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

Lab05: Class, Objects, Methods

You are provided a source code of a program for managing *Date* named the `DateTester.java`. It contains `DateTester` class with `strLeapYear` and `main` static methods (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 Fields/Variables

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 Variables

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 instance fields *year*, *month*, and *day* to be 1900, 1, and 1 respectively; • Increments the static variable *objectCounter*; • Set the variable *objectNumber* to be *objectCouter*.
`MyDate``(int aYear, int aMonth, int aDay)`	• Set instance fields *year*, *month*, and *day* to be `aYear`, `aMonth`, and `aDay` respectively; • Increments the static variable *objectCounter*; • Sets the variable *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., "05 February 2016". **Hint**: Use strMonths array and month value 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 Feb 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 Feb 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).

Note:

Java's array declaration example:

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

Java's array initialization example:

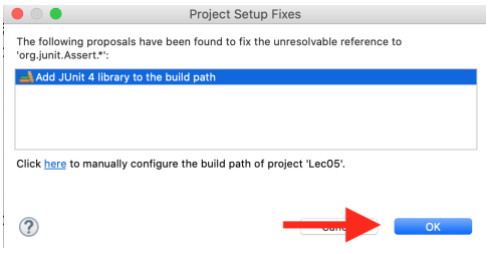
```
int[] myList = {12, 98, 34, 56, 72}; //The size of this array is 5
```

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;     // assign value 35 to index 1
```

Unit Test (Optional):

To help you test your program, the unit test using JUnit 4 library is provided. You must install JUnit library in Eclipse before using this unit test. In the TestCase.java file, follow the quick fix and select "Fix project setup" -> select "Add JUnit 4 library to the build path" -> Click "OK"

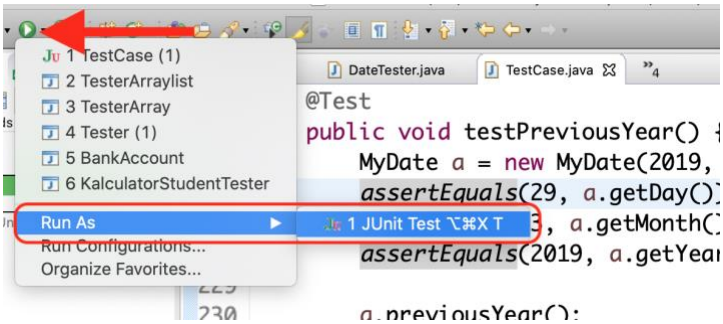


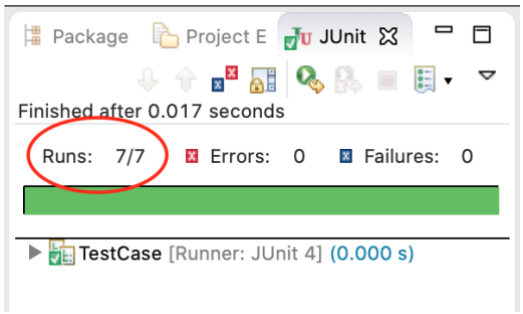
Expected Output from DateTester.java

```
Object Number (a): 1
a's Date: 01 Jan 1900
a's Date: 31 Dec 1899
a's Date: 01 Jan 1900
a's Date: 01 Dec 1899
a's Date: 01 Jan 1900
a's Date: 13 Apr 2000
a's year is 2000, which is a leap year.

Object Number (b): 2
b's Date: 28 Feb 2020
b's Date: 29 Feb 2020
b's Date: 01 Mar 2020
b's Date: 01 Mar 2021
b's Date: 01 Apr 2021
b's Date: 01 Apr 2020
b's year is 2020, which is a leap year.

Object Number (c): 3
c's Date: 02 Mar 2021
c's Date: 01 Mar 2021
c's Date: 28 Feb 2021
c's Date: 28 Feb 2020
c's Date: 29 Feb 2020
c's Date: 28 Feb 2019
c's year is 2019, which is not a leap year.
```

Expected Output from TestCase.java (Optional)

Run TestCase.java as -> JUnit Test. You will see the JUnit tab on the left pane, with number of pass, error, and failure methods. You should have 7/7 pass methods as shown here.





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
```

Challenge Bonus (Optional): [You may submit this task in the lab hour next week]

Your task is to create another static method in `MyDate` class

Method Name and Parameters	Description
`int` **`dateDiff`**`(MyDate a, MyDate b)`	• Return total number of days between two given dates. Be careful about leap year which has 366 days in one year. • You may assume that the MyDate *a* is always come before MyDate *b*

Beside `dateDiff` method, you are allowed to create any additional methods as you wish to complete the task. In DateTester.java, you have to uncomment some code inside the `main` method, as well as the `challenge(MyDate begin, MyDate end)` method to run this challenge. The expected result is shown here

```
--- CHALLENGE ---
Begin date: 01 Jan 2020
End date: 01 Jan 2021
Total number of days between two dates is 366.

Begin date: 01 Jan 2020
End date: 03 Feb 2021
Total number of days between two dates is 399.

Begin date: 01 Dec 2020
End date: 03 Feb 2021
Total number of days between two dates is 64.
```