# Melissa Pothen
# Create Meeting Subsystem

## Task 1: Design

This type of system could be used in any Calendar or Scheduling application. When scheduling events, the user may want to know if there would be more meetings within a certain date range depending on which day of the week the user would use. My first implementation would be most helpful if the user is only trying to determine the number of a certain day of the week there is within a date range. My second implementation is designed to be a more useful Scheduling application. It is a more helpful approach because the user can schedule multiple events, view their calendar and still see how many occurrences of a certain day of a week occur between two dates.

## Task 2: Implementation

*Implementation 1: CreateMeeting1*
The first implementation would be most useful when the user has a text file with a list of start and end dates with the day of the week to schedule the meeting. The program reads in the file and the output printed lists the number of meetings on that day of the week in that period of time.

- Method Descriptions: Meeting Class
    - **Main method:**
        - The user is prompted to enter a file name. All the lines of the file are analyzed and the number of meetings for each line is printed with details of the meeting.
    - **countOccurrences:**
        - This method takes in a start and end date in the format of a string and an integer representing the day of the week. The method returns and integer that is the number of meetings the user would have on the specified day of the week in the specified date range. If the end date entered is before the start date, an Illegal Argument Exception is thrown.
    - **convertDate:**
        - This method takes in a string in the "YYYY-MM-DD" format and returns an array of three integers that represent the month, day and year of the date. If the string is not in the specified format, an Illegal Argument Exception is thrown.
    - **convertDay:**
        - This method takes in a string representing the day of the week and converts it to the corresponding integer (Sunday = 1, Monday = 2, …, Saturday = 7). If any other string is passed in, an Illegal Argument Exception is thrown.

*Implementation 2: CreateMeeting2*
The second implantation is an extension of the first implementation. The system is set up to be able to schedule single events and weekly events and also give feedback to the user. Instead of reading from a file, I chose to use a Scanner to prompt the user and get the user's input until the user decides to close the scheduler. All the meetings are stored in a HashMap that maps a date to an event. For this approach, I decided to create a Date, Time and Event class in addition to the main program, Meeting2 to better represents these aspects of a meeting.

- Date Class: This object represents a date. It has three attributes: the month, day and year.
- Time Class: This object represents a time. It has two attributes: the hour and the minute.
- Event Class: This object represents an event. It has three attributes: the date of the event, the start time of the event and the end time of the event.
- Method Descriptions: Meeting2 Class
    - **Main method:**
        - The user is given four options: 1) schedule a single event, 2) schedule a recurring event, 3) view their entire calendar with the meeting times listed and 4) to close the scheduler. If any other input is given, the user is prompted again. The scheduler is cleared each time the program is run.
    - **countOccurrences**
        - This method takes in a start and end date in the format of a string and an integer representing the day of the week. The method returns and integer that is the number of meetings the user would have on the specified day of the week in the specified date range. If the end date entered is before the start date, an Illegal Argument Exception is thrown.
        - All meetings within the date range are added to the scheduler.
    - **convertDate**
        - This method takes in a string in the "MM-DD-YYYY" format and returns a Date object. If the string is not in the specified format, an Illegal Argument Exception is thrown.
    - **convertDay**
        - This method takes in a string representing the day of the week and converts it to the corresponding integer (Sunday = 1, Monday = 2, …, Saturday = 7). If any other string is passed in, an Illegal Argument Exception is thrown.
    - **convertTime**
        - This method takes in a string in the "HH:MM,HH:MM" format that represents the start and end time of the event. An array of two Time objects (one representing the start and the other representing the end) is returned.
    - **scheduleEvent**
        - This method takes in a Date object and Event object and adds it to the scheduler HashMap

- **showCalendar**
    - This method prints out the entire scheduler, listing all the dates with meetings and the start and end times of each meeting.

## Task 3: Testing

*Implementation 1: CreateMeeting1*
To test this program, I edited the text file to include correctly formatted dates and incorrectly formatted dates. For the incorrectly formatted dates, an exception should be thrown. Some examples of the incorrect data I tested are listed below:

- 2020-01-01, 2019-01-01, Wednesday
- 202-01-02, 2019-01-01, Thursday
- 2020-01-02, 2020-03-03, 2

*Implementation 2: CreateMeeting2*
Since this program is based on the user input when the user is prompted, I tested the code by entering some incorrectly formatted and correctly formatted dates. For the incorrectly formatted dates, an exception should be thrown. The response to each prompt I entered is separated by a "|". Some examples of the incorrect data I tested are listed below:

- 5
    - Not an option for the first prompt
- 1 | 202-01-02
- 2 | 202-010-30 | 487-2030 | Monday | 29:90,49:01