Exercise: Encapsulation

Add a private member function that calculates the number of days in a month, and use it to update the class invariants. Be sure to account for leap years! (https://en.wikipedia.org/wiki/Leap_year#Algorithm)

In [ ]:

```
#include <cassert>

class Date {
public:
  Date(int day, int month, int year);
  int Day() const { return day_; }
  void Day(int day);
  int Month() const { return month_; }
  void Month(int month);
  int Year() const { return year_; }
  void Year(int year);

private:
  bool LeapYear(int year) const;
  int DaysInMonth(int month, int year) const;
  int day_{1};
  int month_{1};
  int year_{0};
};

Date::Date(int day, int month, int year) {
  Year(year);
  Month(month);
  Day(day);
}

bool Date::LeapYear(int year) const {
    if(year % 4 != 0)
        return false;
    else if(year % 100 != 0)
        return true;
    else if(year % 400 != 0)
        return false;
    else
        return true;
}

int Date::DaysInMonth(int month, int year) const {
    if(month == 2)
        return LeapYear(year) ? 29 : 28;
    else if(month == 4 || month == 6 || month == 9 || month == 11)
        return 30;
    else
        return 31;
}

void Date::Day(int day) {
  if (day >= 1 && day <= DaysInMonth(Month(), Year()))
    day_ = day;
}

void Date::Month(int month) {
  if (month >= 1 && month <= 12)
    month_ = month;
}

void Date::Year(int year) { year_ = year; }

// Test
int main() {
  Date date(29, 2, 2016);
  assert(date.Day() == 29);
  assert(date.Month() == 2);
  assert(date.Year() == 2016);

  Date date2(29, 2, 2019);
  assert(date2.Day() != 29);
  assert(date2.Month() == 2);
  assert(date2.Year() == 2019);
}
```

Compile & Run | Explain

Loading terminal (id_gfhfid7), please wait...