ITCS 209 Object Oriented Programming	Name:	Lab	Challenge Bonus	Peer Bonus
	ID:			
	Sec:			

Lab03: Class, Objects, Methods

You are provided a source code of a program for managing *Date* named the `DateTester.java`. It contains `DateTester` class and `strLeapYear` class as a starter class (Do not modify this class !!!). Your task is to implement the `MyDate` class in MyDate.java, which is used by `DateTester.java`, with the following variables, constructors, and methods. Only submit MyDate.java to MyCourses.

Instance variable

Variable Name	Type	description
`year`	int	Value range between 1 to 9999
`month`	int	Value range between 1 to 12
`day`	int	Value 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).
`objectNumber`	int	The object number of the instance

Static Class variable

Variable Name	Type	Description
`objectCounter`	int	Initialized to be zero; Incremented when an instance object of the class MyDate is created.
`strMonths`	String[]	An array of strings for the list of 12 month names ("January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December").

Constructors

Constructor Name and Parameters	Description
`MyDate``()`	• Set variables *year*, *month*, and *day* to be 1900, 1, and 1 respectively; • Increments the variable *objectCounter*; • Set the variable *objectNumber* to be *objectCouter*.
`MyDate``(int aYear, int aMonth, int aDay)`	• Set variables year, month, and day to be `aYear`, `aMonth`, and `aDay` respectively; • Increments objectCounter; • Sets objectNumber to be objectCouter.

Instance Methods

Method Name and Parameters	Description
`int` **`getObjectNumber`**`()`	• Returns the variable objectNumber.
`void` **`setDate`** `(int aYear, int aMonth, int aDay)`	• Sets the variables *year*, *month*, and *day* to be `aYear`, `aMonth`, and `aDay` respectively.
`void` **`setYear`**`(int aYear)`	• Sets *year* to be `aYear`.
`void` **`setMonth`**`(int aMonth)`	• Sets *month* to be `aMonth`.
`void` **`setDay`**`(int aDay)`	• Sets *day* to be `aDay`.
`int` **`getYear`**`()`	• Returns *year*.
`int` **`getMonth`**`()`	• Returns *month*.
`int` **`getDay`**`()`	• Returns *day*.
`String` **`toString`**`()`	• Returns the date string in the format "DD Month YYYY", e.g., "5 February 2016".

	Hint: Use strMonths and month for the index.
`MyDate` **`nextDay`**`()`	• Advance the date (day, month, and year) of the current object by one day. • returns **the same object** (i.e. return this;). Be careful about "31 December" (See algorithm).
`MyDate` **`nextMonth`**`()`	• Advance the date (day, month, and year) of the current object by one month and returns the same object. Be careful about "December".
`MyDate` **`nextYear`**`()`	• Advance the date (day, month, and year) of the current object by one year and returns the same object. Be careful the case Feb 29 going to the next year with Feb 29 (should become day 28).
`MyDate` **`previousDay`**`()`	• Reverse the date (day, month, and year) of the current object by one day and returns the same object. Be careful about "1 January" (See algorithm).
`MyDate` **`previousMonth`**`()`	• Reverse the date (day, month, and year) of the current object by one month and returns the same object. Be careful about "January".
`MyDate` **`previousYear`**`()`	• Reverse the date (day, month, and year) of the current object by one year and returns the same object. Be careful the case Feb 29 going to the previous year with Feb 29 (should become day 28).

Static Method

Method Name and Parameters	Description
`boolean` **`isLeapYear`**`(int year)`	• Check if the year is a leap year. A year is a leap year if its February has 29 days (See leap year algorithm below).

Note :

Java's array declaration example:

```
int[] myList = new int[10]; //10 is the size of the array myList
```

Java's array initialization example:

```
int[] myList = {12, 98, 34, 56, 72}; //The size of this array is 5
```

Java's array element access example (The same as in C language):

```
int a = myList[0];//0 is the index of the element being accessed
myList[1] = 35;
```

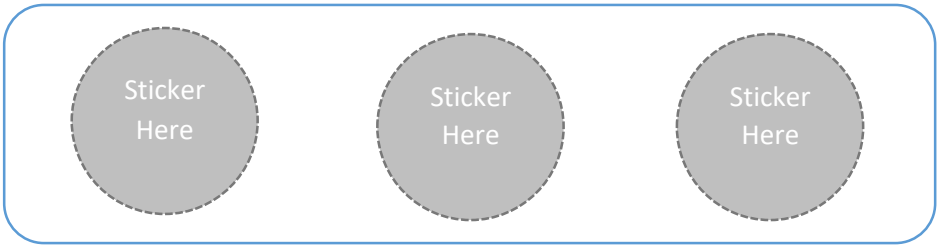
**** Refer to the FULL version of this lab (on MyCourses) for useful algorithms and expected output.**

Challenge Bonus (Optional):

Finish and submit this lab by **4:30PM**!!

* This week's challenge cannot be submitted next week. *

Peer Bonus (Optional):



ALGORITHM (`isLeapYear`)

```
boolean isLeapYear(int year):
1. If year is not divisible by 4 Then
   1.1. Return false (not a leap year)
   Else If year is not divisible by 100 Then
   1.2. Return true (a leap year)
   Else If year is not divisible by 400 Then
   1.3. Return false (not a leap year)
   Else
   1.4. Return true (a leap year)
```

ALGORITHM (`nextDay`)

```
MyDate nextDay():
1. If month = 12 AND day = 31 Then
   1.1. year <- year + 1
   1.2. month <- 1
   1.3. day <- 1
   Else
   1.4. If month = 4 OR 6 OR 9 OR 11 Then
       1.4.1. If day = 30 Then
                1.4.1.1. month <- month + 1
                1.4.1.2. day <- 1
              Else
                1.4.1.3. day <- day + 1
        Else If month ≠ 2 Then
       1.4.2. If day = 31 Then
                1.4.2.1. month <- month + 1
                1.4.2.2. day <- 1
             Else
                1.4.2.3. day <- day + 1
        Else
       1.4.3. If year is leap year AND day = 29 Then
                1.4.3.1. month <- month + 1
                1.4.3.2. day <- 1
             Else If year is not leap year AND day = 28 Then
                1.4.3.3. month <- month + 1
                1.4.3.4. day <- 1
             Else
                1.4.3.5. day <- day + 1
2. Return current object
```

ALGORITHM (`previousDay`)

```
MyDate previousDay():
1. If month = 1 AND day = 1 Then
   1.1. year <- year - 1
   1.2. month <- 12
   1.3. day <- 31
   Else
   1.4. If month = 5 OR 7 OR 10 OR 12 Then
        1.4.1. If day = 1 Then
                 1.4.1.1. month <- month - 1
                 1.4.1.2. day <- 30
               Else
                 1.4.1.3. day <- day - 1
         Else If month ≠ 3 Then
        1.4.2. If day = 1 Then
                 1.4.2.1. month <- month - 1
                 1.4.2.2. day <- 31
               Else
                 1.4.2.3. day <- day - 1
         Else
        1.4.3. If year is leap year AND day = 1 Then
                 1.4.3.1. month <- month - 1
                 1.4.3.2. day <- 29
               Else If day = 1 Then
                 1.4.3.3. month <- month - 1
                 1.4.3.4. day <- 28
               Else
                 1.4.3.5. day <- day - 1
2. Return current object
```

Expected Output

```
Object Number (a): 1
a's Date: 1 January 1900
a's Date: 31 December 1899
a's Date: 1 January 1900
a's Date: 1 December 1899
a's Date: 1 January 1900
a's Date: 13 April 2000
a's year is 2000, which is a leap year.

Object Number (b): 2
b's Date: 28 February 2016
b's Date: 29 February 2016
b's Date: 1 March 2016
b's Date: 1 March 2017
b's Date: 1 April 2017
b's Date: 1 April 2016
b's year is 2016, which is a leap year.

Object Number (c): 3
c's Date: 2 March 2017
c's Date: 1 March 2017
c's Date: 28 February 2017
c's Date: 28 February 2016
c's Date: 29 February 2016
c's Date: 28 February 2015
c's year is 2015, which is not a leap year.
```