Hands-on Experiment # 11: Worksheet

Section_____1_____ Date_____13 April 2020_____

No more than 3 students per one submission of this worksheet.

Student ID _____Name_____

Student ID _____Name_____

Student ID _____Name_____

Part A: Getting Familiar with The MyDate Class[1]

A class called MyDate, which models a date instance, is defined as shown in the class diagram.

```
MyDate
-year:int
-month:int
-day:int
-strMonths:String[] =
   {"Jan","Feb","Mar","Apr","May","Jun",
    "Jul","Aug","Sep","Oct","Nov","Dec"}
-strDays:String[] =
   {"Sunday","Monday","Tuesday","Wednesday",
    "Thursday","Friday","Saturday"}
-daysInMonths:int[] =
   {31,28,31,30,31,30,31,31,30,31,30,31}
+isLeapYear(year:int):boolean
+isValidDate(year:int,month:int,day:int):boolean
+getDayOfWeek(year:int,month:int,day:int):int
+MyDate(year:int,month:int,day:int)
+setDate(year:int,month:int, day:int):void
+getYear():int
+getMonth():int
+getDay():int
+setYear(year:int):void
+setMonth(month:int):void
+setDay(day:int):void
+toString():String
+nextDay():MyDate
+nextMonth():MyDate
+nextYear():MyDate
+previousDay():MyDate
+previousMonth():MyDate
+previousYear():MyDate
```

The MyDate class contains the following private instance variables:

- year (int): Between 1 to 9999.
- month (int): Between 1 (Jan) to 12 (Dec).
- day (int): Between 1 to 28|29|30|31, where the last day depends on the month and whether it is a leap year for Feb (28|29).

[1] This exercise is updated from "Java Programming Tutorial OOP Exercises" at http://www.ntu.edu.sg/home/ehchua/programming/java/J3f_OOPExercises.html

It also contains the following **private static variables** (drawn with underlined in the class diagram):

- strMonths (String[]), strDays (String[]), and dayInMonths (int[]): static variables, initialized as shown, which are used in the methods.

The MyDate class has the following **public static methods** (drawn with underlined in the class diagram):

- 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.
- isValidDate(int year, int month, int day): returns true if the given year, month, and day constitute a valid 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 on Feb.
- 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. This method is provided in "DayOfWeek.java"

The followings are descriptions of **some public methods**:

- toString(): returns a date string in the format "xxxday d mmm yyyy", e.g., "Tuesday 14 Feb 2012".
- next/previousMonth(): must start from the 1st day of that month!
- next/previousYear(): must start from Jan 1 of that year!

Part B: Questions about The MyDate Class

How many attributes and methods in the class?

6 attributes and 18 methods

Please give the code to create an object of "today" date

MyDate date = new MyDate(2020, 4, 13)

Please specify which of the followings are leap years?

- 2000, 2007, 2013, 2004, 2001, 2012

2000, 2004, 2012

Are "1/15/2013" and "1/12/10000" a valid date? If not, why?

Month exceeds 12 and year exceeds 9999

Use "DayOfWeek.java" to find out the day of week of "1/1/2014"?

Wednesday

What should the toString() method return if the input date is "1/1/2014"?

Wednesday 1 Jan 2014

Part C: Coding

Write the code for the MyDate class.

Use the following test statements to test the MyDate class:

```
MyDate d1 = new MyDate(2012, 2, 28);

System.out.println(d1);             // Tuesday 28 Feb 2012
System.out.println(d1.nextDay());   // Wednesday 29 Feb 2012
System.out.println(d1.nextMonth()); // Thursday 1 Mar 2012 – must be "Day 1st"
System.out.println(d1.nextYear());  // Tuesday 1 Jan 2013 – must be "Jan 1"

MyDate d2 = new MyDate(2012, 1, 2);
System.out.println(d2);                 // Monday 2 Jan 2012
System.out.println(d2.previousDay());   // Sunday 1 Jan 2012
System.out.println(d2.previousMonth()); // Thursday 1 Dec 2011
System.out.println(d2.previousYear());  // Saturday 1 Jan 2011

MyDate d3 = new MyDate(2012, 2, 29);
System.out.println(d3.previousYear());  // Saturday 1 Jan 2011

// MyDate d4 = new MyDate(2099, 11, 31); // Invalid year, month, or day!
// MyDate d5 = new MyDate(2011, 2, 29);  // Invalid year, month, or day!
```

Include the screenshots below.

```
Tuesday 28 Feb 2012
Wednesday 29 Feb 2012
Thursday 1 Mar 2012
Tuesday 1 Jan 2013
Monday 2 Jan 2012
Sunday 1 Jan 2012
Thursday 1 Dec 2011
Saturday 1 Jan 2011
Saturday 1 Jan 2011
Invalid year, month, or day!
Invalid year, month, or day!
```

List all your source code here.

```
public class MyDate {

    // Attributes
    private int year;
    private int month;
    private int day;
    private static final String[] strMonths = new String[] { "Jan", "Feb", "Mar", "Apr", "May",
"Jun", "Jul", "Aug",
            "Sep", "Oct", "Nov", "Dec" };
    private static final String[] strDays = new String[] { "Sunday", "Monday", "Tuesday",
"Wednesday", "Thursday",
            "Friday", "Saturday" };
    private static final int[] daysInMonths = new int[] { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
};

    // Methods
    public static boolean isLeapYear(int year) {
        return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    }

    public static boolean isValidDate(int year, int month, int day) {
        return (year >= 1 && year <= 9999) && (month >= 1 && month <= 12)
                && (day >= 1 && day <= ((month == 2 && MyDate.isLeapYear(year)) ? 29 :
MyDate.daysInMonths[month - 1]));
    }

    public static int getDayOfWeek(int year, int month, int day) {
        return DayOfWeek.getDayOfWeek(day, month, year);
    }

    public MyDate(int year, int month, int day) {
        this.setDate(year, month, day);
    }
```

```java
public void setDate(int year, int month, int day) {
    if (!MyDate.isValidDate(year, month, day))
        System.out.println("Invalid year, month, or day!");
    else {
        this.year = year;
        this.month = month;
        this.day = day;
    }
}

public int getYear() {
    return this.year;
}

public int getMonth() {
    return this.month;
}

public int getDay() {
    return this.day;
}

public void setYear(int year) throws IllegalArgumentException {
    if (!MyDate.isValidDate(year, this.month, this.day))
        throw new IllegalArgumentException("Invalid year!");
    this.year = year;
}

public void setMonth(int month) throws IllegalArgumentException {
    if (!MyDate.isValidDate(this.year, month, this.day))
        throw new IllegalArgumentException("Invalid month!");
    this.month = month;
}

public void setDay(int day) throws IllegalArgumentException {
    if (!MyDate.isValidDate(this.year, this.month, day))
```

```
        throw new IllegalArgumentException("Invalid day!");
    this.day = day;
  }

  public String toString() {
    return MyDate.strDays[MyDate.getDayOfWeek(this.year, this.month, this.day)] + " " +
getDay() + " "
          + MyDate.strMonths[getMonth() - 1] + " " + getYear();
  }

  public MyDate nextDay() {
    if (isValidDate(this.year, this.month, this.day + 1))
      return new MyDate(this.year, this.month, this.day + 1);
    return this.nextMonth();
  }

  public MyDate nextMonth() {
    if (isValidDate(this.year, this.month + 1, 1))
      return new MyDate(this.year, this.month + 1, 1);
    return this.nextYear();
  }

  public MyDate nextYear() {
    return new MyDate(this.year + 1, 1, 1);
  }

  public MyDate previousDay() {
    if (isValidDate(this.year, this.month, this.day - 1))
      return new MyDate(this.year, this.month, this.day - 1);
    else if (isValidDate(this.year, this.month - 1, 1)) {
      return new MyDate(this.year, this.month - 1,
            ((this.month - 1 == 2 && MyDate.isLeapYear(this.year)) ? 29 :
MyDate.daysInMonths[this.month - 2]));
    } else
      return new MyDate(this.year - 1, 12, 31);
  }
```

```
  public MyDate previousMonth() {
    if (isValidDate(this.year, this.month - 1, 1))
      return new MyDate(this.year, this.month - 1, 1);
    return new MyDate(this.year - 1, 12, 1);
  }

  public MyDate previousYear() {
    return new MyDate(this.year - 1, 1, 1);
  }
}
```

Submit this worksheet (by only one member of the group) via http://www.myCourseVille.com (Assignments > Hands-on Experiment # 11) **within the day after your lecture**.