Vorgehensweise und Entscheidungsfindung

Zunächst habe ich im Internet nach einem Algorithmus zur Berechnung des Datums von Ostern gesucht. Entschieden habe ich mich für den Algorithmus nach Butcher. Dieser Algorithmus ist recht einfach gehalten und man muss am Ende des Algorithmus nicht noch, wie bei Gauß, nach Erhalt des Datumswert weiter rechnen. Falls das Ergebnis >31 ist müsste man dann noch abfangen, dass Ostern dann in den April fallen würde usw.
Der Algorithmus nach Butcher ist ab 1583 gültig und behält seine Gültigkeit, solange die Kirche keine Veränderungen zum Termin von Ostern unternimmt.

Die Klasse „EasterCalc“ beherbergt alle Methoden zur Berechnung. Es gibt eine Methode „easterDateCalc“ zur Berechnung des Osterdatums für ein übergebenes Jahr. Da man mittels return nur den Inhalt eines Feldes zurückgeben kann, habe ich mich dazu entschieden, dass errechnete Datum in einem Objekt der Klasse „calendar“ zu speichern. Diese wird dann zur weiteren Berechnung zurückgegeben und enthält Tag, Monat und Jahr.
Die beiden Methoden „earlyEasterNext“ und „earlyEasterPrev“ berechnen, wann Ostern wieder auf selbigen Tag und Monat fallen, wie im vorher berechneten Jahr, bzw. wann Ostern das letzte Mal auf dieses Datum fiel. Die Methoden geben jeweils nur das berechnete Jahr zurück.

In der Klasse „EasterInputOutput „finden nur Eingaben und Ausgaben auf der Konsole statt. Die Methode „readYear“ liest einen Jahreszahl von der Konsole ein. Dabei wird geprüft, ob die Jahreszahl auch für den Gültigkeitsbereich des Algorithmus passend ist. Dazu muss gewährleistet sein, dass die Jahreszahl nicht kleiner als 1583 sein kann und auch nicht größer als 10.000. In der while-Schleife wird damit geprüft, dass die eingegeben Jahreszahl nur vierstellig sein kann und nicht kleiner als 1583 wird. Somit ist das Programm allerdings auch nicht für Ostern nach dem Jahre 9999 geeignet.
Diese Methode liefert das eingegeben Jahr zurück.

Die Methoden „earlyEasterNextOutput“ und „earlyEasterPrevOutput“ bekommen das von der Konsole eingelesene Jahr und das entsprechend neu berechnete Jahr für den letzten bzw. nächsten Ostertermin als Übergabeparameter und geben diese in einem passenden Satz auf der Konsole aus.

Die Methode „easterMenu“ erwartet eine Tastatureingabe des Benutzers, um das Programm entweder zu beenden oder den Programmablauf erneut zu starten.

Die beiden Klassen „EasterCalc“ und „EasterInputOutput“ enthalten keine globalen Attribute oder Konstruktoren. Da ich nur mit lokalen Variablen arbeite, die nur innerhalb ihrer Methode gültig sind, wurden keine Konstruktoren benötigt. So kann auch sichergestellt werden, dass bei Methodenaufruf nur mit gültigen Werten gerechnet wird.

Die main-Mathode befindet sich in der Klasse „EasterApp“.

In der Klasse „EasterApp“ befindet sich neben der main-Methode noch die Methode „runEasterApp“.
In der Methode „runEasterApp“ werden die einzelnen Rechenmethoden und Ein-/Ausgabemethoden zu einem lauffähigen Programm verknüpft. In der main-Methode werden

dann nur noch zwei neue Objekte erstellt, um einmal das Programm zu starten und nach dessen Ablauf das Menue aufzurufen.

Die Klasse „EasterCalcTest“ beinhaltet einige JUnit Tests, um die Rechenmethoden der Klasse „EasterCalc“ zu prüfen.

Frage: Welches ist der früheste mögliche Termin für Ostern, welches der spätest mögliche?

Der Frühling beginnt am 21. März. Da Ostern auf den Sonntag nach dem ersten Frühjahrsvollmond fällt, kann Ostern frühestens am 22. März nach Frühlingsbeginn statt finden oder spätestens am 25. April.
Zum Beispiel fiel 1693 Ostern auf den 22.03 und 1666 auf den 25.04.
Als mögliche Lösung in programmiertechnischer Sicht würde mir nur einfallen, Osterdaten für ein Intervall zu berechnen und dann zu sortieren, der niedrigste Wert wäre dann mein frühster und der höchste Wert mein spätester Ostertermin. Mit einem Quicksort oder Bubblesort wäre dies sicher eine mögliche Anwort, die Implementierung ist mir jedoch noch nicht gelungen…

Quellenangaben

Eine Tabelle mit fertigen Osterdaten zum Vergleichen:
Das Münchener Astro Archiv
Stand 16. Juni 1997
C. Kronberg
URL: http://www.maa.mhn.de/StarDate/feiertage.html
(abgerufen am 30.03.2010)

Algorithmus zur Berechnung des Osterdatums nach Butcher:
Marcos J. Montes
Stand 31. Juli 2001
URL: http://www.smart.net/~mmontes/nature1876.html
(abgerufen am 30.03.2010)

Kalender Klasse
Sun Microsystems, Inc.; Class Calendar
URL: http://java.sun.com/j2se/1.4.2/docs/api/java/util/Calendar.html
(abgerufen am 30.03.2010)

Klasse SimpleDateFormat
Sun Microsystems, Inc.: Class SimpleDateFormat
http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html
(abgerufen am 30.03.2010)

Ideenaustausch über die Konzeption fand statt mit folgenden Kommilitonen:
Marcel Ehrlitzer
Tristan Rudat