```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

public class App {

    private static boolean continueExec = true;
    static Database db;
    static Scanner scanner = new Scanner(System.in);
    static ArrayList<ScheduleFile> scheduleFile = new ArrayList<ScheduleFile>
();

    public static void main(String[] args) throws Exception {
        if (checkDatabaseExists()) {
            // System.out.println("Database exists");
            // Create a new instance of the Database
            db = new Database();
            if (requestDBReset()) {
                // Drop
                db.drop();
                // Migrate the DB tables first to ensure they exist
                db.migrate();
                // Seed the tables
                db.seed();
                // Continue
                promptUserMenu();
            } else {
                // Continue
                promptUserMenu();
            }
        } else {
            // File does not exist
            // System.out.println("Database does not exist");
            // Create a new database file
            createDatabaseFile();

            // Initialize the database
            Database db = new Database();
            db.migrate();
            db.seed();
        }
    }

    private static boolean checkDatabaseExists() {
        File file = new File("db.sqlite");
        return file.exists();
    }

    private static void promptDatabaseReset() {
```

```
    private static void promptDatabaseReset() {
        System.out.print("\nWould you like to reset the database? (y/n): ");
    }

    private static boolean requestDBReset() {
        // Prompt request to user
        promptDatabaseReset();

        String response = scanner.nextLine();

        // Check if user input is y or n
        if (response.equals("y")) {
            // System.out.println("Requesting resetting of database");
            return true;
        } else if (response.equals("n")) {
            // System.out.println("Not resetting database");
            return false;
        } else {
            System.out.println("Invalid response");
            return requestDBReset();
        }
    }

    private static void createDatabaseFile() {
        // Create a new file
        new File("db.sqlite");

        // Confirm that the file was created
        if (checkDatabaseExists()) {
            // System.out.println("Database file created");
        } else {
            // System.out.println("Database file not created");
        }
    }

    private static void promptUserMenu() {
        // While the continueExec boolean is true, continue to prompt the
user
        // for input regarding which menu option they want.
        while (continueExec) {
            clearConsole();
            System.out.println("\nPlease select an option:");
            System.out.println("1. Import Schedule File");
            System.out.println("2. Report - Date/Time");
            System.out.println("3. Report - Faculty Members");
            System.out.println("4. Report - Classes");
            System.out.println("5. Exit");
            System.out.println("6. Exit and Destroy Database");
            System.out.print("Enter your choice: ");

            // Get the user's choice and store it in the variable
```

```
            // Get the user input and determine what option they selected
            int option = scanner.nextInt();

            switch (option) {
                case 1:
                    // Import Schedule File
                    // System.out.println("Importing schedule file");
                    readScheduleFile();
                    break;
                case 2:
                    // Report - Date/Time
                    // System.out.println("Report - Date/Time");
                    reportDateTime();
                    break;
                case 3:
                    // Report - Faculty Members
                    // System.out.println("Report - Faculty Members");
                    reportFacultyMembers();
                    break;
                case 4:
                    // Report - Classes
                    // System.out.println("Report - Classes");
                    reportClasses();
                    break;
                case 5:
                    // Exit
                    // System.out.println("Exiting");
                    System.exit(0);
                    break;
                case 6:
                    // Exit & Destroy Database
                    // System.out.println("Exiting and destroying database");
                    db.drop();
                    deleteDBFile();
                    System.exit(0);
                    break;
                default:
                    System.out.println("Invalid option");
                    break;
            }
        }
    }

    private static void deleteDBFile() {
        File file = new File("db.sqlite");
        file.delete();
    }

    private static void readScheduleFile() {
        // Prompt user for file name
```

```
        System.out.print("Enter the file path: ");
        String response = scanner.next();
        if (checkFileExists(response)) {
            // File exists
            // System.out.println("File exists");

            // Read the schedule file and store the data into the
            // scheduleFile ArrayList
            try {
                readScheduleFile(response);

                if (scheduleFile.size() > 0) {
                    // System.out.println("Schedule file has data");
                    // System.out.println("Schedule file size: " +
scheduleFile.size());
                    importScheduleFile();
                } else {
                    System.out.println("Schedule file is empty");
                }
            } catch (FileNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        } else {
            // File does not exist
            System.out.println("File does not exist");
            readScheduleFile();
        }
    }

    private static boolean checkFileExists(String filePath) {
        File file = new File(filePath);
        return file.exists();
    }

    private static void readScheduleFile(String filePath) throws
FileNotFoundException {
        // Read the file
        File file = new File(filePath);
        Scanner scanner = new Scanner(file);
        while (scanner.hasNextLine()) {
            String line = scanner.nextLine();
            // Break the line apart by tabs
            String[] lineArray = line.split("\t");
            // Create a new ScheduleFile object
            // I am assuming that the file is formatted correctly and data is
clean
            scheduleFile.add(new ScheduleFile(lineArray[0],
                    lineArray[1],
```

```
                    separateDays(lineArray[2]),
                    convert12HrTo24Hr(addAmOrPmToTime(lineArray[3])),
                    convert12HrTo24Hr(addAmOrPmToTime(lineArray[4]))));
            // Print what is in the array
            // System.out.println(lineArray[0] + " " + lineArray[1] + " " +
lineArray[2] + "
            // " + lineArray[3] + " " + lineArray[4]);
        }
        // Close the scanner/file
        scanner.close();
    }

    private static void importScheduleFile() {
        // Loop through the scheduleFile ArrayList and insert the data into
the database
        for (ScheduleFile sf : scheduleFile) {
            // Insert the data into the database
            db.insertScheduleFile(sf);
        }
        scheduleFile.clear();
    }

    private static String addAmOrPmToTime(String time) {
        // Since the times are formatted like this: 8:00
        // I want to distinguish if they're supposed to be AM or PM
        // So I can convert them to 24 hour time

        // Split the time into an array
        String[] timeArray = time.split(":");
        // Get the hour
        int hour = Integer.parseInt(timeArray[0]);

        // If the hour is more than 7 and less than 12
        // Then it's AM
        // If its more than 12 and less than 8
        // Then it's PM
        // I'm writing this based off the given JITClasses.txt file
        // and the explaination of the available timeslots.

        // I'm doing this so I can easily subtract the dates
        // to see if they overlap

        if (hour > 7 && hour < 12) {
            // It's AM
            return time + " AM";
        } else {
            // It's PM
            return time + " PM";
        }
    }
```

```
    }

    private static String convert12HrTo24Hr(String time) {
        // A string of format hh:mm AM/PM
        // to 24 hour format and return it
        // as a string

        // Split the string into an array
        String[] timeArray = time.split(":");
        String hour = timeArray[0];
        String minute = timeArray[1].substring(0, 2);
        String ampm = timeArray[1].substring(3, 5);

        // Convert the hour to 24 hour format
        if (ampm.equals("PM")) {
            if (hour.equals("12")) {
                hour = "12";
            } else {
                hour = Integer.toString(Integer.parseInt(hour) + 12);
            }
        }

        // Format h:mm to hh:mm
        if (hour.length() == 1) {
            hour = "0" + hour;
        }

        // Return the 24 hour format
        return hour + ":" + minute + ":00";
    }

    private static String separateDays(String days) {
        // Days are formatted like this: M,T,W,R,F,MW, or TR

        // Check if the days is in format MW OR TR
        // if so, then separate them M AND w OR T AND R

        // I'm doing this so when I query the database for dates
        // I can easily use the IN function to insert an array
        // of days. So I can check if an M class is on the same day
        // as a MW class
        if (days.length() == 2) {
            return days.substring(0, 1) + "," + days.substring(1, 2);
        } else {
            return days;
        }
    }

    private static void reportDateTime() {
        clearConsole();
```

```
292
293         ArrayList<Schedule> schedule = db.getSchedule();
294
295         System.out.format("%-8s%-10s%-20s%-15s\n", "Days", "Classroom",
    "Time", "Course");
296
297         // Loop through the schedule ArrayList
298         for (Schedule s : schedule) {
299             // Format a string as the following and print it
300             // Date: days | Time: startTime:endTime | Class: course |
    Classroom: classroom
301             System.out.format("%-8s%-10s%-20s%-15s\n", s.getDays(),
    s.getClassroom(), (s.getStartTime() + "-" + s.getEndTime()), s.getCourse());
302
303
304             // System.out.println("Date: " + s.getDays() + " | Time: " +
    s.getStartTime() + ":" + s.getEndTime() + " | Class: " + s.getCourse());
305             // Print the data
306             // System.out.println(s.getDays() + " " + s.getStartTime() + " -
    " + s.getEndTime() + " " + s.getCourse());
307         }
308
309         // Prompt user to press x to exit
310         Boolean exit = false;
311         System.out.println("Press x to exit");
312
313         // Listen for the user to just press the x key
314         while (!exit) {
315             Scanner scanner = new Scanner(System.in);
316             String input = scanner.nextLine();
317             if (input.equals("x")) {
318                 exit = true;
319             }
320         }
321     }
322
323     private static void reportFacultyMembers() {
324         clearConsole();
325
326         ArrayList<Professor> professors = db.getProfessors();
327
328         System.out.format("%-10s%-8s%-10s%-20s%-15s\n", "Professor", "Days",
    "Classroom", "Time", "Course");
329
330         // Loop through the schedule ArrayList
331         for (Professor prof : professors) {
332             int totalCredits = 0; // I could of done this via query
333             // but just decided to do this here instead
334             // Format a string as the following and print it
335             // Date: Professor | Time: startTime:endTime | Class: course
```

```
            System.out.format("%-10s\n", prof.getName());
            ArrayList<Schedule> schedule =
db.getScheduleByProfessor(prof.getId());
            for (Schedule s : schedule) {
                totalCredits += s.getCredits();
                System.out.format("%-10s%-8s%-10s%-20s%-15s\n", "",
s.getDays(), s.getClassroom(), (s.getStartTime() + "-" + s.getEndTime()),
s.getCourse());
            }
            System.out.format("%-10s%-8s%-10s%-20s%-15s\n", "", "", "", "",
"Total Credits: " + totalCredits);
        }

        // Prompt user to press x to exit
        Boolean exit = false;
        System.out.println("Press x to exit");

        // Listen for the user to just press the x key
        while (!exit) {
            Scanner scanner = new Scanner(System.in);
            String input = scanner.nextLine();
            if (input.equals("x")) {
                exit = true;
            }
        }
    }

    private static void reportClasses() {
        clearConsole();

        ArrayList<Course> courses = db.getCourses();

        System.out.format("%-10s%-10s%-10s\n", "Course", "Classroom",
"Capacity");

        // Loop through the schedule ArrayList
        for (Course course : courses) {
            // Format a string as the following and print it
            // Date: Course | Section: section | Capacity: capacity
            System.out.format("%-10s\n", course.getName());
            ArrayList<Schedule> schedule =
db.getScheduleByCourse(course.getId());
            for (Schedule s : schedule) {
                System.out.format("%-10s%-10s%-10s\n", "", s.getClassroom(),
s.getCapacity());
            }
        }

        // Prompt user to press x to exit
        Boolean exit = false;
```

```
        boolean exit = false;
        System.out.println("Press x to exit");

        // Listen for the user to just press the x key
        while (!exit) {
            Scanner scanner = new Scanner(System.in);
            String input = scanner.nextLine();
            if (input.equals("x")) {
                exit = true;
            }
        }
    }

    private static void clearConsole() {
        System.out.print("\033\143");
    }
}
```