
│   └── EBR/
├── lib/ # Required JAR libraries (Apache POI)
├── result/ # Default output folder for logs and exports
│   └── SBMT/
│   └── log/
├── src/ # Java source files (Main.java, DataPipeline.java, etc.)
├── bin/ # Compiled .class files (output of compilation)
├── pom.xml # Maven Project Object Model (if used)
└── README.md # This file
```

```
## Dependencies

*   **Java Application:**
    *   Apache POI: Used for handling Microsoft Office formats (likely Excel for data input/outp
*   **Helper Scripts:**
    *   Microsoft Windows
    *   Microsoft Excel
```