CSC 122 001 Computer Science II
Julius Ranoa

Chapter 11 Programming Challenge 2 Day of the Year
Assuming that a year has 365 days, write a class that takes an integer representing a day of the year and translates it to a string consisting of the month followed by day of the month.

Screenshot of runtime:

```
2 January 2
32 February 1
365 December 31
```

Files: (1) main.cpp, (2) DayOfYear.h, (3) DayOfYear.cpp

main.cpp

```
#include <iostream>
#include "DayOfYear.h"

int main() {

    DayOfYear d;
    int test[ ] = {2, 32, 365};
    int SIZE = sizeof(test) / sizeof(test[0]);
    for (int i = 0; i < SIZE; i++) {
        d.setDay(test[i]);
        std::cout << test[i] << " ";
        d.print();
        std::cout << "\n";
    }

    return 0;
}
```

DayOfYear.h

```
#ifndef CH11_PR2_DAY_OF_THE_YEAR_DAYOFYEAR_H
#define CH11_PR2_DAY_OF_THE_YEAR_DAYOFYEAR_H

#include <string>

class DayOfYear {

public:
    static const int dayMax;
    static const int numMonths;
    static int daysPerMonth[ ]; // One-based indexing. 1 = January
    static std::string monthNames[ ];

private:
    int numDay; // -nth day of the year

    // Results
    int numMonth;
    std::string month;
    int dayOfMonth;

    void extractDetails();

public:
    DayOfYear();
    DayOfYear(int);

    void setDay(const int);
    bool isInRange(int);
    void print();

};


#endif //CH11_PR2_DAY_OF_THE_YEAR_DAYOFYEAR_H
```

DayOfYear.cpp

```cpp
#include <iostream>
#include "DayOfYear.h"

const int DayOfYear::dayMax = 365;
const int DayOfYear::numMonths = 12;

// One-based indexing. January = 1.
// Assuming no leap years.
int DayOfYear::daysPerMonth[DayOfYear::numMonths + 1] = {
    0,
    31, 28, 31, 30, 31, 30,
    31, 31, 30, 31, 30, 31
};

std::string DayOfYear::monthNames[DayOfYear::numMonths + 1] = {
    "",
    "January", "February", "March",
    "April", "May", "June",
    "July", "August", "September",
    "October", "November", "December"
};

// Constructors

DayOfYear::DayOfYear(int num) {
    if (!isInRange(num)) {
        std::cout << "Day not in range.";
        exit(-1);
    }
    numDay = num;
    extractDetails();
}

// Constructor delegated
DayOfYear::DayOfYear() : DayOfYear(1) {}

// PRIVATE METHODS

void DayOfYear::extractDetails() {
    int temp = numDay;
    for (int i = 1; i <= numMonths; i++) {
        temp = temp - daysPerMonth[i];
        if (temp <= 0) {
            numMonth = i;
            month = monthNames[i];
            dayOfMonth = temp + daysPerMonth[i];
            break;
        }
    }
}

// PUBLIC METHODS

bool DayOfYear::isInRange(int val) {
```

```
    if (val > 0 && val <= dayMax) {
        return true;
    } else return false;
}

void DayOfYear::setDay(const int val) {
    if (!isInRange(val)) {
        std::cout << "Day not in range.";
        exit(-1);
    }
    numDay = val;
    extractDetails();
}

void DayOfYear::print() {
    std::cout << month << " " << dayOfMonth;
}
```