Project 02: Roommate Compatibility Program
CSE-021-01 **Points: 100**

Overview

In this project you will be creating a roommate compatibility program using four classes. You may work in pairs or by yourself. The expectation is that you work on the project outside of lab time, but you may certainly use any extra time in lab. Your partner can be in any section. Both of you will make separate submissions and demonstrate individually, each noting the collaborator/partner in the "Comments ..." text-box in the submission page.

Project Requirements

You have been hired by the Empire State University's Office of Housing and Residence Life to prepare a program which measures compatibility between potential roommates. You are given a file with the list of students. For each student, you are given their name, gender and birthday. You will also have their preference for quiet time, music, reading and chatting. Using this information, you can then determine the compatibility between two people for rooming together, according to a formula developed by the Office of Housing and Residence Life. You will need to create four classes to do this project: `Match.java`, `Student.java`, `Date.java` and `Preference.java`. We now describe how each of these classes should function.

`Student` class

Write a Student class that stores the following information about a person as instance variables:

- `name`: of type `String` to store the student name
- `gender`: of type `char` to store the sex of student, '`M`' or '`F`'
- `birthdate`: of type `Date` (explained later) to store the student's date of birth
- `pref`: of type `Preference` (explained later) to store their answers to four ***activities***
- `matched`: of type `Boolean` to convey whether there is a match

The Student class should at least have the following (you may add more as you deem fit):

- A constructor which sets the instance variables to their appropriate input parameters (4 inputs).
- Accessor methods for each of the 5 instance variables.
- Mutator method for `matched` to set it to `true` after a successful match with another student (note: should set the variable on BOTH students).
- A `compare(Student st)` method that returns the compatibility score between the current student (the object of `Student` class calling this method) and the `Student` input parameter, `st`.

The calculation of the compatibility score returned by `compare` is as follows:

- Highest score is `100` and lowest is `0`
- Different genders get a score `0` (only match same gender students as roommates)
- The formula for computing the compatibility score is: `(40 - Preferences) + (60 - AgeDifference)`, where
  - `Preferences` is the absolute differences in each of the 4 ***activities*** added together
  - `AgeDifference` is the number of months between two birthdates with the maximum being `5` years (`60` months)

Note: Use `Math.abs()` method call to calculate absolute value, e.g., `Math.abs(age2 - age1);`.

`Date` class

The class named `Date` should have the following instance variables (we assume there are no leap years):

- `year`: of type `int` in the range `1900` to `3000`
- `month`: in the range `1` to `12` (January is `1`)
- `day`: in the range `1` to `31` and appropriate to the month (you can assume February is always `28` days)

The `Date` class should at least have the following methods:

- A constructor which sets the instance variables to their appropriate input parameters (3 inputs)
- Accessor methods for each of the 3 instance variables
- A `compare(Date dt)` method that returns the difference (in number of months) between the current birthdate (the object of `Date` class calling this method) and the `Date` input parameter, `dt`.

The calculation of difference between dates of birth returned by `compare` is as follows:

- First calculate the absolute difference between the birth years, and multiply the difference by `365` to get `yearsDifferenceInDays`.
- We provide you with a method called `dayOfYear()` below that counts the number of days from the beginning of the birth year till a ***birthday*** (we distinguish between a ***birthday*** comprising of just the birth month and day, and ***birthdate*** comprising of the birth month, day and year).

  For instance, `dayOfYear()` returns `114` for someone born on April 24, 1996, since there are `114` days between January 1 and April 24 (and ignoring that 1996 was a leap year).

  Use `dayOfYear()` to calculate the absolute number of days between the ***birthdays*** to get `daysDifference`.
- Compute `totalDifference` as the absolute difference between `yearsDifferenceInDays` and `daysDifference`. And finally, divide the total number of days differential, `totalDifference`, by `30` to approximate the number of difference in birth months, `monthsDifference`.
- If the month difference is higher than `60` then you should only return `60` as a max value.

The `dayOfYear()` method of the `Date` class is as follows:

```
public int dayOfYear() {
   int totalDays = 0;
   switch (month) {
      case 12: totalDays += 30;
      case 11: totalDays += 31;
      case 10: totalDays += 30;
      case 9 : totalDays += 31;
      case 8 : totalDays += 31;
      case 7 : totalDays += 30;
      case 6 : totalDays += 31;
      case 5 : totalDays += 30;
      case 4 : totalDays += 31;
      case 3 : totalDays += 28;
      case 2 : totalDays += 31;
   }
   totalDays += day;
   return totalDays;
}
```

`Preference` class

Write a `Preference` class that records preferences for different types of activities with the following instance variables:

- `quietTime`
- `music`
- `reading`
- `chatting`

Each of these is of type `int` and must contain values in the range `0` to `10` (inclusive). A value of `0` means the person hates the activity. A value of `10` means the person loves the activity.

The `Preference` class should at least have the following methods:

- A constructor which sets the instance variables to their appropriate input parameters (4 inputs)
- Accessor methods for each of the 4 instance variables
- A `compare(Preference pref)` method that returns the difference between the current preference (the object of `Preference` class calling this method) and the `Preference` input parameter, `pref`.

To calculate the difference between preferences returned by `compare`, you simply sum up the absolute differences in the 4 activities.

`Match` class

Write a `Match` class that (with `main()`) that performs the following functions:

- Create an array of `Students` (max = `100`)
- Read all student information (tab delimited) from a text file
  - Name (`String`)
  - Gender (`char`)
  - Birthdate (`String`: Month-Day-Year, hyphen '-' delimited)
  - Quiet time preference (`int` ranging from `0` to `10`)
  - Music preference (`int` ranging from `0` to `10`)
  - Reading preference (`int` ranging from `0` to `10`)
  - Chatting preference (`int` ranging from `0` to `10`)
- For each line in the text file
  - Extract the information for each student
  - Create a `Student` object pointed to be an entry in the array
  - Keep a count of actual number of students (less than `100`)
- For each student in the array (after completely done reading)
  - Check against every other student (assuming they are not matched already) their compatibility scores
  - Find the Best Score and Best Match person
  - Match the two roommates up and print out the result

Note: Your algorithm for matching should work something like this:

```
Foreach student NOT currently matched
   Foreach rest of students NOT currently matched
      currentScore = studentA.compare(studentB)
      if the currentScore is better than MaxScore
         bestMatchStudent is student
         bestMatchScore is currentScore

   studentA is now Matched
   bestMatchStudent is now Matched
```

Also note that you have an array of students presumably. So `studentA.compare(studentB)` will actually look something like this:

```
if(!students[j].getMatched()) // student not matched already
   currentScore = students[i].compare(students[j]);
```

In the code above, `i` is the index of the outer loop, `j` is the index of the inner loop, and `getMatched()` is the accessor method for matched.

Testing

We have included two test files `Students.txt` and `FullRoster.txt`. To give you an idea of correct behavior, your program should generate the following expected output for `Students.txt`:

```
Abey matches with Melissa with the score 60
John matches with Jeff with the score 100
Craig has no matches.
```

Copy and paste the output of your console for **both** test files into `output.log`. To be precise, you will create an empty text file called `output.txt`, copy and paste the console output for the two program runs in Eclipse (corresponding to the two test files) into `output.txt`, and rename the file to `output.log`.

What to hand in

When you are done with this project, submit all your work through CatCourses.

Before you submit, make sure you have done the following:

- Completed `Student.java`, `Date.java`, `Preference.java` and `Match.java`.
- Placed your console output for the two runs corresponding to the two test files in `output.log`.
- Attached the `Student.java`, `Date.java`, `Preference.java`, `Match.java` and `output.log` files in the submission page.
- Filled in your collaborator's name (if any) in the "Comments…" text-box at the submission page.

Also, remember to demonstrate your code to the TA or instructor before the end of the demo period.