Lesson 4 Solutions

Tim Magoun and Aravind Koneru

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

```
package week4;

/* Date.java */

import java.io.*;

public class DateS {

        /* Put your private data fields here. */
        private int month, day, year;

        /** 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 DateS(int month, int day, int year) {
                this.month = month;
                this.day = day;
                this.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 (year % 4 == 0);
```

```
}

/** 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) {
        if(month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month
                return 31;
        }else{
                //for February
                if(month == 2){
                        if(isLeapYear(year)) return 29;
                        else return 28;
                }
                return 30;
        }
}

/** 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) {
        if(month < 13 && month > 0){
                if(day > 0 && day<= daysInMonth(month, year)){
                        if(year > 0){
                                return true;
                        }
                }
        }
        return false;
}

/** Determines whether this Date is before the Date d.
 *  @return true if and only if this Date is before d.
 */
public boolean isBefore(DateS d) {
        if(year < d.year){
                return true;
        }
        else if(year == d.year){
                if(month < d.month){
                        return true;
                }
                else if(month == d.month){
                        if(day < d.day){
                                return true;
                        }
                }
        }
        return false;
}

/** Determines whether this Date is after the Date d.
 *  @return true if and only if this Date is after d.
 */
public boolean isAfter(DateS d) {
        return !isBefore(d) && (!d.equals(this));

}

public boolean equals(DateS d){
        if(d.day == this.day && d.month == this.month && d.year == this.year){ return
        return false;
}

/** 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() {
        int days = 0;
        for(int i= 1; i < month; i++){
                days += daysInMonth(i, year);
        }
        days += day;
        return days;
}

/** 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(DateS d) {
                int years = year - d.year, days = 0;
                days += years * 365;
                days += dayInYear() - d.dayInYear();
                return days;
        }

        public static void main(String[] args) {
        }
}


package week4;

public class Primes {
        public static boolean isPrime(int x){
                for(int i = 2; i <= Math.sqrt(x); i++){
                        if(x % i == 0) return false;
                }return true;
        }
        public static void printPrimes(int x){
                int counter = 0;
                int i = 2;
                while(counter < x){
                        if(isPrime(i)){
                                System.out.println(i);
                                counter++;
                        }
                        i++;
                }
        }

        public static void main(String[] args) {
                printPrimes(10);
        }
}
```