Lab Exercise (Chapter 8: Part 1)

Description of the Problem

Create class **Date** with the following capabilities:

a) Output the date in multiple formats, such as
   DD/MM/YYYY (e.g. 27/04/2012)
   MMM DD, YYYY (e.g. April 27, 2012)
   DDD YYYY (e.g. 118 2012)

b) Use overloaded constructors to create **Date** objects, which you can initialize with dates of either formats in part(a), where:
   1. First option, the constructor should receive three integer values.
   2. Second option, the constructor should receive a String and two integer values.
   3. Third option, the constructor should receive two integer values, the first of which represents the day number in the year.

Then create a **DateTest** application with the 'main' method and a 'getMenuChoice' method. The logic in the application should be:

1. Ask the user to choose the choice of date format that he/she want to input by calling the 'getMenuChoice' method.
2. The 'getMenuChoice' method will ask user to choose number 1, 2 or 3 for 3 choices of date format, and number 4 to exit, return the choice that user has chosen.
3. You should repeat your application unless user chooses choice 4 – to exit.
4. If user choose choice 1, ask user to key in day, month and year
5. If user choose choice 2, ask user to key in MonthName, day and year
6. If user choose choice 3, ask user to key in number of days and year
7. Based on the choices, create the Date object by calling the respective Date constructor. (you can use the switch control statements)
8. Then, from the Date object created, call the 'toString', 'toMonthNameDateString', and 'toDayDateString' method to display the date in three different formats.

Try run the program, choose option 1 and key in day 0. Observe what happen?
Last, if you have not done so, enhance your code by introduce the throw IllegalAgumentException coding in the **Date** class for invalid day, month and year, and include the try..catch block in the **DateTest** program. Try to repeat step above and observe the output.

[Hint: To compare string, use the **equals** method. For example: s1.equals(s2)]

The Date class UML diagram is given as below:

```
Date
-------------------------------------------------------------------------------
-    day: int
-    month:int
-    year:int
-    monthNames[12]: String = {"January", "February","March", "April", "May", "June", "July", "August",
     "September", "October", "November", "December"}
-    monthDays[12]:int = { 31, 28, 31, 30, 31, 30,31, 31, 30, 31, 30, 31 }
-------------------------------------------------------------------------------
<<constructor>> Date()
<<constructor>> Date(dd:int, mm:int, yyyy:int)
<<constructor>> Date(mm:String, dd:int, yyyy:int)
<<constructor>> Date(ddd:int, yyyy:int)
+ setDay(dd:int)
+ setMonth(mm:int)
+ setYear(yyyy:int)
+ toString(): String
+ toMonthNameDateString(): String
+ toDayDateString(): String
- convertFromMonthName(monthName:String)
- daysInMonth():int
- leapYear(): Boolean
- convertFromDayOfYear(ddd:int)
- convertToDayOfYear():int
```

Note: The monthNames and monthDays is constant
Method description:

- **<<constructor>> Date():**
  Set the default value for month and day to 1, and year to 2012
- **<<constructor>> Date(dd:int, mm:int, yyyy:int):**
  Set the month, day, year value
- **<<constructor>> Date(mm:String, dd:int, yyyy:int)**
  Set the day and year value, and call 'convertFromMonthName' method to set the month value based on the month by string
- **<<constructor>> Date(ddd:int, yyyy:int)**
  Set the year value, and call 'convertFromDayOfYear' method to set the day and month value based on the total day stated.
- **+ setDay(dd:int)**
  Set the day value if day is not negative and is not exceed the total day for that particular month by calling method "daysInMonth"
- **+ setMonth(mm:int)**
  Set the month value if month is it's more than 0 and less than or equal to 12
- **+ setYear(yyyy:int)**
  Set the year value if year is more than or equal to 1900 and less than or equal to 2100
- **+ toString(): String**
  Return date in format: dd/mm/yyyy (tips: you can using String.format method to format the string)
- **+ toMonthNameDateString(): String**
  Return date in format: MonthName dd, yyyy
- **+ toDayDateString(): String**
  Return date in format DDD yyyy
  Call the 'convertToDayOfYear' method to get the total days
- **- convertFromMonthName(monthName:String)**
  Convert from month name to month number. Any invalid month default it to 1.
- **- daysInMonth():int**
  Return the number of days in the month by calling method 'leapYear' to check if it's a leap year, if yes then return 29, else return the days as per in the monthDays array
- **- leapYear(): Boolean**
  Test for a leap year, logic is given as below:
  ```
  if (year % 400 == 0 || (year % 4 == 0 && year % 100 != 0))
          return true;
  else
          return false;
  ```
- **- convertFromDayOfYear(ddd:int)**
  Sets the day and month to the proper values based on ddd. If ddd must be in the range of 1-365, else set the ddd to 1.
- **- convertToDayOfYear():int**
  Convert mm and dd to ddd. This function will call method 'daysInMonth' to get the total days for calculation purpose.

Sample Output:

```
Enter 1 for format (DD/MM/YYYY)
Enter 2 for format (MonthName DD, YYYY)
Enter 3 for format (DDD YYYY)
Enter 4 to exit
Pick your choice:
```

If user choose choice 1

```
Enter Day of Month : 27
Enter Month (1-12): 4
Enter Year: 2012
```

If user choose choice 2

```
Enter Month Name: April
Enter Day of Month: 27
Enter Year: 2012
```

If user choose choice 3

```
Enter Day of Year: 118
Enter Year: 2012
```

For any choices user made, after user input the values, the results of all the formats will be printed.

For example as below:

```
27/04/2012
April 27, 2012
118 2012
```