Hand in assignment 2 for Programming 2IS45 (fall 2012)

For this assignment you should hand in two java-files `Date.java` and `Date_alt.java` via the PEACH-system (https://peach.win.tue.nl/) before Sunday 14th October 23.00h. Your solutions should be in the style and form of the example projects presented during the lectures (these projects can be found on the website of the course http://www.win.tue.nl/~wsinswan/programmeren_2IS45/).

We consider the design of a class `Date` which represents and maintains a calendar date. It should have the following operations which are described informally:

- Creation and initialization
- Queries/Inspection
  - current date in integers representing year, month, day
  - current date in integers representing year and daynumber in year
  - current date in string in ISO-format: yyyy-mm-dd; e.g. 2008-08-28
- Commands/Modification
  - reset to initial date
  - set to given year, month, day
  - set to given year and daynumber in year
  - advance date by a number of days
  - next, i.e. advance date by one day

This informal description leads to a class interface of the following form, where the operations have been grouped according to their main role, i.e.

- construction and initialization
- queries (“observable entities”), i.e. inspection of the current state
- validators, used to express and verify preconditions
- commands, i.e. changes of the current state

```java
public class Date {
    // constructor
    public Date() {...}
    // queries
    public int year() {...}
    public int month() {...}
    public int day() {...}
    public int dayNumber() {...}
    public int lengthOfMonth(int aYear, int aMonth) {...}
    public int lengthOfYear(int aYear) {...}
    public String isoString() {...}
    // validators
    public boolean validYear(int aYear) {...}
    public boolean validMonth(int aMonth) {...}
    public boolean validDay(int aYear, int aMonth, int aDay) {...}
    public boolean validDayNumber(int aYear, int aDayNumber) {...}
    // commands
    public void reset() {...}
    public void setYMD(int aYear, int aMonth, int aDay) {...}
    public void setYearDayNumber(int aYear, int aDayNumber) {...}
    public void advance(int aNumberOfDays) {...}
    public void next() {...}
}
```
The functions `year`, `month`, and `day` are primitive queries with obvious meanings.

The function `dayNumber` expresses the daynumber of the current date in year `year()`.

The function `isoString` returns the current date as a string in ISO-format: `yyyy-mm-dd`; e.g. 2008-08-28.

The other functions `lengthOfMonth`, `lengthOfYear`, and `isLeapYear` have an obvious result and are added as auxiliary functions within the class.

These functions, which provide different pieces of information about the current date, are not independent. Their exact relationships will be given later on.

The method `reset` resets the current date to the initial date.

The method `setYMD` sets the current date to given years, months and days.

The method `setYearDayNumber` sets the current date to given year and daynumber.

The method `advance` advances the current date with a given number of days.

The method `next` advances the current date by 1 day.

The informal requirements leave open what date should be taken as the initial date. We shall take as initial date January 1, 1583, the first day of the first full year according to the Gregorian calendar. (Note: we have implicitly assumed that we are dealing with dates according to the Gregorian calendar. Large parts of the world are using different calendars, e.g. the Islamic, Persian, Hebrew, Hindu, and Chinese calendars. For an impressive overview see e.g. the Wikipedia entry on calendars: http://en.wikipedia.org/wiki/Calendar).

The precise effect of these procedures on the observable entities is given in the listing of file `Date.java` at the end of this document (the file can be downloaded from the website). The listing provides specifications of all methods but not the corresponding implementations. Furthermore, it proposes to internally represent the date with two instance variables `fYear` and `fDayNumber` giving the year and the number of the day within the year.

**Assignments**

1. Give the implementations of the methods in `Date.java` using the proposed internal representation. In your implementation you are not allowed to use any class from the standard libraries or other libraries. Hand in the adapted file. (Before handing in the file be sure to test your implementation in a number of cases.)

2. Give an implementation of a class `Date_alt` that has the same interface and the same contracts as the class `Date` but uses an internal representation consisting of a triple of instance variables `fYear`, `fMonth`, and `fDay` representing the date by the number of the year, the number of the month within that year, and the number of the day within that month. What are the (internal/representation) invariants in this case? Give the implementations of all methods using the changed internal representation. Hand in this other implementation in a file `Date_alt.java`.

---

```java
public class Date {

private int fYear; // internal representation by
private int fDayNumber; // the number of the year fYear and

// invariants
// Pri0: validYear(fYear)
// Pri1: validDayNumber(fYear, fDayNumber)

// Date constants dcXXX
int [] dcLengthOfMonth = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

// constructor
```
// post: year() == 1583, month() == 1, day() == 1
public Date() {
    // TODO implement actual body
}

// queries
// return: the year number of the date represented by this
public int year() {
    // TODO implement actual body
}

// return: the month number of the date represented by this
public int month() {
    // TODO implement actual body
}

// return: the day number within the month of the date represented by this
public int day() {
    // TODO implement actual body
}

// return: the day number within the year of the date represented by this
public int dayNumber() {
    // TODO implement actual body
}

// pre: validYear(aYear), validMonth(aMonth)
// ret: length in days of month aMonth in year aYear (according to Gregorian calendar)
public int lengthOfMonth( int aYear, int aMonth ) {
    // TODO check precondition with assert statement
    // TODO implement actual body
}

// pre: validYear(aYear)
// ret: length in days of year aYear (according to Gregorian calendar)
public int lengthOfYear( int aYear ) {
    // TODO check precondition with assert statement
    // TODO implement actual body
}

// pre: validYear(aYear)
// ret: whether year aYear is a leap year (according to Gregorian calendar)
public boolean isLeapYear( int aYear ) {
    // TODO check precondition with assert statement
    // TODO implement actual body
}

// ret: current date in ISO-format yyyy-mm-dd, e.g. 2008-01-28
public String isoString() {
    // TODO implement actual body
}

// validators
// ret: 1583 <= AYear
public boolean validYear( int aYear ) {
    // TODO implement actual body
}

// ret: 1 <= AMonth <= 12
public boolean validMonth( int aMonth ) {
    // TODO implement actual body
}

// pre: validYear(aYear), validMonth(aMonth)
public boolean validDay( int aYear, int aMonth, int aDay ) {
    // TODO implement actual body
}

public boolean validDayNumber( int aYear, int aDayNumber ) {
    // TODO implement actual body
}

// invariants
// I0: validYear( year() )
// I1: validMonth( month() )
// I2: validDay( year() , month() , day() )
// I3: validDayNumber( year() , dayNumber() )
// I4: dayNumber() = (Sum m. 1 <= m < month(): lengthOfMonth(year(), m)) + day()

// commands
public void reset() {
    // TODO implement actual body
}

public void setYMD( int aYear, int aMonth, int aDay ) {
    // TODO check precondition with assert statement
    // TODO implement actual body
}

public void setYearDayNumber( int aYear, int aDayNumber ) {
    // TODO check precondition with assert statement
    // TODO implement actual body
}

public void advance( int aNumberOfDays ) {
    // TODO check precondition with assert statement
    // TODO implement actual body
}

public void next() {
    // TODO implement actual body
}