Introductory Programming (02100, 02115, and 02199)
Mandatory assignment 1: day of week

The problem to be solved is to find the name of the day (*Monday, Tuesday,* etc. using the English names!), given a date in the form (year,month,day). You should write a Java program that implements your own algorithm (i.e. without use of any class library related to calendars) to provide an answer for any date since year 1.

Overview

The calendar to be used is the Gregorian calendar, i.e. the one we use today. Note that although the calendar was first adopted in 1582, you may allow the rules to be used for the dates before. Among other things the Gregorian calendar differs from it predecessor in its rules for definition of a leap year:

> *Year number N is a leap year, if N is divisible by four, except when N is also divisible by 100, in which case N is a leap year only if N is divisible by 400.*

So 1996, 2000, and 2004 are leap years, but 2100 is not.

> **Background information:** You might want to use the occasion to read a little about the motivation for the Gregorian calendar. Then you will learn when it was adopted in various countries, and perhaps also that there is some ambiguity about the birthday of George Washington. You might also enjoy the rules for determining Easter day (a Sunday), and how they relate to astronomy. Start with
>
> > L.E. Doggett: Calendars
> > http://astro.nmsu.edu/~lhuber/leaphist.html
>
> but you can find many other nice papers about calendars on the net. You may even find a program that can be used to check the results of the program you are requested to write, which is ok, but do not copy any program!

You should write a short report about your program, your reasoning behind it, and your testing it. The size and structure of the report should adhere to the guidelines in *Practical information* made available earlier. You should have both a functional and a structural test.

Your program will have to abide to rules determined by our intention to check some of it automatically. This corresponds, in fact, to the workplace situation where you might have to contribute to a big program and abide by the rules determined by the needs of other parts. The next section gives details.

HINT: the complexity of the program depends on an appropriate but simple analysis of the problem, so think carefully before you start to write the program.

Important details

The part that will be checked automatically is a class that must have the name `Gregor`. You will need to write another class, `GregorTest`, with a `main` method to test `Gregor`, i.e. with text in a file `GregorTest.java`.

The day-of-week names must be the English ones, with a capital letter first. Otherwise the comparisons performed by the automatic check will fail, and you will be blamed for an error (you may find this a nuisance, but it is quite realistic — just as failing to abide!)

The `Gregor` class must contain a method:

```
String dayOfWeek(int year, int month, int day);
  // Finds the name of date year/month/day, for year > 0
  // Raises Exception("Bad date") when year/month/day is invalid
```

Note that the text given when the method raises an exception must be `"Bad date"`.

In addition to handing in a printed report, you should submit your programs as explained below.

1. Make a directory containing

   (a) those .java files (e.g. Gregor.java, GregorTest.java) that constitute your solution to the mandatory assignment

   (b) the corresponding compiled .class files

   The directory must not contain anything else, and in particular not things that can offend the reciever. (In case of the latter, it will be treated as an attempt to cheat at an examination and be reported to the university management.)

2. Go to that directory and issue the command

   ```
   jar cf myJar.jar *
   ```

   This results in a file named "myJar.jar" containing all the files of the current directory.

3. Test use of `myJar.jar` by copying it to an empty directory and run `GregorTest` by the command

   ```
   java -classpath myJar.jar GregorTest
   ```

   If the output is what you expect, `myJar.jar` should be ready for submission.

4. Send an e-mail to classid@imm.dtu.dk, where classid is your class identifier (info1, info2, info3, andre2100, elektro1, elektro2, elektro3, andre2199 or c02115). The subject field should be "Mandatory assignment 1" and the mail should contain the jarfile as attachment. Inside your mail you should write the names and study numbers of the two group members.

If you have any questions in relation to this, please consult your teaching assistant or the course secretary (c02100@imm.dtu.dk, c02199@imm.dtu.dk).

Evaluation criteria

The reports will be assessed first by the teaching assistants and later by the course staff. The teaching assistants will be given a form to complete for each report to indicate their evaluation of it. They are asked to assess the report in a stereotyped way expressed as statements below. Free-text remarks may be added.

Report

- Impression based on formulation, spelling, and grammar
  - is good
  - is ok
  - is not favourable
- The required sections
  - are all present
  - are not all present
- The summary section
  - is very good
  - is good
  - is average
  - is below average
  - is insufficient
  - is missing
- The analysis section
  - is very good
  - is good
  - is average
  - is below average
  - is insufficient
  - is missing

- The test section
  - is very good
  - is good
  - is average
  - is below average
  - is insufficient
  - is missing

Technical contents: program design

- The overall impression
  - is very good
  - is good
  - is average
  - is below average
  - is bad

Technical contents: testing

- Your test design
  - is fine
  - should cover better
  - is missing
- Your test results
  - are very good
  - display most errors
  - display too few errors

Program text

- Identification of authors
  - is satisfactory
  - is missing
- Comments about classes
  - are very good
  - are good
  - are average
  - are below average
  - are totally missing

  ... but not all get a comment
- Comments about methods
  - are very good
  - are good
  - are average
  - are below average
  - are totally missing

  ... but not all get a comment
- Comments about variables
  - are very good
  - are good
  - are average
  - are below average
  - are totally missing

  ... but not all get a comment
- Level of nesting and indentation
  - are proportional
  - are partly proportional
  - are not proportional