Parser Control Flow

The transcript parser takes in a PDF of a CSE or ISE student transcript, parses the contents of the transcript, and creates a tabular document representing the mandated classes the student has taken including remainder courses. The program is capable of producing these documents through 12 modules, each of which are dependent on one another. The primary data structure that is used to store the parsed information is a dictionary. The entire control flow of the program is explained below:

1) The program begins by first parsing the name of the transcript the user provides as input. The format of such input is the following: **input\TranscriptName.pdf**

2) Once the name of the transcript has found, the main program initializes ten state dictionaries in order to populate them:
    - **studentInformation**: Contains information regarding basic student information (i.e. Cumulative GPA, Cumulative Credits, Major, etc)
    - **upperDivisionCourses:** Contains information regarding upper division courses that the student has taken
    - **lowerDivisionCourses:** Contains information regarding lower division courses that the student has taken
    - **technicalCourses:** Contains information regarding technical courses that the student has taken
    - **mathRequiredCourses:** Contains information regarding the **required** math courses that students need to take (classes will only be added if a class that the student has taken falls under the required courses)
    - **scienceCourses:** Contains information regarding the **required** science courses that students need to take (classes will only be added if a class that the student has taken falls under the required courses)
    - **specializeCoursesISE:** Contains information regarding **required specialization** courses for ISE students (classes will only be added if a class that the student has taken falls under the specialization courses)
    - **specializeCoursesCSE:** Contains information regarding **required specialization** courses for CSE students (classes will only be added if a class that the student has taken falls under the specialization courses)
    - **classesPerSemester:** Contains information regarding the classes that the student has taken per semester

3) Once these state dictionaries have been initialized, the program makes a call to a function in **StudentInfo.py** to populate some of the student information.

4) Once the function has finished, the program then calls a function in **PlanBackwards.py** to parse and store the **major** and **specialization** of the student.

5) The program then creates six threads, each of which will manage a state dictionary to populate information. One thread will finish populating the student information dictionary by calling a function in **CumulativeInfo.py** while the remaining five threads will populate the remainder by calling multiple functions in **ParseClasses.py**

6) Once the state dictionaries have all been filled, the program formulates the tabular document. This is done by calling a function in **CreateTabularDocument.py**, creating tables that are both formatted and displayed with readable information. In total, at least **three** tables will be created to display the content.

7) Within **CreateTabularDocument.py**, the module also calls three different functions in three different modules: **CreateCSESpecDocument.py, CreateISESpecDocument.py,** and **CreateSingularSpecDocument.py**. Each of these modules will create its own table within the same XLSX file in order to increase organization of the overall output

8) Once all the tables have been created with proper formatting and readable output, the program will store the newly created XLSX file into the **output** folder for it to be interpreted and read by a user.

Python Modules

Main.py

This is the program's entry, where multiple functions are called and threads are created to handle state dictionaries as their resources. Such state dictionaries are initialized in the main module.

StudentInfo.py

This is the first function that the main module calls in order to populate the student information. The main program calls the **trackStudentInformation()**, which takes in the following input:

- **Text:** Contains a list of strings that has information from the pdf
- **studentInformation**: Takes in the state dictionary to populate after parsing

This function will then add the following data to the state dictionary:

- **Last Updated:** The time when the XLSX file was last updated
- **Name:** The student's name
- **ID Number:** The student's ID number
- **Requirement Term:** When the student began their undergraduate tenure
- **CSHP:** If a student is in the Computer Science Honors Program

PlanBackwards.py

This module has a function called **studentInfoBackwards()** which takes in the following input:

- **currentPage:** The current PDF page being parsed
- **afterCP:** The page after the current page
    - ❖ Note: If the currentPage is the lastmost page in the PDF, then **afterCP** will equal to **currentPage**
- **studentInformation:** The state dictionary to populate the student information

The function is responsible for combining all the semesters that the student has taken into a list. Once all semesters have been added, the function calls **parsePlanInformation()**, which takes in the following parameters:

- **semesterList:** The list of semesters
- **currentPage:** The current PDF page that we are parsing information from
- **afterCP:** The page after the current page that we are parsing information from
    - ❖ Note: If the currentPage is the lastmost page, then the page after will **equal** to the current page
- **studentInformation:** The state dictionary to populate the student information

The function will then attempt to populate the student information state dictionary by adding the following data:

- **Major:** The major that the student has taken
- **Specialization:** Any specialization declared by the student

CumulativeInfo.py

This module is responsible for storing the remainder of the student information. The function called is **trackCumulativeInformation(),** which is done by a thread and has the following parameters:

- **Text:** Contains a list of strings that has lines of text from the pdf
- **cumulativeInformation:** State dictionary to be populated

This function will then call other functions within the module in attempts to populate the remainder student information with the following data:

- **Cumulative GPA:** The cumulative GPA of the student throughout their entire undergraduate tenure
- **Cumulative Credits:** The number of credits the student has taken in total throughout their entire undergraduate tenure
- **Upper Division Credits:** The number of upper division classes the student has taken in the form of credits

ParseClasses.py

This module is responsible for populating the remaining state dictionaries. Once the student information dictionary has all the necessary data, the module will then use certain **keys** (particularly the **major** key) in order to distinguish between a CSE major and an ISE major. The primary function that is called is **majorClassTracker()** which takes in the following input:

- **Text:** Contains a list of strings that has information from the pdf
- **studentInformation**: Takes in the state dictionary to distinguish between CSE and ISE student
- **upperDivisionCourses:** Takes in the state dictionary in order to hold any upper division courses the student has taken regardless of major
- **lowerDivisionCourses:** Takes in the state dictionary in order to hold any lower division courses the student has taken regardless of major
- **technicalCourses:** Takes in the state dictionary in order to hold any technical courses that the student has taken regardless of major
- **specializeRequiredCourses:** Takes in a state dictionary that holds any required specialized courses an **ISE student** has taken
- **specializeOptionalCourses:** Takes in a state dictionary that holds any optional courses an **CSE student** has taken
- **mathRequiredCourses:** Takes in a state dictionary that holds any of the required math courses a student has taken regardless of major

MinimalClassObject.py

This module is responsible for defining a class titled **SimpleClassObject** with the following field variables:

- **courseName:** Represents the name of the course the student has taken
- **Credits:** The number of credits earned from that course
- **Grade:** The letter grade the student received for that course
- **Term:** The term (i.e. Fall, Spring, Summer, etc) in which the class was taken
- **Year:** The year in which the course was taken

The primary purpose of this is to organize the information about a class into an object so that it will be manageable to extract information about the object and put it in a table. This module is used in **ParseClasses.py**

SBCClassObject.py

This module is responsible for defining a class titled **SBCCourse** with the following field variables:

- **SBC:** A list holding the different SBCs that are granted from that course
- **courseName:** Represents the name of the course the student has taken
- **Credits:** The number of credits earned from that course
- **Grade:** The letter grade the student received for that course
- **Term:** The term (i.e. Fall, Spring, Summer, etc) in which the class was taken
- **Year:** The year in which the course was taken
- **Comments:** Any comments that are added to the course

The primary purpose of this is to organize the information about a class into an object so that it will be manageable to extract information about the object and put it in a table. This module is used in **ParseClasses.py**

UniversalClassObject.py

This module is responsible for defining a class titled **UniversalClassObject** with the following field variables:

- **courseName:** Represents the name of the course the student has taken
- **Credits:** The number of credits earned from that course
- **Grade:** The letter grade the student received for that course
- **Term:** The term (i.e. Fall, Spring, Summer, etc) in which the class was taken
- **Year:** The year in which the course was taken
- **Comments:** Any comments that are added to the course

The primary purpose of this is to organize the information about a class into an object so that it will be manageable to extract information about the object and put it in a table. This module is used in **ParseClasses.py**

CreateTabularDocument.py

This module is responsible for creating the tabular document. Specifically, this module creates at least three tabular documents that are displayed on the XLSX file in a sheet. The primary function that is called is titled **createDocument(),** which takes in the following parameters:

- **fileNameInput:** The name of the XLSX file to be created. This is based on the name of the PDF file that was used as input

This function will then call three separate functions and potentially another function such that each of these functions is responsible for producing a table that is going to be used to display the information.

CreateSingularSpecDocument.py

This module is responsible for displaying the specialization that the student has declared. If there is no specialization that is declared, then this module will not be enforced. However, if there is a specialization that is declared, then the **CreateTabularDocument.py** module will call the **createSingularSpecDocument()** function to produce such a table. This table shares a similar format as to the major requirements of the student, and thus will outline the potential classes that the student can take in order to fulfill the specialization.

CreateCSESpecDocument.py

This module is responsible for listing the possible specializations that the student within the CSE major can take. There are roughly five specializations that are offered to CSE majors, thus this module produces five tables to reflect all specializations. If a student were to take a class under one of the specializations then the module will utilize this information and update the information in the table in such a way that the satisfactory class will become visible on the table.

CreateISESpecDocument.py

This module is responsible for listing the possible specializations that the student within the ISE major can take. There are roughly five specializations that are offered to ISE majors, thus this module produces five tables to reflect all specializations. If a student were to take a class under one of the specializations then the module will utilize this information and update the information in the table in such a way that the satisfactory class will become visible on the table.