Laboratory 3: Pair Programming

Name	:	______________________
Student Number	:	______________________
Laboratory Time	:	______________________

Objectives

- Gain experience with pair programming
- Write code to satisfy test cases that have already been prepared
- Work with two key practices from Extreme Programming

Preamble

Pair programming is perhaps the most controversial aspect of Extreme Programming [1, 2], although it is an interesting development technique by itself. Simply put, pair programming is a technique where two people work as a unit at a single computer (with one keyboard, one mouse, and one monitor) to solve a development problem (design, implementation, testing, etc.). One person "drives," while the other person "navigates"; the driver makes sure the car doesn't go off the road, while the navigator makes sure that they get to the destination. Often these roles can switch during a single working session. Pairs are fluid; they can change throughout the day on a team where everyone uses the technique. Some of the benefits of pair programming are higher quality code (fewer defects) resulting in improved overall productivity and better communication and cohesiveness among team members.

In this lab, you will solve a small programming problem using pair programming. The Extreme Programming web site has the following advice, "The best way to pair program is to just sit side by side in front of the monitor. Position the monitor so both people can see it. Slide the key board and mouse back and forth. One person types and thinks tactically about the method being created, while the other thinks strategically about how that method fits into the class. It takes time to get used to pair programming so don't worry if it feels awkward at first" [3].

[1] Laurie Williams, Robert R. Kessler, Ward Cunningham, and Ron Jeffries. **"Strengthening the Case for Pair Programming"** IEEE Software, pp. 19-25 July/August, 2000.
http://ieeexplore.ieee.org/iel5/52/18557/00854064.pdf?tp=&arnumber=854064&isnumber=18557
[2] Roy W. Miller, **"Demystifying Extreme Programming: Winning with a pair"**
http://www-106.ibm.com/developerworks/java/library/j-xp03113.html
[3] Don Wells**, "Extreme Rules: Pair Programming"**
http://www.extremeprogramming.org/rules/pair.html

Grading Checklist

- ☐ Participated in pair programming
- ☐ DateDiff program passes al tests in DateTester
- ☐ Written Report turned in to Checkmate
    - ☐ Question 1
    - ☐ Question 2
    - ☐ Question 3

TA Initials: ____

Instructions for the Laboratory

Put your answers to Questions 1, 2, and 3 in a file called **Lab3.doc** or **Lab3.pdf**, and turn it in via Checkmate under **IN4MATX111**, assignment **Lab3**. This is due by the end of the day.
You won't turn in your Java source or compiled code.

Section I: Preparation

Question 1. Have you used Pair Programming before? If so, was it a positive or negative experience?

Question 2. What are your expectations for this lab? Do you expect to like or dislike pair programming? What problems or benefits do you expect to encounter?

Section 2: Observations

Step 1. Find a partner. Log on to one of your accounts.

Step 2. Complete the programming problem stated below. Use Eclipse to write and run your Java program. Two files have been provided for you on the course web page. The first, *DateDiff.java*, contains a skeleton program to get you started. The other is *DateTester.class*, which will test whether your solution is complete. Be sure to switch between "driver" and "navigator" roles during the lab. In your code, use meaningful names for identifiers and add appropriate comments.

Given two dates, where m1, d1, and y1 is the month, day, and year of the first date, and m2, d2, and y2 is the month, day, and year of the second day, determine the number of days (inclusive) between the two dates.

Remember to account for leap years. A year is a leap year if:
The year is divisible by 4 but not divisible by 100,
OR the year is divisible by 400.
For example, 1984 and 2000 are leap years, but 1900 is not.

You may assume that the input is a valid date, and that the year will be between 1 and 9999 inclusive. You may assume the second date will not come before the first date.

Examples:

```
new DateDiff().difference(1,1,2002,1,2,2003) -> 367 days
new DateDiff().difference(1,1,2002,1,1,2003) -> 366 days
new DateDiff().difference(2,1,2004,3,1,2004) -> 30 days
```

Step 3. When your program passes all the tests in *DateTester*, show your solution to your TA.

Section 3: Conclusions

Question 3. Write a short testimonial about pair programming. It should be 2-3 paragraphs long. You should give a brief description of your experience, describing the good points and the bad points. It does not have to be a glowing, positive review. The following web page has good examples of testimonials:
http://www.c2.com/cgi/wiki?ProgrammingInPairsTestimonials