CSC 142 **Project 4 - Date.java and DateClient.java** **Due: See Canvas**

Description: This project focuses on creating a Java class, Date.java, along with a client class, DateCleint.java, which will contain code that utilizes your Date.java class. You will submit both files (each with appropriate commenting). A very similar project is described in the Programming Projects section at the end of chapter 8, which you may find helpful as reference.

Use (your own) Java class file Date.java, and a companion DateClient.java file to perform the following:

- Ask user to enter Today's date in the form (month day year): *2 28 2018*
- Ask user to enter Birthday in the form (month day year): *11 30 1990*
- Output the following:

  The date they were born in the form (year/month/day).
  The day of the week they were born (Sunday – Saturday).
  The number of days between their birthday and today's date (their age in days).
  Example: You were born on 1990/11/30, which was a Friday.
  You are 9952 days old.

To do this you will need to design, implement and test a Java class file, **Date.java**:

Date.java is a class of objects which you write, then utilize in DateClient.java to solve the problem above. The Date.java class needs to have the following methods (the object calling the methods below is referenced by ***this Date object*** in all the descriptions, as needed).

- **public Date(int year, int month, int day)**
  //constructor to create new Date object with given year, month and day.

- **public Date()**
  //constructor to create new default Date object with date January 1, 1753 *(1753 1 1).*
  (As a side note, January 1, 1753 was a Monday, which will be used later).

- **public int getYear()**
  //accessor method to return this Date object's year (between 1753 & 9999).

- **public int getMonth()**
  //accessor method to return this Date object's month (between 1 & 12).

- **public int getDay()**
  //accessor method to return this Date object's day (between 1 & 31).

- **public String toString()**
  //accessor method to return String of this Date object's field in form *Year/Month/Day (2018/2/28).*

- **public boolean equals(Date otherDate)**
  //accessor method to return true if this Date object equals otherDate object (false otherwise).

- **public Boolean isLeapYear()**
  //accessor method to return true if this Date object's year field is leap year (false otherwise). Leap years are all years divisible by 4, except those also divisible by 100, but includes those that are divisible by 400. [1756, 2004, 1600 & 2000 are leap years, but 1755, 1900 & 2100 are not].

- **pubic void nextDay()**
  //mutator method that advances this Date object's fields to represent the next date. If this Date object's fields are 2018/2/28, then calling nextDay() method will advance it to 2018/3/1. Similarly, if starting with 2018/12/31, calling nextDay() method will advance it to 2019/1/1. In other words, nextDay() advances this Date object's day, month and/or year, as needed, to the next day.

- **public int advanceTo(Date endDate)**
  //mutator method that advances this Date object's fields to the endDate object's fields and returns the number of days between the two dates. Like nextDay(), advanceTo(Date endDate) can advance days, months and/or years as needed. It may be assumed that the endDate object's date is after this Date object's date.

- **public String getDayOfWeek()**
  //accessor method that returns a string giving the day of the week this Date object's date falls on (*Sunday thru Saturday*). (We will need to know the day of a start date, for which we can use that January 1, 1753 was a Monday.) *This method should be written utilizing the other methods listed above—no outside code from internet or elsewhere is needed.*

  None of the methods within the Date.java class should interact directly with the user or print anything to the console. All interaction with the user (and the console) should be contained within DateClient.java class.

  You may not use any of Java's date-related classes, nor should you use any code or fragments of code obtained through outside or internet research. The concepts needed here rely on content from chapters 1-8 in our text, and development of these ideas independent of outside resources is the purpose of this assignment.

  Development strategy:
  - Write constructor and get*() methods in Date.java class.
  - Write a DateTesting.java class to test all methods utilizing Date.java class.
  - Write toString, equals & isLeapYear accessor methods in Date.java class.
  - Write nextDay, (you may want to write an additional intermediate method that returns the number of days in this Date object's month. You will also need to call isLeapYear within this method.)
  - Write advanceTo by utilizing other methods written within Date.java class.
  - Write getDaysOfWeek, again utilizing other methods written within the Date.java class.

  Grading Rubric: See Canvas Rubric for score break-down, but good programming styles (commenting, reducing redundancy) is expected. In addition, points will be awarded for each portion of Date.java and DateClient.java classes that is developed.