**CUSTOM CHINESE CALENDAR FOR LIB RECUR**

**1. Các class mới được thêm vào trong thư viện**

Trong library lib-recur tại thư mục CalendarMatrics sẽ bổ sung các class sau LunarCalendarUtils, ChineseCalendarMatrics, NumDayOfMonthResult vai trò của các chức năng của các class trên sẽ được mô tả chi tiết bên dưới.

**a. NumDayOfMonthResult**

\* Đây là một class biểu diễn thông tin trả về của khi tiến hành lấy số ngày của một tháng âm lịch. Class này gồm 2 thuộc tính đó là:

**int numDay** (số ngày của tháng)

**int**[] **firstDayOfMonth** (mảng lưu giữ thông tin của ngày đầu tiên tháng âm đó gồm các thông tin như ngày, tháng, năm và trạng thái tháng là có nhuận hay không)

**b. LunarCalendarUtils**

\* Đây là class cung cấp một số method hổ trợ việc xử lý đối với ngày âm. Trong đó có một số method quan trọng như:

+ **public static int** jdFromDate(**int** dd, **int** mm, **int** yy)

=> Chức năng: tính toán số ngày từ ngày 1/1/4713 TCN đến ngày được truyền vào trong tham số

- Đầu vào: là ngày, tháng và năm ở dạng số nguyên, trong đó ngày, tháng bắt đầu bằng 1

- Đầu ra: là số ngày tính từ ngày 1/1/4713 trước công nguyên

+ **public static int**[] jdToDate(**int** jd)

=> Chức năng: Chuyển đổi một số nguyên là số ngày tính từ ngày 1/1/4713 TCN thành một ngày dương lịch

- Đầu vào: là một số nguyên chính là số ngày tính từ ngày 1/1/4713 trước công nguyên

- Đầu ra: là một mảng số nguyên trong đó phần tử thứ nhất là giá trị ngày, phần tử thứ hai là giá trị tháng và phần tử thứ ba là giá trị năm theo lịch dương sau khi chuyển đổi thành công.

+ **public static int**[] convertSolar2Lunar(**int** dd, **int** mm, **int** yy, **double** timeZone)

=> Chức năng: chuyển đổi từ ngày dương sang ngày âm

- Đầu vào: ngày, tháng, năm dương lịch và timeZone

- Đầu ra: là một mảng số nguyên trong đó giá trị đầu tiên là ngày, thứ hai là tháng, thứ ba là năm và thứ tư là trạng thái tháng chứa ngày đó có nhuận hay không.

+ **public static int**[] convertLunar2Solar(**int** lunarDay, **int** lunarMonth, **int** lunarYear, **int** lunarLeap, **double** timeZone)

=> Chức năng: Chuyển đổi từ ngày âm sang ngày dương

- Đầu vào: ngày, tháng, năm âm lịch và timeZone

- Đầu ra: là một mảng số nguyên trong đó phần tử thứ nhất là ngày, thứ hai là tháng, thứ ba là năm theo dương lịch. Nếu quá trình chuyển đổi không thành công thì giá trị các phần tử trong mảng trả về đều bằng 0.

+ **public static** NumDayOfMonthResult getNumDayOfMonth(**int** lunarYear, **int** lunarMonth, **int** isLeapMonth)

=> Chức năng: Tính số ngày của một tháng âm

- Đầu vào: năm âm, tháng âm và trạng thái của tháng là có phải tháng nhuận hay không

- Đầu ra: Một object lưu giữ những thông tin về số ngày của tháng đó và thông tin về ngày đầu tiên của tháng đó.

**c. ChineseCalendarMatris**

\* Đây là class được kế thừa từ lớp cha là CalendarMatrics sau đó override một số phương thức của lớp cha cung cấp để có thể tương thích với quá trình xử lý của library này. Công việc của ta là xử lý logic cho các hàm bắt buộc phải override đó cho đúng như vậy ta sẽ có đủ cơ sở cho việc mở rộng thêm loại Calendar support của library này.

\* Các thuộc tính quan trọng trong class

**public static int** *isEndOfMonth* (là một số nguyên biểu diễn trạng thái của một ngày đang xét có phải là ngày cuối tháng theo loại lịch ChineseCalendarMatrics hay không, ở đây một ngày được xác định là cuối tháng theo lịch ChineseCalendar khi tháng đó có 30 ngày và ngày đang xét cũng là ngày 30 của tháng.

**public final static** String ***CALENDAR\_SCALE\_ALIAS*** và **public final static** String ***CALENDAR\_SCALE\_NAME*** đây là các chuỗi giúp phân biệt giữa các loại lịch khi được thêm vào Map quản lý các loại lịch, Map này sẽ được nói rõ hơn ở mục sau.

\* Các method quan trọng trong class

Điều lưu ý: Hầu hết các method trong class này nếu method nào có đầu vào liên quan đến ngày, tháng, năm (chỉ cần 1 hoặc vài thành phần chứ không nhất thiết phải đủ 3 thành phần ngày, tháng, năm) thì ta hiểu đó là ngày, tháng, năm theo lịch âm, nếu đầu vào ở dạng milisecond thì đó là theo lịch dương. Và đầu ra của các medthod nếu là ngày, tháng, năm, hoặc giá trị ở dạng milisecond thì ta hiểu đó là giá trị theo ngày dương được tính toán từ đầu vào là của lịch âm. Ở đây ta nói là hầu hết vì sẽ có trường hợp ngoại lệ cụ thể là method **public int** getDaysPerPackedMonth(**int** year, **int** packedMonth) hàm này đầu vào sẽ là năm và tháng dương, đầu ra là số ngày của tháng dương. Tại sao có sự khác biệt như vậy sẽ giải thích rõ khi nói về method này.

Dưới đây là một số method quan trọng của class này:

+ **public int** getPackedMonthOfYearDay(**int** year, **int** yearDay)

=> Chức năng: Trả về thông tin là tháng thứ bao nhiêu khi biết năm âm lịch và số thứ tự của ngày đó trong năm đó, số thứ tự ngày tính từ 1

- Đầu vào: năm âm và số thứ tự của ngày đang xét trong năm đó.

- Đầu ra: số thứ tự của tháng chứa ngày đang xét, số thứ tự tháng tính từ 0

+ **public int** getDayOfMonthOfYearDay(**int** year, **int** yearDay)

=> Chức năng: vị trí của ngày đang xét trong tháng khi biết năm âm và vị trí của ngày đó trong năm.

- Đầu vào: năm âm và vị trí của ngày đang xét trong năm

- Đầu ra: Số thứ tự của ngày đó trong tháng, số thứ tự ngày trong tháng tính từ 1

+ **public int** getMonthAndDayOfYearDay(**int** year, **int** yearDay)

=> Chức năng: trả về thông tin ngày và tháng khi biết năm âm và vị trí ngày đang xét trong năm đó.

- Đầu vào: năm âm và vị trí của ngày đang xét trong năm đó

- Đầu ra: là một số nguyên biểu diễn thông tin về ngày tháng của ngày đang xét, trong đó byte thấp nhất là thông tin về vị trí tháng mà ngày đang xét thuộc vào, byte thấp thứ hai là vị trí ngày trong tháng của ngày đang xét thuộc vào.

+ **public int** getYearDaysForPackedMonth(**int** year, **int** packedMonth)

=> Chức năng: trả về vị trí ngày đầu tiên của tháng đang xét trong năm

- Đầu vào: năm âm và tháng âm (giá trị tháng bắt đầu từ 0)

- Đầura: là vị trí ngày trong năm của ngày đầu tiên tháng đang xét.

+ **public long** toMillis(TimeZone timeZone, **int** year, **int** packedMonth, **int** dayOfMonth, **int** hours, **int** minutes, **int** seconds, **int** millis)

=> Chức năng: chuyển đổi thời gian ngày âm sang milisecond ở dạng ngày dương

- Đầu vào: là thông tin theo ngày âm

- Đầu ra: giá trị milisecond của thời gian âm đang xét ở dạng ngày dương

+ **public long** toInstance(**long** timestamp, TimeZone timeZone)

=> Chức năng: Chuyển đổi milisecond ngày dương với timeZone đã biết sang milisecond ở dạng ngày dương.

- Đầu vào: milisecond ngày dương và timezone

- Đầura: milisecond theo ngày dương ứng với timeZone đã cho

+ **public long** nextMonth(**long** instance)

=> Chức năng: trả về milisecond của ngày sau thời điểm truyền vào 1 tháng.

+ Đầu vào: milisecond ngày dương

+ Đầu ra milisecond ngày dương của ngày sau đó một tháng

- **public long** nextMonth(**long** instance, **int** n)

=> Tương tự như method ở trên nhưng bước nhảy là n tháng sau đó

- **public long** prevMonth(**long** instance)

=> Trả về thời điểm trước đó một tháng

- **public long** prevMonth(**long** instance, **int** n)

=> Trả về thời điểm trước đó n tháng.

(Tương tự cũng có 4 trường hợp với ngày)

+ **public int** getDaysPerPackedMonth(**int** year, **int** packedMonth)

=> Chức năng: Trả về số ngày của một tháng theo lịch dương, với đầu vào là năm dương và tháng dương.

+ Đầu vào: năm và tháng dương

+ Đầu ra: Số ngày của tháng dương đang xét.

Tại sao method này lại phải có đầu vào là năm và tháng dương. Lý do là hàm này được gọi trong quá trình validate kết quả tính toán được có phải là một ngày hợp lệ hay không. Tuy nhiên qúa trình validate diễn ra sau khi đã tính toán. Tức là trước đó đã gọi một method nào đó khác của ChineseCalendarMatrics (gọi là công việc 1), nhưng đặc trưng của hầu hết các method khác là có đầu vào là theo lịch âm nhưng đầu ra là theo lịch dương. Chính vì thế sau khi thực hiện xong công việc 1 thì kết quả trả về đã là một giá trị là ngày theo lịch dương, tiếp theo đó mới đem giá trị này đi validate, trong lúc validate có gọi đến method **public int** getDaysPerPackedMonth(**int** year, **int** packedMonth)như vậy đầu vào lúc này của method phải là năm và tháng theo lịch dương.

**2. Một số class gốc của thư viện được chỉnh sửa**

**a. FastBirthdayIterator (đường dẫn recur/FastBirthdayIterator.java)**

\* Đây là class kế thừa từ class ByExtender mà ByExtender lại kế thừa tử RuleIterator. Như vậy đây là một Interator hổ trợ việc duyệt danh sách theo năm. Trong class này ta cần chú ý đến method **public long** next() đây là method giúp trả về instance kế tiếp trong danh sách. Điều cần làm là trong phần xử lý của method này là ta thực hiện việc rẽ nhánh, nếu loại calendar đang sử dụng là thuộc các loại mặc định của library cung cấp thì xử lý theo mặc đinh, ngược lại nếu là một loại calendar nào khác do ta bổ sung thêm thì ta sẽ viết code xử lý trả về đối tượng kế tiếp theo cho loại calendar đó cụ thể trong trường hợp hiện tại thì ta đang support thêm ChineseCalendarMatrics.

**b. FastWeeklyIterator (đường dẫn recur/FastWeeklyIterator.java)**

\* Đây là một class có vai trò tương tự như FastBirthdayIterator nhưng đang thực hiện lặp theo chu kì là tuần. Vì thế công việc phải làm ở class này là tương tự như đã mô tả ở trên.

=> Lưu ý đối với trường hợp lặp theo ngày thì ta không cần phải sửa gì thêm.

**c. UnicodeCalendarScales (đường dẫn recur/UnicodeCalendarScales.java)**

\* Trong file này ta quan tâm đến thuộc tính **private final static** Map<String, CalendarMetricsFactory> ***CALENDAR\_SCALES*** đây là một HashMap lưu giữ tất cả các loại Calendar mà library có thể support. Do đó sau khi đã tạo được một ChineseCalendarMatrics, cũng như đã chỉnh sửa xong một số thứ trong những file mô tả ở trên thì để tích hợp loại calendar này vào library, công việc cần làm là thêm một Instance loại ChineseCalendarMatrics vào trong HashMap CALENDAR\_SCALE là được.