Problem Set 4

Tim Magoun and Aravind Koneru

Compiled on Friday 29th July, 2016 at 15:50

Do these problems for additional practice and challenge

Reminders:

- Make sure that problem 2 and 3 are both methods
- Submit these problems to Dr. Rogers no later than Tuesday, August 2 by 9:00 am
- Email the .java files as we discussed in class instead of copying the code into the email
- If you don't solve every problem, it's fine. Just send us what you did. That being said, please refrain from copying the answers from online. All the instructors are aware that many of the problems that we give you can be found on StackOverflow and other community sites AND we do notice when a solution has been plagiarized.
- There is no challenge question this week. Instead, prioritize doing as much of problem 1 as you can before moving onto problems 2 and 3.

1. **Date**: This problem will take the majority of your time this week. All the details are in the code as comments. Try to complete as many of the methods as you can.

```
package homework2;

/* Date.java */

import java.io.*;

public class Date {

        int month, day, year;
```

```java
/** Constructs a date with the given month, day and year.  If the date is
 *  not valid, the entire program will halt with an error message.
 *  @param month is a month, numbered in the range 1...12.
 *  @param day is between 1 and the number of days in the given month.
 *  @param year is the year in question, with no digits omitted.
 */
public Date(int _month, int _day, int _year) {
        month = _month;
        day = _day;
        year = _year;
}

/** Checks whether the given year is a leap year.
 *  @return true if and only if the input year is a leap year.
 */
public static boolean isLeapYear(int year) {
  return true;                        // replace this line with your solution
}

/** Returns the number of days in a given month.
 *  @param month is a month, numbered in the range 1...12.
 *  @param year is the year in question, with no digits omitted.
 *  @return the number of days in the given month.
 */
public static int daysInMonth(int month, int year) {
  return 0;                           // replace this line with your solution
}

/** Checks whether the given date is valid.
 *  @return true if and only if month/day/year constitute a valid date.
 *
 *  Years prior to A.D. 1 are NOT valid.
 */
public static boolean isValidDate(int month, int day, int year) {
  return true;                        // replace this line with your solution
}

/** Returns a string representation of this date in the form month/day/year.
 *  The month, day, and year are expressed in full as integers; for example,
 *  12/7/2006 or 3/21/407.
 *  @return a String representation of this date.
```

```
   */
  public String toString() {
    return "stuff";                    // replace this line with your solution
  }

  /** Determines whether this Date is before the Date d.
   *  @return true if and only if this Date is before d.
   */
  public boolean isBefore(Date d) {
    return true;                       // replace this line with your solution
  }

  /** Determines whether this Date is after the Date d.
   *  @return true if and only if this Date is after d.
   */
  public boolean isAfter(Date d) {
    return true;                       // replace this line with your solution
  }

  /** Returns the number of this Date in the year.
   *  @return a number n in the range 1...366, inclusive, such that this Date
   *  is the nth day of its year.  (366 is used only for December 31 in a leap
   *  year.)
   */
  public int dayInYear() {
    return 0;                          // replace this line with your solution
  }

  /** Determines the difference in days between d and this Date.  For example,
   *  if this Date is 12/15/2012 and d is 12/14/2012, the difference is 1.
   *  If this Date occurs before d, the result is negative.
   *  @return the difference in days between d and this date.
   */
  public int difference(Date d) {
    return 0;                          // replace this line with your solution
  }

  public static void main(String[] argv) {
        //FOR TESTING        YOUR Code
  }
}
```

2. **isPrime(int x)**: We did this problem a few weeks ago, but now convert it into a method that accepts an `int` and returns a `boolean`. For example, I should be able to call `isPrime(3)` and it would return `true`.

3. **printPrimes(int x)**: This method should print out the first x many primes. For example, I should be able to call `printPrimes(5)` and it would print out the first 5 prime numbers.

   Hint: Consider using the `isPrime` method to solve this problem.