Project 1-2
Date Calculation[*]

Jiashuo Wang
5100309436

April 3, 2013

Instructor: Ling Gong

I. Objective

This problem is much more flexible, any reasonable implementation is acceptable. Please implement a class called MyDate which can represent a date. The class should have at least three int fields: year, month, day, and several methods to handle them. Please make sure the values set to these fields are all valid. Pay attention to the special cases: some month can only 28/29 days, and some month can have 31 days.

Other the basic methods, there are two additional required methods:

- **laterThan(MyDate date)**
  This method decides whether I am a later date than the passed-in parameter date, and return true or false. For example, if A is a MyDate object representing date 9/15/2004, and B representing date 9/22/2004, A.laterThan(B) should return false, while B.laterThan(A) should return true, and A.laterThan(A) should also return false.
- **dayDifference(MyDate date1, MyDate date2)**
  This method should be a static method. It takes two dates and returns an integer indicating how many days are there between them. For example, the days between 9/15/2004 and 9/22/2004 is 7. date2 shouldn't have to be larger (later) than date1, and you need to judge it yourself. Hint: you may try use previous method inside this method.

To test the code, you can write you own main() function, or any other Class you want to. You can do console input, or hard-code the date creation inside your code is also fine here. The purpose of this assignment is just let you get familiar with class definition and manipulate with objects.

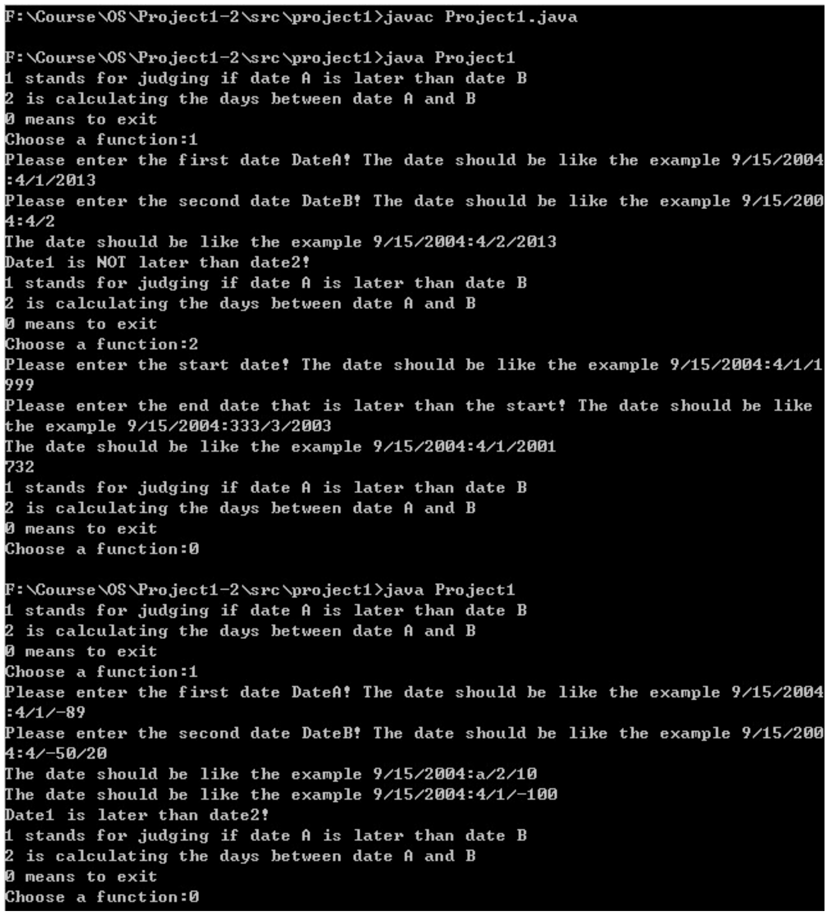
II. Algorithm

The two methods are the most important problem to be solve in this project.

II.1 laterThan(MyDate date)

This method is much easier to implement. Simply comparing the year, month and day in order is enough to judge which is the later one.

[*]Designed by LaTeX



```
F:\Course\OS\Project1-2\src\project1>javac Project1.java

F:\Course\OS\Project1-2\src\project1>java Project1
1 stands for judging if date A is later than date B
2 is calculating the days between date A and B
0 means to exit
Choose a function:1
Please enter the first date DateA! The date should be like the example 9/15/2004
:4/1/2013
Please enter the second date DateB! The date should be like the example 9/15/200
4:4/2
The date should be like the example 9/15/2004:4/2/2013
Date1 is NOT later than date2!
1 stands for judging if date A is later than date B
2 is calculating the days between date A and B
0 means to exit
Choose a function:2
Please enter the start date! The date should be like the example 9/15/2004:4/1/1
999
Please enter the end date that is later than the start! The date should be like
the example 9/15/2004:333/3/2003
The date should be like the example 9/15/2004:4/1/2001
732
1 stands for judging if date A is later than date B
2 is calculating the days between date A and B
0 means to exit
Choose a function:0

F:\Course\OS\Project1-2\src\project1>java Project1
1 stands for judging if date A is later than date B
2 is calculating the days between date A and B
0 means to exit
Choose a function:1
Please enter the first date DateA! The date should be like the example 9/15/2004
:4/1/-89
Please enter the second date DateB! The date should be like the example 9/15/200
4:4/-50/20
The date should be like the example 9/15/2004:a/2/10
The date should be like the example 9/15/2004:4/1/-100
Date1 is later than date2!
1 stands for judging if date A is later than date B
2 is calculating the days between date A and B
0 means to exit
Choose a function:0
```

Figure 1: *Screenshots of Date Calculation*

II.2 dayDifference(MyDate date1, MyDate date2)

In the class of last Friday evening, we learned a new way to solve this kind of problem, which is transport the date into the form of Julian Day and then just making a subtraction is OK.

After class, I thought finding a new way is much more interesting. So I tried to search the Internet to find a new way to do so. Then I found the Java class **Calendar**, which is also in the java.util. So in my code, I do this job in the following steps:

- Turn the format of date into the format of *Calendar*;
- Use the For statement to traverse the days between the two dates and the variance *daysBetween* will add by 1;
- After the traversing, we will get the days between the two dates.

Before doing these, it is necessary, as mentioned in the requirement, to judge if the end date is later than the start date, which is easier because the first method will help to do so.

II.3 More details

Before using the date, we have to judge if the format of date is valid to be used. So we make a method *setDate* first to help us check it.

First of all, I use one of the method, $Strings1[] = s.split("/")$, of the class *String* to split the date into three part, month, day, year. If there are not 3 parts judged by $if(s1.length! = 3)$, you have to input again. Then I use the method, $s1[0].matches("[0-9]1,2")$, which was shown by our TA last Friday, to check the format for the first time. For further check, I use some If statements to judge if it is reasonable. If not, users have to input the date again.

III. Results and Conclusions

III.1 Environment

- Windows 8
- NetBeans IDE 7.3

III.2 Screenshots of the result

Use Command Line to compile and execute the program in Figure 1.

III.3 Thoughts

Finding a new way to solve an old problem is interesting!