Homework 3. Exercises on Java Basics

Writing Good Programs

The only way to learn programming is program, program and program. Learning programming is like learning cycling, swimming or any other sports. You can't learn by watching or reading books. Start to program immediately. On the other hands, to improve your programming, you need to read many books and study how the masters program.
It is easy to write programs that work. It is much harder to write programs that not only work but also easy to maintain and understood by others – I call these good programs. In the real world, writing program is not meaningful. You have to write good programs, so that others can understand and maintain your programs.
Pay particular attention to:

1. **Coding style**:
   - Read Java code convention: "Google Java Style Guide".
   - Follow the Java Naming Conventions for variables, methods, and classes STRICTLY. Use CamelCase for names. Variable and method names begin with lowercase, while class names begin with uppercase. Use nouns for variables (e.g., radius) and class names (e.g., Circle). Use verbs for methods (e.g., getArea(), isEmpty()).
   - **Use Meaningful Names**: Do not use names like a, b, c, d, x, x1, x2, and x1688 - they are meaningless. Avoid single-alphabet names like i, j, k. They are easy to type, but usually meaningless. Use single-alphabet names only when their meaning is clear, e.g., x, y, z for co-ordinates and i for array index. Use meaningful names like row and col (instead of x and y, i and j, x1 and x2), numStudents (not n), maxGrade, size (not n), and upperbound (not n again). Differentiate between singular and plural nouns (e.g., use books for an array of books, and book for each item).
   - Use consistent indentation and coding style. Many IDEs (such as Eclipse / NetBeans) can re-format your source codes with a single click.
2. **Program Documentation**: Comment! Comment! and more Comment to explain your code to other people and to yourself three days later.
3. The only way to learn programming is program, program and program on challenging problems. The problems in this tutorial are certainly NOT challenging. There are tens of thousands of challenging problems available – used in training for various programming contests (such as International Collegiate Programming Contest (ICPC), International Olympiad in Informatics (IOI)).

1 Exercises on Method

1.1 DateUtility

Complete the following methods in a class called DateUtility:

- boolean isLeapYear(int year): returns true if the given year is a leap year. A year is a leap year if it is divisible by 4 but not by 100, or it is divisible by 400.
- boolean isValidDate(int year, int month, int day): returns true if the given year, month and day constitute a given date. Assume that year is between 1 and 9999, month is between 1 (Jan) to 12 (Dec) and day shall be between 1 and 28|29|30|31 depending on the month and whether it is a leap year.
- int getDayOfWeek(int year, int month, int day): returns the day of the week, where 0 for SUN, 1 for MON, ..., 6 for SAT, for the given date. Assume that the date is valid.
- String toString(int year, int month, int day): prints the given date in the format "xxxday d mmm yyyy", e.g., "Tuesday 14 Feb 2012". Assume that the given date is valid.

Hints

To find the day of the week (Reference: Wiki "Determination of the day of the week"):

1. Based on the first two digit of the year, get the number from the following "century" table.

1700-	1800-	1900-	2000-	2100-	2200-	2300-	2400-
4	2	0	6	4	2	0	6

   Take note that the entries $4, 2, 0, 6$ repeat.

2. Add to the last two digit of the year.
3. Add to "the last two digit of the year divide by 4, truncate the fractional part".
4. Add to the number obtained from the following month table:

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Non-Leap Year	0	3	3	6	1	4	6	2	5	0	3	5
Leap Year	6	2	3	6	1	4	6	2	5	0	3	5

5. Add to the day.
6. The sum modulus 7 gives the day of the week, where 0 for SUN, 1 for MON, ..., 6 for SAT.

For example: 2012, Feb, 17

EX

```
(6 + 12 + 12/4 + 2 + 17) % 7 = 5 (Fri)
```

The skeleton of the program is as follows:

```
/**
 * Utilities for Date Manipulation
 */
public class DateUtility {

   // Month's name - for printing
   public static String[] strMonths
     = {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
        "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

   // Number of days in each month (for non-leap years)
   public static int[] daysInMonths
     = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

   // Returns true if the given year is a leap year
   public static boolean isLeapYear(int year) {
      ......
   }

   // Return true if the given year, month, day is a valid date
   // year: 1-9999
   // month: 1(Jan)-12(Dec)
   // day: 1-28|29|30|31. The last day depends on year and month
   public static boolean isValidDate(int year, int month, int day) {
      ......
   }

   // Return the day of the week, 0:Sun, 1:Mon, ..., 6:Sat
   public static int getDayOfWeek(int year, int month, int day) {
      ......
   }

   // Return String "xxxday d mmm yyyy" (e.g., Wednesday 29 Feb 2012)
   public static String printDate(int year, int month, int day) {
      ......
   }

   // Test Driver
   public static void main(String[] args) {
     System.out.println(isLeapYear(1900));  // false
     System.out.println(isLeapYear(2000));  // true
     System.out.println(isLeapYear(2011));  // false
     System.out.println(isLeapYear(2012));  // true
```

```
45      System.out.println(isValidDate(2012, 2, 29));  // true
        System.out.println(isValidDate(2011, 2, 29));  // false
47      System.out.println(isValidDate(2099, 12, 31)); // true
        System.out.println(isValidDate(2099, 12, 32)); // false
49
        System.out.println(getDayOfWeek(1982, 4, 24));  // 6:Sat
51      System.out.println(getDayOfWeek(2000, 1, 1));   // 6:Sat
        System.out.println(getDayOfWeek(2054, 6, 19));  // 5:Fri
53      System.out.println(getDayOfWeek(2012, 2, 17));  // 5:Fri

55      System.out.println(toString(2012, 2, 14)); // Tuesday 14 Feb 2012
    }
57 }
```

Notes

You can compare the day obtained with the Java's Calendar class as follows:

```
1 // Construct a Calendar instance with the given year, month and day
  Calendar cal = new GregorianCalendar(year, month - 1, day);  // month is
     ↪ 0-based
3 // Get the day of the week number: 1 (Sunday) to 7 (Saturday)
  int dayNumber = cal.get(Calendar.DAY_OF_WEEK);
5 String[] calendarDays = { "Sunday", "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday" };
7 // Print result
  System.out.println("It is " + calendarDays[dayNumber - 1]);
```

The calendar we used today is known as Gregorian calendar, which came into effect in October 15, 1582 in some countries and later in other countries. It replaces the Julian calendar. 10 days were removed from the calendar, i.e., October 4, 1582 (Julian) was followed by October 15, 1582 (Gregorian). The only difference between the Gregorian and the Julian calendar is the "leap-year rule". In Julian calendar, every four years is a leap year. In Gregorian calendar, a leap year is a year that is divisible by 4 but not divisible by 100, or it is divisible by 400, i.e., the Gregorian calendar omits century years which are not divisible by 400. Furthermore, Julian calendar considers the first day of the year as march 25th, instead of January 1st.

This above algorithm work for Gregorian dates only. It is difficult to modify the above algorithm to handle pre-Gregorian dates. A better algorithm is to find the number of days from a known date.

1.2 GradesStatistics

Write a program called GradesStatistics, which reads in n grades (of int between 0 and 100, inclusive) and displays the average, minimum, maximum, median and standard deviation. Display the floating-point values upto 2 decimal places. Your output shall look like:

```
Command window

Enter the number of students: 4
Enter the grade for student 1: 50
Enter the grade for student 2: 51
Enter the grade for student 3: 56
Enter the grade for student 4: 53
The grades are: [50, 51, 56, 53]
The average is: 52.50
The median is: 52.00
The minimum is: 50
The maximum is: 56
The standard deviation is: 2.29
```

The formula for calculating standard deviation is: $\sigma = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} x_i^2 - \mu^2}$, where μ is the mean.

Hints

```
public class GradesStatistics {
   public static int[] grades;  // Declare an int[], to be allocated later
   // This array is accessible by all the methods.

   public static void main(String[] args) {
      readGrades();  // Read and save the inputs in static int[] grades
      System.out.println("The grades are: ");
      print(grades);
      System.out.println("The average is " + average(grades));
      System.out.println("The median is " + median(grades));
      System.out.println("The minimum is " + min(grades));
      System.out.println("The maximum is " + max(grades));
      System.out.println("The standard deviation is " + stdDev(grades));
   }

   // Prompt user for the number of students and allocate the static "grades" array.
   // Then, prompt user for grade, check for valid grade, and store in "grades".
   public static void readGrades() {
      ......
   }

   // Print the given int array in the form of [x1, x2, x3,..., xn].
   public static void print(int[] array) {
      ......
```

```
25    }
27    // Return the average value of the given int[]
      public static double average(int[] array) {
29       ......
      }
31
      // Return the median value of the given int[]
33    // Median is the center element for odd-number array,
      // or average of the two center elements for even-number array.
35    // Use Arrays.sort(anArray) to sort anArray in place.
      public static double median(int[] array) {
37       ......
      }
39
      // Return the maximum value of the given int[]
41    public static int max(int[] array) {
        int max = array[0];   // Assume that max is the first element
43      // From second element, if the element is more than max, set the max to this element.
        ......
45    }
47    // Return the minimum value of the given int[]
      public static int min(int[] array) {
49       ......
      }
51
      // Return the standard deviation of the given int[]
53    public static double standardDeviation(int[] array) {
        ......
55    }
  }
```

Take note that besides readGrade() that relies on class variable grades, all the methods are self-contained general utilities that operate on any given array.

1.3 GradesHistogram

Write a program called GradesHistogram, which reads in n grades (as in the previous exercise), and displays the horizontal and vertical histograms. For example:

Command window

```
 0 -   9: ***
10 -  19: ***
20 -  29:
30 -  39:
40 -  49: *
50 -  59: *****
60 -  69:
70 -  79:
80 -  89: *
90 -100: **

 *
 *
 *       *                       *
 *       *                       *                       *
 *       *                       *                       *
0-9   10-19 20-29 30-39 40-49 50-59 60-69 70-79 80-89 90-100
```