ĐẠI HỌC ĐÀ NẴNG TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

PBL4: DỰ ÁN HỆ ĐIỀU HÀNH & MẠNG MÁY TÍNH

Đề tài 406:

Xây dựng chương trình client – server

để in lịch thế kỷ

SINH VIÊN THỰC HIỆN:

Nguyễn Quang Hoàng 20TCLC_DT4 NHÓM: 6

Trần Thị Hương Trinh 20TCLC_DT4 NHÓM: 6

GIẢNG VIÊN HƯỚNG DẪN: Th.S Nguyễn Văn Nguyên

Đà Nẵng 11/ 2022

MỤC LỤC

DANH SÁCH HÌNH VẼ

MỞ ĐẦU

Ngày nay, lịch là một khái niệm không hề xa lạ với con người, chắc hẳn bạn sẽ không thể tưởng tượng ra được nếu thế giới mình đang sống mà không có lịch, không có thời gian được quy định tính toán rõ ràng thì cuộc sống bạn sẽ hỗn loạn ra sao. Vì thế lịch đã là một khái niệm xuất hiện từ rất lâu, bộ lịch đầu tiên đã được người Ai Cập sáng tạo ra hơn 10.000 năm, sau đó nhiều nước đã kế thừa và phát triển như bộ lịch mà chúng ta thấy ngày nay. Lịch được xây dựng theo quy tắc riêng, có hệ thống rõ ràng giúp con người có thể sắp xếp các khoảng thời gian hợp lý trong đời sống, công việc cũng như các mục đích lịch sử và khoa học khác nhau v..v.

Công nghệ ngày càng phát triển, lịch từ đó cũng phát triển theo nhu cầu của con người qua thời gian. Từ lịch giấy, lịch trên ứng dụng, con người đã phổ biến trong việc sử dụng lịch trên web, chỉ cần gõ vào con số một năm bất kì, chưa đầy 3 giây đã có toàn bộ dữ liệu của năm đó, ngoài ra việc sử dụng lịch trên web đảm bảo được sự cập nhật, đổi mới thông tin chính xác mà không cần phải tốn quá nhiều dung lượng để cài đặt phần mềm. Nhờ sự phát triển của công nghệ web và sự hộ trợ ngày càng mạnh của các ngôn ngữ lập trình, việc tạo một trang web có khả năng hoạt động với các chức năng như một ứng dụng được cài trên máy tính là hoàn toàn có thể. Vì vậy, việc đưa một ứng dụng trên máy tính thành một trang web trở thành một nhu cầu thiết thực và cần thiết.

Với những lí do trên, đồ án PBL4 lần này chúng em sẽ xây dựng chương trình client – server in ra lịch thế kỷ dựa trên những kiến thức về lập trình mạng và mạng máy tính.

Chúng em xin chân thành cảm ơn sự hướng dẫn tận tình của thầy Nguyễn Văn Nguyên đã trao đổi, góp ý giúp chúng em hoàn thành đề tài này. Với sự hạn chế về mặt kiến thức nên báo cáo của chúng em vẫn còn nhiều thiết sót, vậy chúng em rất mong nhận được sự góp ý thêm từ quý thầy cô và các bạn.

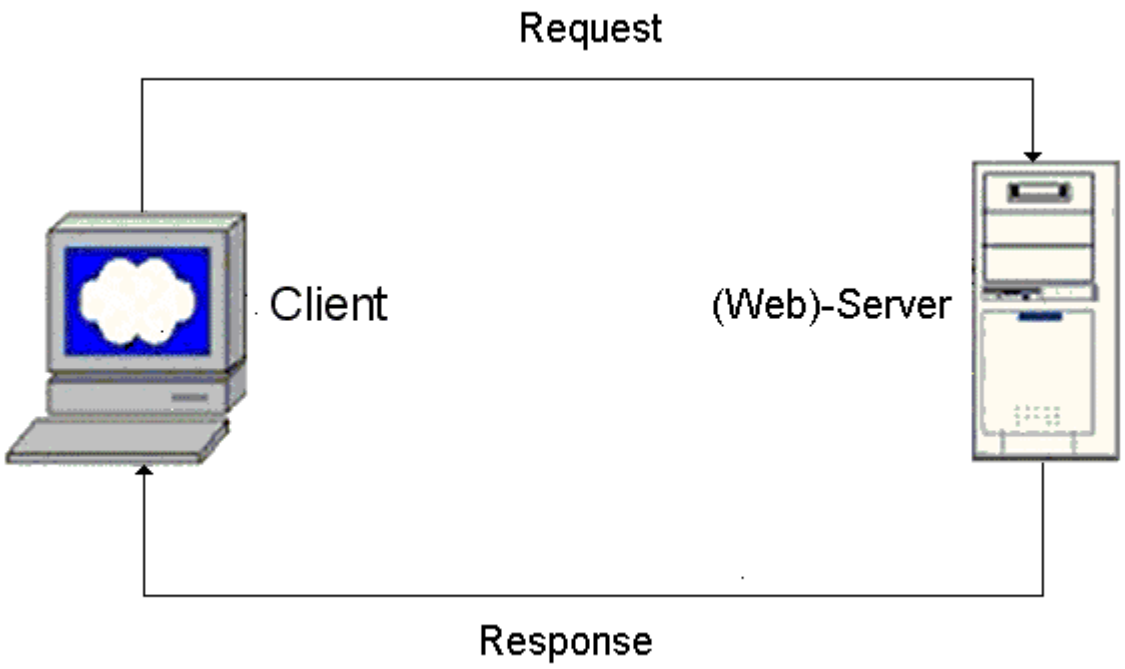
CHƯƠNG 1 CƠ SỞ LÝ THUYẾT VÀ THUẬT TOÁN

1.1 Mô hình client – server

1.1.1 Khái niệm

Mô hình client - server là mô hình giúp các máy tính giao tiếp truyền tải dữ liệu cho nhau. Trong mô hình này gồm có hai thành phần chính là máy trạm (client) và máy chủ (server). Một hoặc nhiều máy trạm sẽ kết nối đến server để yêu cầu một tài nguyên, xử lý tác vụ nào đó. Trong khi đó, server sẽ tiếp nhận các yêu cầu, xử lý và trả về kết quả cho máy trạm.

1.1.2 Nguyên lý hoạt động



Hình 1: Nguyên lý hoạt động mô hình client – server

- Client:
    - Khởi tạo liên lạc với server
    - Yêu cầu dịch vụ nào đó với server
    - Đối với web, client được thực hiện trong browser
- Server:
    - Cung cấp dịch vụ, yêu cầu cho client
    - Chẳng hạn web server gửi web yêu cầu hay mail server phân phát email

1.1.3 Ưu, nhược điểm

- Ưu điểm:
    - Tính tập trung: Truy cập tài nguyên và bảo mật dữ liệu, tính tập trung thông qua server
    - Tính co giãn: Có thể nâng cấp bất cứ thành phần nào khi cần thiết
    - Tính mềm dẻo: Công nghệ mới có thể dễ dàn tích hợp vào hệ thống
    - Tính trao đổi tương tác: Tất cả các thành phần (client, mạng, server) cùng nhau làm việc
- Nhược điểm:
    - Quản trị hệ thống khó khăn: Duy trì thông tin cấu hình luôn cập nhật và nhất quán giữa tất cả các thiết bị
    - Nâng cấp phiên bản mới khó đồng bộ
    - Phụ thuộc độ tin cậy của mạng
    - Chi phí thiết kế, cài đặt, quản trị và bảo trì cao
    - Phải giải quyết:
        - Sự xung đột trong hệ thống
        - Tính tương thích của các thành phần

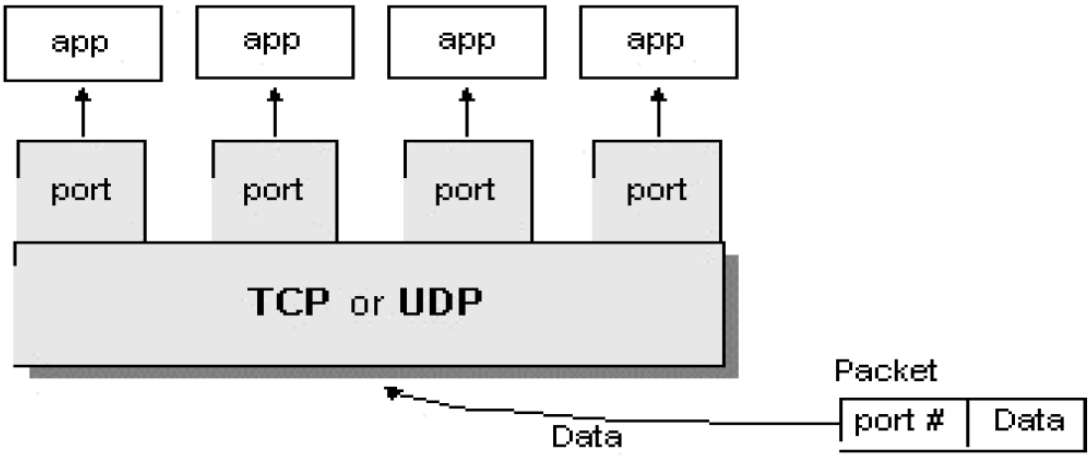
1.2 Lập trình với giao thức TCP

1.2.1 Giao thức TCP

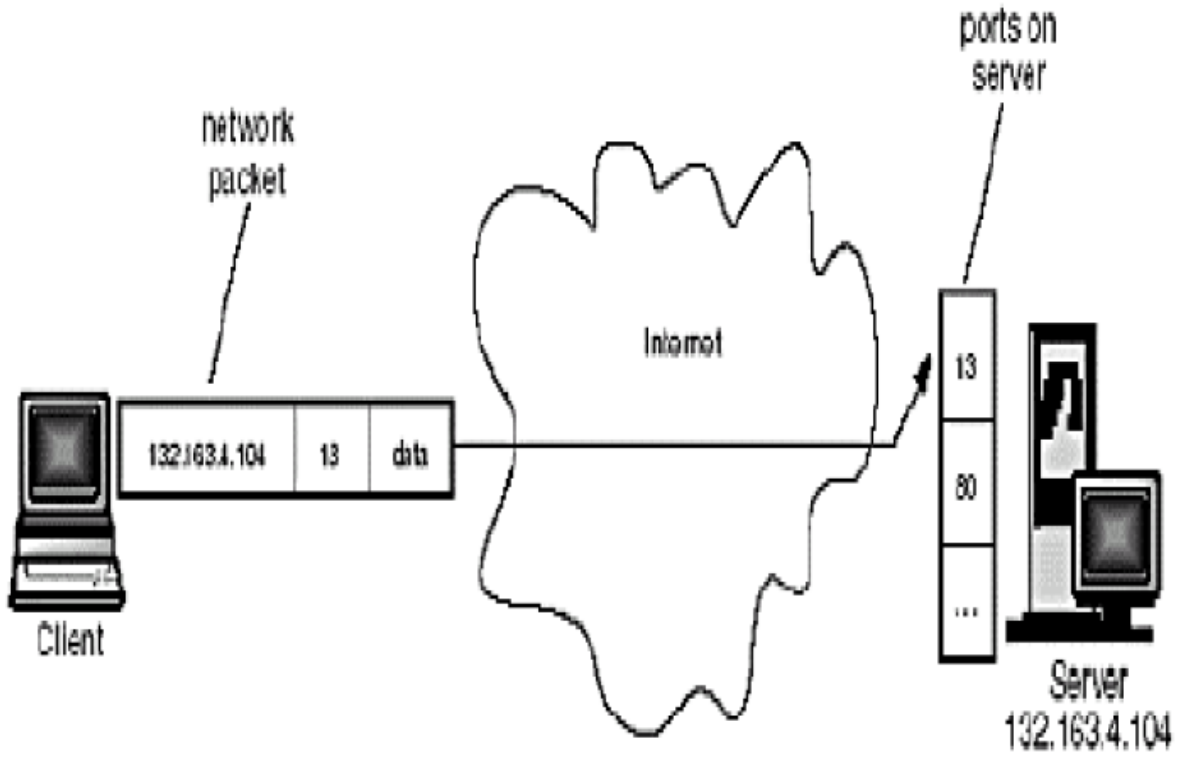
Giao thức TCP là giao thức hướng kết nối (Connection oriented) truyền thông kết nối và tin cậy. Cung cấp một kênh point-to-point cho các ứng dụng yêu cầu truyền thông.'

1.2.2 Cổng (port)

Cổng (port) là một số (nhãn) đặc biệt được gán cho một tiến trình mạng. Mỗi tiến trình mạng đều được gán cho một cổng duy nhất.



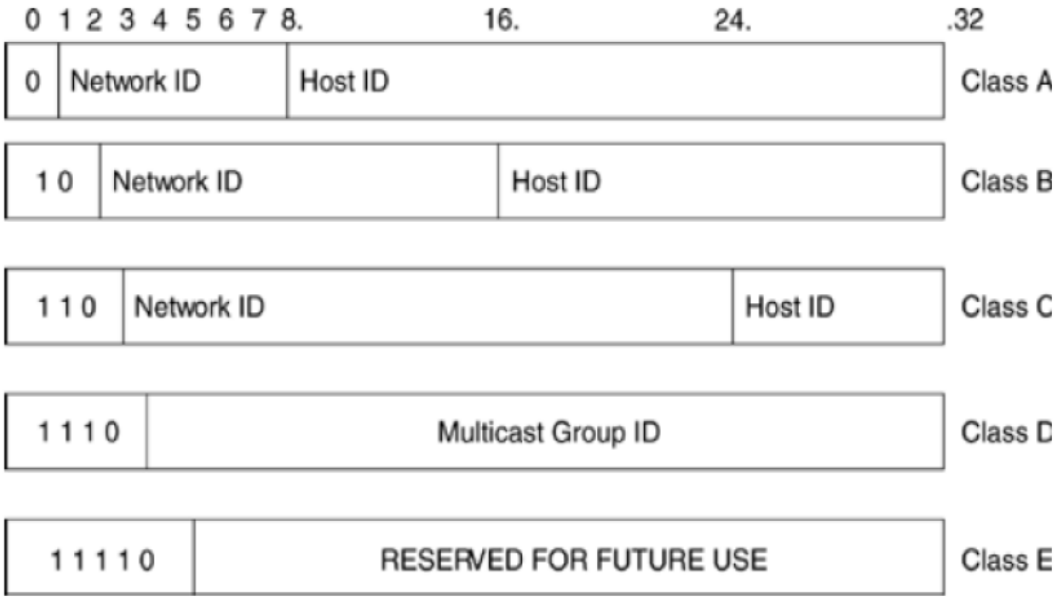
Hình 2: Mô tả cổng



Hình 3: Cổng được gán cho tiến trình kết nối client-server

1.2.3 Địa chỉ IP

Mỗi thực thể mạng chỉ có một địa chỉ IP duy nhất. IPV4 có 32 bit, tạo thành từ 4 octecs. Địa chỉ IP chia thành các lớp sau:



Hình 4: Các lớp trong địa chỉ IP

1.2.4 Socket

1. Khái niệm

Một ứng dụng trên mạng được xác định thông qua địa chỉ IP mà nó chạy trên một hệ thống và số hiệu cổng riêng được gán riêng cho nó. Hai ứng dụng mạng liên lạc được với nhau cần phải thiết lập kết nối, mỗi đầu kết nối tương ứng với một Socket.

Vậy SocKet là một đầu cuối của một sự truyền thông 2 chiều liên kết liên kết giữu hai chương trình chạy trên mạng. Một socket được gán với một số hiệu cổng, vì thế tầng giao vận có thể nhận biết ứng dụng mà dữ liệu được chuyển đến

Các kiểu hoạt động của Socket:

- Kết nối đến máy ở xa
- Gửi dữ liệu
- Nhận dữ liệu
- Đóng kết nối

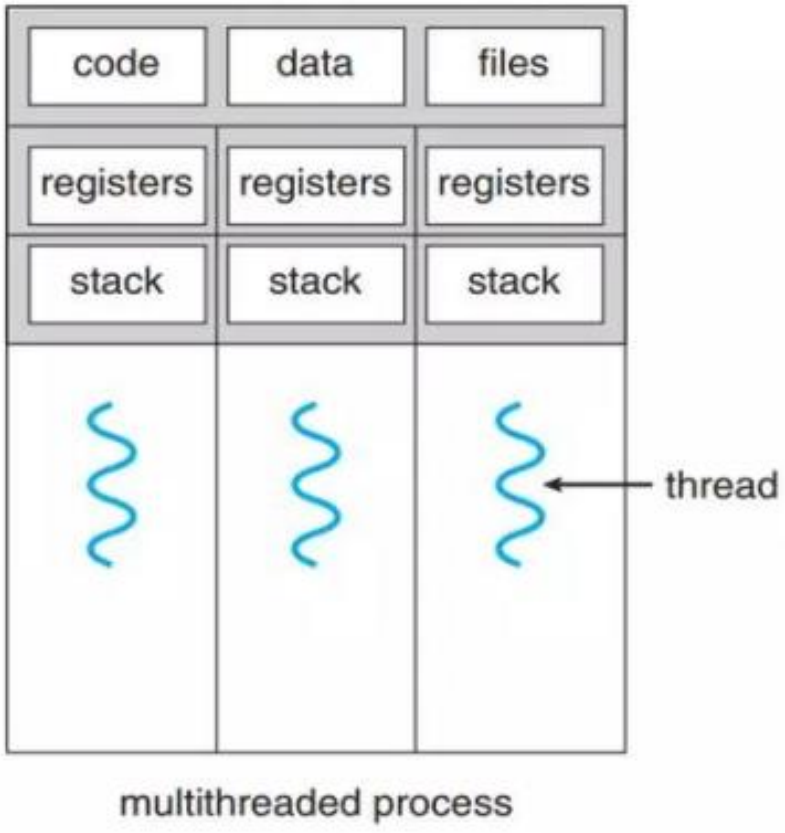
- Gắn với một cổng
- Lắng nghe dữ liệu đến
- Chấp nhận kết nối từ máy ở xa trên cổng được gán

2. Nguyên lý hoạt động

Chức năng của Socket chính là kết nối giữa client và server thông qua TCP/IP và UDP để truyền cũng như nhận dữ liệu qua Internet. Giao diện Socket chỉ có thể hoạt động khi đã có số hiệu cổng của 2 ứng dụng cần trao đổi dữ liệu và thông số IP. Đầu tiên, server sẽ mở một kết nối trên địa chỉ mà các client có thể kết nối đến. Sau khi kết nối được mở, server sẽ chờ các client gửi request đến. Quá trình trao đổi dữ liệu giữa máy khách và máy chủ diễn ra khi máy khách kết nối với máy chủ thông qua socket. Máy chủ thực hiện yêu cầu của khách hàng và gửi lại trả lời cho khách hàng.

1.3 Lập trình đa luồng

Đa luồng là một tiến trình thực hiện nhiều luồng đồng thời. Để cho phép nhiều client có thể kết nối đến server cùng lúc thì server phải là chương trình đa tuyến. Mỗi tuyến ứng với một kết nối từ client. Một ứng dụng ngoài luồng chính có thể có các luồng khác thực thi đồng thời làm ứng dụng chạy nhanh và hiệu quả hơn.



Hình 5: Đa luồng

1.4 Năm dương lịch

1.4.1 Tổng quan

Năm dương lịch được tính bằng đơn vị thời gian trái đất quay quanh một vòng Mặt trời. Một vòng quay của Trái đất quanh Mặt trời hết 365,2422 ngày (365 ngày 5 giờ 48 phút 46 giây). Để tiện tính toán người ta tính chẵn 265 ngày là một năm dương lịch. Do trong 265 ngày có 12 lần mặt trăng tròn khuyết nên người ta chia thành 12 tháng. Vì 365 không chia hết cho 12 nên đành chia thành tháng đủ (31 ngày) và tháng thiếu (30 ngày). Riêng tháng 2 cũng là tháng thiếu nhưng chỉ có 28 ngày. Như vậy cộng 12 tháng lại vừa đủ 265 ngày, đó là năm bình thường.

Nhưng còn dư 5 giờ 48 phút 46 giây thì tính sao đây? Trong 4 năm liền, số dư đó cộng lại suýt soát 1 ngày. Và một ngày đó được cộng vào tháng 2 của năm thứ tư. Năm thứ tư ấy gọi là "năm nhuận" có 366 ngày. Tháng 2 của năm nhuận có 29 ngày.

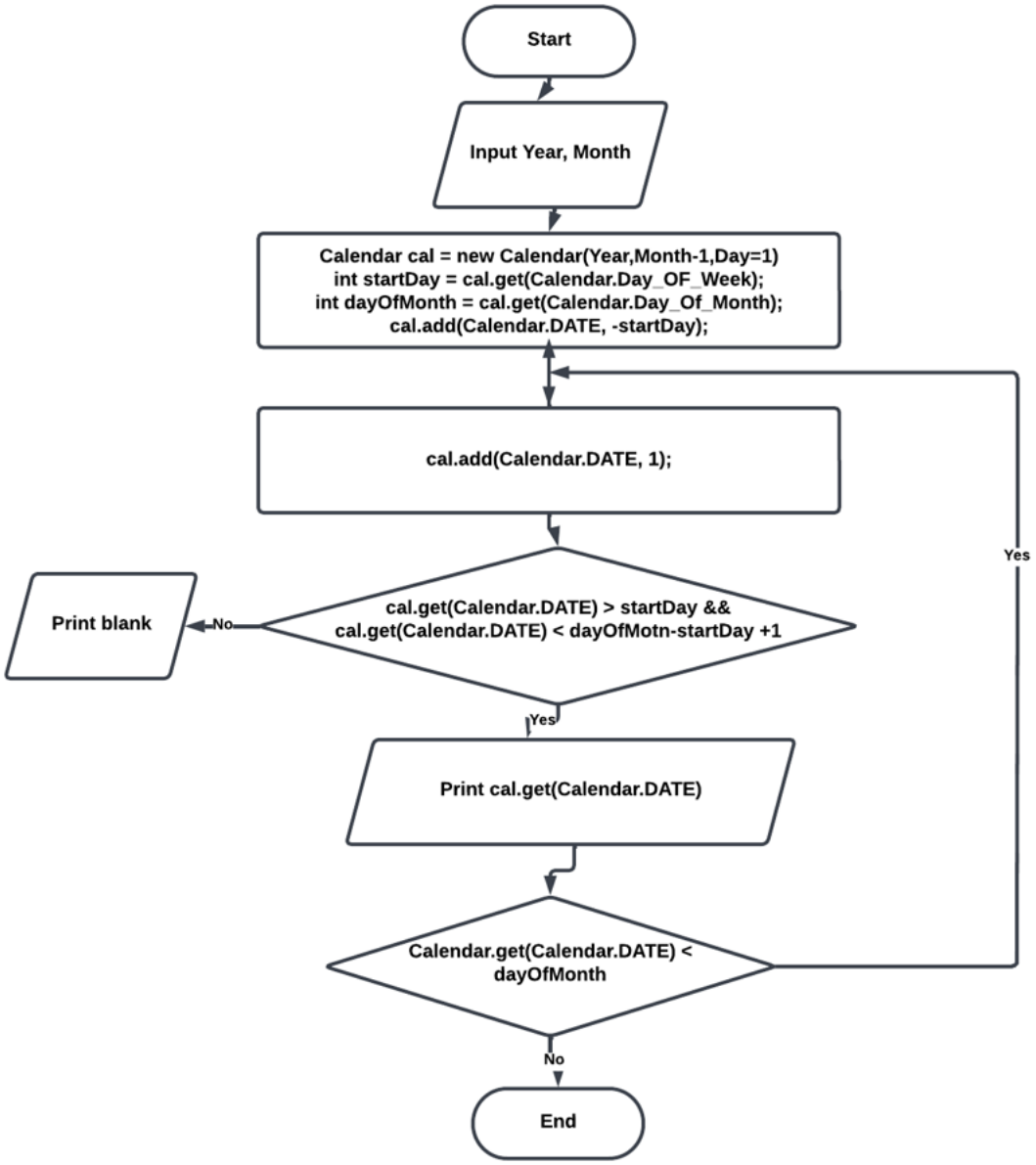
Ngày thứ 29 ấy gọi là "ngày nhuận".

1.4.2 Thuật toán

Ta sử dụng lớp Calendar trong java. Lớp Calendar là một lớp trừu tượng cung cấp phương thức chuyển đổi ngày giữa một thời điểm cụ thể theo thời gian và một tập hợp các trường của calendar như MONTH, YEAR, HOUR, ... Nó kế thừa lớp Object và implements giao diện Comparable

Các phương thức trong lớp Calendar:

Phương thức	Mô tả
abstract void add(int field, int amount)	Nó được sử dụng để thêm hoặc trừ số lượng thời gian nhất định vào trường calendar đã cho, dựa trên các quy tắc của calendar.
int get(int field)	Được sử dụng để trả lại trường Calendar đã cho
static Calendar getInstance()	Được dùng để lấy Calendar sử dụng timezone và locate mặc định
abstract int getMaximum(int field)	Trả về giá trị max cho trường calendar đã cho của thể hiện calendar hiện tại
void set(int field, int value	Dùng để thiết lập trường cho trước với các giá trị đã cho
Date getTime()	Nó được sử dụng để trả về đối tượng Date biểu diễn giá trị time của Calendar.



Hình 6: Sơ đồ khối thuật toán lịch dương

1.5 Năm âm lịch

1.5.1 Tổng quan

Âm lịch Việt Nam là một loại lịch thiên văn. Nó được tính toán dựa trên sự chuyển động của mặt trời, trái đất và mặt trăng. Ngày tháng âm lịch được tính dựa theo các nguyên tắc sau:

1. Ngày đầu tiên của tháng âm lịch là ngày chứa điểm Sóc
2. Một năm bình thường có 12 tháng âm lịch, một năm nhuận có 13 tháng âm lịch
3. Đông chí luôn rơi vào tháng 11 âm lịch
4. Trong một năm nhuận, nếu có 1 tháng không có Trung khí thì tháng đó là tháng nhuận. Nếu nhiều tháng trong năm nhuận đều không có Trung khí thì chỉ tháng đầu tiên sau Đông chí là tháng nhuận
5. Việc tính toán dựa trên kinh tuyến 105° đông.

Sóc là thời điểm hội diện, đó là khi trái đất, mặt trăng và mặt trời nằm trên một đường thẳng và mặt trăng nằm giữa trái đất và mặt trời. (Như thế góc giữa mặt trăng và mặt trời bằng 0 độ). Gọi là "hội diện" vì mặt trăng và mặt trời ở cùng một hướng đối với trái đất. Chu kỳ của điểm Sóc là khoảng 29,5 ngày. Ngày chứa điểm Sóc được gọi là ngày Sóc, và đó là ngày bắt đầu tháng âm lịch.

Trung khí là các điểm chia đường hoàng đạo thành 12 phần bằng nhau. Trong đó, bốn Trung khí giữa bốn mùa là đặc biệt nhất: Xuân phân (khoảng 20/3), Hạ chí (khoảng 22/6), Thu phân (khoảng 23/9) và Đông chí (khoảng 22/12).

Bởi vì dựa trên cả mặt trời và mặt trăng nên lịch Việt Nam không phải là thuần âm lịch mà là âm-dương-lịch. Theo các nguyên tắc trên, để tính ngày tháng âm lịch cho một

năm bất kỳ trước hết chúng ta cần xác định những ngày nào trong năm chứa các thời điểm Sóc (New moon) . Một khi bạn đã tính được ngày Sóc, bạn đã biết được ngày bắt đầu và kết thúc của một tháng âm lịch: ngày mùng một của tháng âm lịch là ngày chứa điểm sóc. Sau khi đã biết ngày bắt đầu/kết thúc các tháng âm lịch, ta tính xem các Trung khí (Major solar term) rơi vào tháng nào để từ đó xác định tên các tháng và tìm tháng nhuận.

Đông chí luôn rơi vào tháng 11 của năm âm lịch. Bởi vậy chúng ta cần tính 2 điểm sóc: Sóc A ngay trước ngày Đông chí thứ nhất và Sóc B ngay trước ngày Đông chí thứ hai. Nếu khoảng cách giữa A và B là dưới 365 ngày thì năm âm lịch có 12 tháng, và những tháng đó có tên là: tháng 11, tháng 12, tháng 1, tháng 2, …, tháng 10. Ngược lại, nếu khoảng cách giữa hai sóc A và B là trên 365 ngày thì năm âm lịch này là năm nhuận, và chúng ta cần tìm xem đâu là tháng nhuận. Để làm việc này ta xem xét tất cả các tháng giữa A và B, tháng đầu tiên không chứa Trung khí sau ngày Đông chí thứ nhất là tháng nhuận. Tháng đó sẽ được mang tên của tháng trước nó kèm chữ "nhuận".

Khi tính ngày Sóc và ngày chứa Trung khí cần lưu ý xem xét chính xác múi giờ. Đây là lý do tại sao có một vài điểm khác nhau giữa lịch Việt Nam và lịch Trung Quốc.Ví dụ, nếu ta biết thời điểm hội diện là vào lúc yyyy-02-18 16:24:45 GMT thì ngày Sóc của lịch Việt Nam là 18 tháng 2, bởi vì 16:24:45 GMT là 23:24:45 cùng ngày, giờ Hà nội (GMT+7, kinh tuyến 105° đông). Tuy nhiên theo giờ Bắc Kinh (GMT+8, kinh tuyến 120° đông) thì Sóc là lúc 00:24:45 ngày yyyy-02-19, do đó tháng âm lịch của Trung Quốc lại bắt đầu ngày yyyy-02-19, chậm hơn lịch Việt Nam 1 ngày.

1.5.2 Thuật toán

Trong tính toán thiên văn người ta lấy ngày 1/1/4713 trước công nguyên của lịch Julius (tức ngày 24/11/4714 trước CN theo lịch Gregorius) làm điểm gốc. Số ngày tính từ điểm gốc này gọi là số ngày Julius (Julian day number) của một thời điểm. Ví dụ, số ngày Julius của 1/1/2000 là 24515455.

Dùng các công thức sau ta có thể chuyển đổi giữa ngày/tháng/năm và số ngày Julius. Phép chia ở 2 công thức sau được hiểu là chia số nguyên, bỏ phần dư: 23/4=5.

1.5.2.1 Đổi ngày dd/mm/yyyy ra số ngày Julius jd

a = (14 - mm) / 12

y = yy+4800-a

m = mm+12*a-3

Lịch Gregory:

jd = dd + (153*m+2)/5 + 365*y + y/4 - y/100 + y/400 - 32045

Lịch Julius:

jd = dd + (153*m+2)/5 + 365*y + y/4 – 32083

1.5.2.2 Tính ngày Sóc

Như trên đã nói, để tính được âm lịch trước hết ta cần xác định các tháng âm lịch bắt đầu vào ngày nào.

Ta sẽ tính ngày Sóc thứ k kể từ điểm Sóc ngày 1/1/1900. Kết quả trả về là số ngày Julius của ngày Sóc cần tìm.

Với hàm này ta có thể tính được tháng âm lịch chứa ngày N bắt đầu vào ngày nào: giữa ngày 1/1/1900 (số ngày Julius: 2415021) và ngày N có khoảng k=INT((N-2415021)/29.530588853) tháng âm lịch, như thế dùng hàm này sẽ biết ngày đầu tháng âm lịch chứa ngày N, từ đó ta biết ngày N là mùng mấy âm lịch.

1.5.2.3 Tính tọa độ mặt trời

Để biết Trung khí nào nằm trong tháng âm lịch nào, ta chỉ cần tính xem mặt trời nằm ở khoảng nào trên đường hoàng đạo vào thời điểm bắt đầu một tháng âm lịch. Ta chia đường hoàng đạo làm 12 phần và đánh số các cung này từ 0 đến 11: từ Xuân phân đến Cốc vũ là

0; từ Cốc vũ đến Tiểu mãn là 1; từ Tiểu mãn đến Hạ chí là 2; v.v.. Cho jdn là số ngày Julius của bất kỳ một ngày, phương pháp sau này sẽ trả lại số cung nói trên.

Với hàm này ta biết được một tháng âm lịch chứa Trung khí nào. Giả sử một tháng âm lịch bắt đầu vào ngày N1 và tháng sau đó bắt đầu vào ngày N2 và hàm getSunLongitude cho kết quả là 8 với N1 và 9 với N2. Như vậy tháng âm lịch bắt đầu ngày N1 là tháng chứa Đông chí: trong khoảng từ N1 đến N2 có một ngày mặt trời di chuyển từ cung 8 (sau Tiểu tuyết) sang cung 9 (sau Đông chí). Nếu hàm **Tính tọa độ Mặt trời** trả lại cùng một kết quả cho cả ngày bắt đầu một tháng âm lịch và ngày bắt đầu tháng sau đó thì tháng đó không có Trung khí và như vậy có thể là tháng nhuận.

1.5.2.4 Xác định tháng nhuận

Nếu giữa hai tháng 11 âm lịch (tức tháng có chứa Đông chí) có 13 tháng âm lịch thì năm âm lịch đó có tháng nhuận. Để xác định tháng nhuận, ta sử dụng hàm **tính tọa độ mặt trời** như đã nói ở trên. Cho a11 là ngày bắt đầu tháng 11 âm lịch mà một trong 13 tháng sau đó là tháng nhuận. Hàm sau cho biết tháng nhuận nằm ở vị trí nào sau tháng 11 này.

Giả sử hàm **Xác định tháng nhuận** trả lại giá trị 4, như thế tháng nhuận sẽ là tháng sau tháng 2 thường. (Tháng thứ 4 sau tháng 11 đáng ra là tháng 3, nhưng vì đó là tháng nhuận nên sẽ lấy tên của tháng trước đó tức tháng 2, và tháng thứ 5 sau tháng 11 mới là tháng 3).

1.5.2.5 Đổi ngày dương dd/mm/yyyy ra ngày âm

Với các phương pháp hỗ trợ trên ta có thể đổi ngày dương dd/mm/yy ra ngày âm dễ dàng. Trước hết ta xem **ngày Sóc** tháng âm lịch chứa ngày này là ngày nào (dùng hàm **Tính ngày Sóc** như trên đã nói). Sau đó, ta tìm các ngày a11 và b11 là ngày bắt đầu các tháng 11 âm lịch trước và sau ngày đang xem xét. Nếu hai ngày này cách nhau dưới 365 ngày thì ta chỉ còn cần xem **ngày Sóc** và a11 cách nhau bao nhiêu tháng là có thể tính được dd/mm/yy nằm trong tháng mấy âm lịch. Ngược lại, nếu a11 và b11 cách nhau khoảng 13 tháng âm lịch thì ta phải tìm xem tháng nào là tháng nhuận và từ đó suy ra ngày đang tìm nằm trong tháng nào.

CHƯƠNG 2 PHÂN TÍCH THIẾT KẾ HỆ THỐNG

2.1 Phân tích yêu cầu

- Yêu cầu bài toán:
    - Input: năm cần in lịch
    - Output: Tờ lịch của năm đó: tổng quan 12 tháng của năm đó trong đó có cả ngày âm và ngày dương, có thể xem thông tin chi tiết theo ngày

2.2 Phân tích chức năng

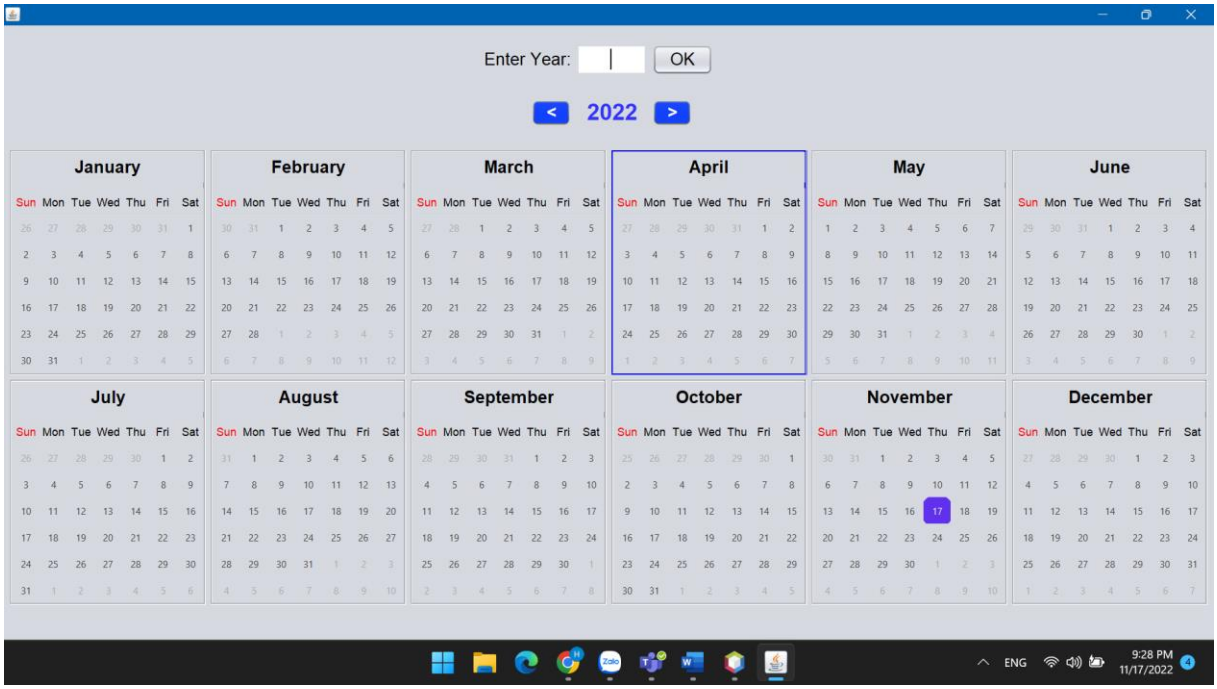
- Server:
    - Gán 1 cổng với socket
    - Chờ và lắng nghe yêu cầu kết nối từ client
    - Chấp nhận kết nối, tạo socket tương ứng
    - Lắng nghe yêu cầu của client, tiến hành sử dụng các thuật toán tính lịch dương, âm
    - Trả về kết quả cho client lịch năm đó
    - Đóng kết nối
- Client:
    - Tạo một TCP socket với địa chỉ IP và số cổng mà chương trình server đang chạy
    - Thiết lập kết nối đến server
    - Nhập năm cần xem lịch và gửi yêu cầu đến server đã kết nối
    - Nhận kết quả từ server hiển thị ra giao diện màn hình
    - Đóng kết nối

CHƯƠNG 3 TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

3.1 Triển khai

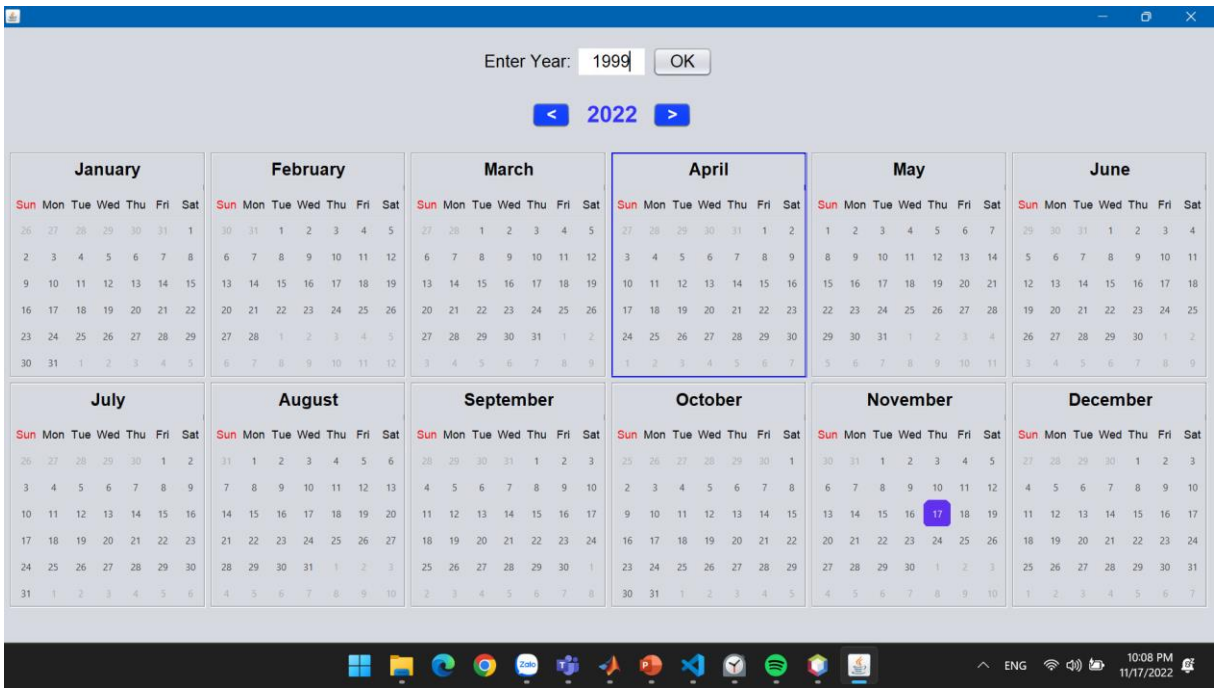
- Ứng dụng xây dựng theo mô hình Client – Server.
- Sử dụng giao thức TCP để truyền và nhận dữ liệu.
- Xây dựng chương trình client – server bằng Java
- Sử dụng GitHub để quản lý mã nguồn

3.2 Kết quả

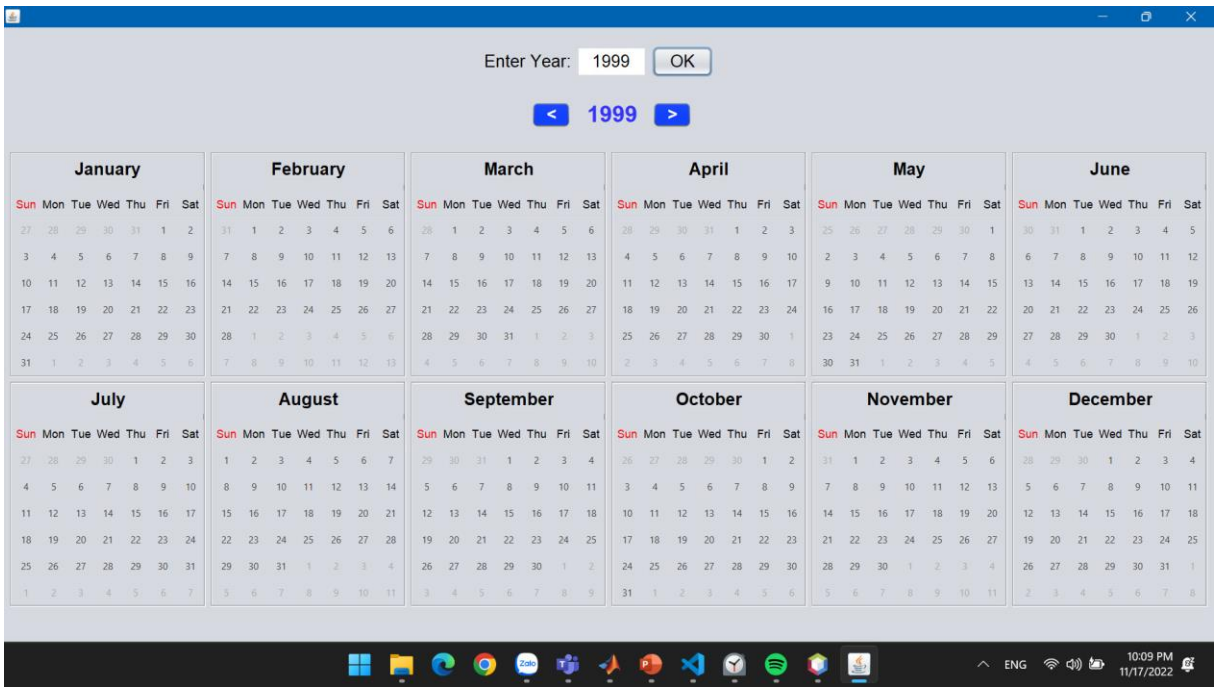


Hình 7: Giao diện chính của client là ngày tháng năm thời điểm hiện tại

Hình 8: Mô tả chi tiết tháng mà client chọn xem



Hình 9: Client nhập vào năm cần in ra



Hình 10: Server đáp ứng yêu cầu của client

Hình 11: Lịch chi tiết của tháng mà client chọn xem trong năm đó

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

- Kết luận:
    - Trong quá trình nghiên cứu, chúng em đã nắm vững về mô hình client – server, giao thức TCP, các kiến thức về mạng máy tính và hệ điều hành, biết được thuật toán để tạo ra lịch âm, lịch dương
    - Chúng em đã xây dựng được chương trình client –server in lịch theo đúng yêu cầu có cả lịch âm và lịch dương, giao diện ưa nhìn và dễ sử dụng.
- Hướng Phát triển:
    - Thêm một vài chức năng:
        - Thêm cơ sở dữ liệu thông báo ngày lễ
        - Người sử dụng có thể tạo lịch làm việc, lịch nhắc nhở, note…
        - Thêm tài khoản khác vào để có thể theo dõi lịch trình của mình
    - Tối ưu server:
        - Đảm bảo được số lượng truy cập vào hệ thống đủ lớn mà hệ thống không bị sập hoặc đứng
    - Tối ưu client:
        - Cập nhật giao diện đổi mới khi có ngày lễ…
        - Xây dựng giao diện đẹp hơn và dễ sử dụng hơn

TÀI LIỆU THAM KHẢO

[1] Phạm Minh Tuấn, Bài giảng môn học lập trình mạng

[2] Hồ Ngọc Đức, Thuật toán tính âm lịch, 2008

[3] Nguyễn Phương Lan-Hoàng Đức Hải, Java lập trình mạng, Nhà xuất bản giáo dục, 2001

[4] Nguyễn cao Đạt , Giáo trình lập trình mạng, Trường Đại học Bách Khoa

Hồ Chí Minh

[5] Hoàng Ngọc Giao, Lập trình Java thế nào?, Nhà xuất bản thống kê Hà Nội, 1998

Phụ Lục

- **Code in tháng dương lịch**

```
- private void setDate() {
-         Calendar calendar = Calendar.getInstance();
-         calendar.set(Calendar.YEAR, year);
-         calendar.set(Calendar.MONTH, month - 1);  //  month jan as 0 so start from 0
-         calendar.set(Calendar.DATE, 1);
-         int startDay = calendar.get(Calendar.DAY_OF_WEEK) - 1;  //  get day of week - 1 to index
-         calendar.add(Calendar.DATE, -startDay);
-         ToDay toDay = getToDay();
-         for (Component com : jPanel2.getComponents()) {
-             Cell cell = (Cell) com;
-             if (!cell.isTitle()) {
-                 cell.setText(calendar.get(Calendar.DATE) + "");
-                 cell.setDate(calendar.getTime());
-                 cell.currentMonth(calendar.get(Calendar.MONTH) == month - 1);
-                 if (toDay.isToDay(new ToDay(calendar.get(Calendar.DATE), calendar.get(Calendar.MONTH) + 1, calendar.get(Calendar.YEAR)))) {
-                     cell.setAsToDay();
-                 }
-                 calendar.add(Calendar.DATE, 1); //  up 1 day
-             }
```

```
-           }
-       }
```

- **Code chuyển dương lịch sang dương lịch**

```
public class VietCalendar {
    public static final double PI = Math.PI;
    /**
     *
     * @param dd
     * @param mm
     * @param yy
     * @return the number of days since 1 January 4713 BC (Julian calendar)
     */
    public static int jdFromDate(int dd, int mm, int yy) {
        int a = (14 - mm) / 12;
        int y = yy+4800-a;
        int m = mm+12*a-3;
        int jd = dd + (153*m+2)/5 + 365*y + y/4 - y/100 + y/400 - 32045;
        if (jd < 2299161) {
```

```
        jd = dd + (153*m+2)/5 + 365*y + y/4 - 32083;
    }
    //jd = jd - 1721425;
    return jd;
}
public static int[] jdToDate(int jd) {
    int a, b, c;
    if (jd > 2299160) { // After 5/10/1582, Gregorian calendar
        a = jd + 32044;
        b = (4*a+3)/146097;
        c = a - (b*146097)/4;
    } else {
        b = 0;
        c = jd + 32082;
    }
    int d = (4*c+3)/1461;
    int e = c - (1461*d)/4;
    int m = (5*e+2)/153;
```

```
        int day = e - (153*m+2)/5 + 1;
        int month = m + 3 - 12*(m/10);
        int year = b*100 + d - 4800 + m/10;
        return new int[]{day, month, year};
    }
    /**
     * Solar longitude in degrees
     * Algorithm from: Astronomical Algorithms, by Jean Meeus, 1998
     * @param jdn - number of days since noon UTC on 1 January 4713 BC
     * @return
     */
    public static double SunLongitude(double jdn) {
        //return CC2K.sunLongitude(jdn);
        return SunLongitudeAA98(jdn);
    }
    public static double SunLongitudeAA98(double jdn) {
        double T = (jdn - 2451545.0 ) / 36525; // Time in Julian centuries from 2000-01-01 12:00:00 GMT
        double T2 = T*T;
```

```
        double dr = PI/180; // degree to radian
        double M = 357.52910 + 35999.05030*T - 0.0001559*T2 - 0.00000048*T*T2; // mean anomaly, degree
        double L0 = 280.46645 + 36000.76983*T + 0.0003032*T2; // mean longitude, degree
        double DL = (1.914600 - 0.004817*T - 0.000014*T2)*Math.sin(dr*M);
        DL = DL + (0.019993 - 0.000101*T)*Math.sin(dr*2*M) + 0.000290*Math.sin(dr*3*M);
        double L = L0 + DL; // true longitude, degree
        L = L - 360*(INT(L/360)); // Normalize to (0, 360)
        return L;
    }
    public static double NewMoon(int k) {
        //return CC2K.newMoonTime(k);
        return NewMoonAA98(k);
    }
    /**
     * Julian day number of the kth new moon after (or before) the New Moon of 1900-01-01 13:51 GMT.
     * Accuracy: 2 minutes
```

```
    * Algorithm from: Astronomical Algorithms, by Jean Meeus, 1998
    * @param k
    * @return the Julian date number (number of days since noon UTC on 1 January 4713 BC) of the New Moon
    */
   public static double NewMoonAA98(int k) {
      double T = k/1236.85; // Time in Julian centuries from 1900 January 0.5
      double T2 = T * T;
      double T3 = T2 * T;
      double dr = PI/180;
      double Jd1 = 2415020.75933 + 29.53058868*k + 0.0001178*T2 - 0.000000155*T3;
      Jd1 = Jd1 + 0.00033*Math.sin((166.56 + 132.87*T - 0.009173*T2)*dr); // Mean new moon
      double M = 359.2242 + 29.10535608*k - 0.0000333*T2 - 0.00000347*T3; // Sun's mean anomaly
      double Mpr = 306.0253 + 385.81691806*k + 0.0107306*T2 + 0.00001236*T3; // Moon's mean anomaly
      double F = 21.2964 + 390.67050646*k - 0.0016528*T2 - 0.00000239*T3; // Moon's argument of latitude
      double C1=(0.1734 - 0.000393*T)*Math.sin(M*dr) + 0.0021*Math.sin(2*dr*M);
```

```
        C1 = C1 - 0.4068*Math.sin(Mpr*dr) + 0.0161*Math.sin(dr*2*Mpr);
        C1 = C1 - 0.0004*Math.sin(dr*3*Mpr);
        C1 = C1 + 0.0104*Math.sin(dr*2*F) - 0.0051*Math.sin(dr*(M+Mpr));
        C1 = C1 - 0.0074*Math.sin(dr*(M-Mpr)) + 0.0004*Math.sin(dr*(2*F+M));
        C1 = C1 - 0.0004*Math.sin(dr*(2*F-M)) - 0.0006*Math.sin(dr*(2*F+Mpr));
        C1 = C1 + 0.0010*Math.sin(dr*(2*F-Mpr)) + 0.0005*Math.sin(dr*(2*Mpr+M));
        double deltat;
        if (T < -11) {
           deltat= 0.001 + 0.000839*T + 0.0002261*T2 - 0.00000845*T3 -
0.000000081*T*T3;
        } else {
           deltat= -0.000278 + 0.000265*T + 0.000262*T2;
        };
        double JdNew = Jd1 + C1 - deltat;
        return JdNew;
    }
    public static int INT(double d) {
        return (int)Math.floor(d);
    }
```

```
public static double getSunLongitude(int dayNumber, double timeZone) {
    return SunLongitude(dayNumber - 0.5 - timeZone/24);
}
public static int getNewMoonDay(int k, double timeZone) {
    double jd = NewMoon(k);
    return INT(jd + 0.5 + timeZone/24);
}
public static int getLunarMonth11(int yy, double timeZone) {
    double off = jdFromDate(31, 12, yy) - 2415021.076998695;
    int k = INT(off / 29.530588853);
    int nm = getNewMoonDay(k, timeZone);
    int sunLong = INT(getSunLongitude(nm, timeZone)/30);
    if (sunLong >= 9) {
        nm = getNewMoonDay(k-1, timeZone);
    }
    return nm;
}
public static int getLeapMonthOffset(int a11, double timeZone) {
```

```
    int k = INT(0.5 + (a11 - 2415021.076998695) / 29.530588853);
    int last; // Month 11 contains point of sun longutide 3*PI/2 (December solstice)
    int i = 1; // We start with the month following lunar month 11
    int arc = INT(getSunLongitude(getNewMoonDay(k+i, timeZone), timeZone)/30);
    do {
        last = arc;
        i++;
        arc = INT(getSunLongitude(getNewMoonDay(k+i, timeZone), timeZone)/30);
    } while (arc != last && i < 14);
    return i-1;
}
/**
 *
 * @param dd
 * @param mm
 * @param yy
 * @param timeZone
 * @return array of [lunarDay, lunarMonth, lunarYear, leapOrNot]
```

```
 */
public static int[] convertSolar2Lunar(int dd, int mm, int yy, double timeZone) {
    int lunarDay, lunarMonth, lunarYear, lunarLeap;
    int dayNumber = jdFromDate(dd, mm, yy);
    int k = INT((dayNumber - 2415021.076998695) / 29.530588853);
    int monthStart = getNewMoonDay(k+1, timeZone);
    if (monthStart > dayNumber) {
        monthStart = getNewMoonDay(k, timeZone);
    }
    int a11 = getLunarMonth11(yy, timeZone);
    int b11 = a11;
    if (a11 >= monthStart) {
        lunarYear = yy;
        a11 = getLunarMonth11(yy-1, timeZone);
    } else {
        lunarYear = yy+1;
        b11 = getLunarMonth11(yy+1, timeZone);
    }
```

```
lunarDay = dayNumber-monthStart+1;
int diff = INT((monthStart - a11)/29);
lunarLeap = 0;
lunarMonth = diff+11;
if (b11 - a11 > 365) {
    int leapMonthDiff = getLeapMonthOffset(a11, timeZone);
    if (diff >= leapMonthDiff) {
        lunarMonth = diff + 10;
        if (diff == leapMonthDiff) {
            lunarLeap = 1;
        }
    }
}
if (lunarMonth > 12) {
    lunarMonth = lunarMonth - 12;
}
if (lunarMonth >= 11 && diff < 4) {
    lunarYear -= 1;
```

```
        }
        return new int[]{lunarDay, lunarMonth, lunarYear, lunarLeap};
    }
    public static int[] convertLunar2Solar(int lunarDay, int lunarMonth, int lunarYear, int lunarLeap, double timeZone) {
        int a11, b11;
        if (lunarMonth < 11) {
            a11 = getLunarMonth11(lunarYear-1, timeZone);
            b11 = getLunarMonth11(lunarYear, timeZone);
        } else {
            a11 = getLunarMonth11(lunarYear, timeZone);
            b11 = getLunarMonth11(lunarYear+1, timeZone);
        }
        int k = INT(0.5 + (a11 - 2415021.076998695) / 29.530588853);
        int off = lunarMonth - 11;
        if (off < 0) {
            off += 12;
        }
        if (b11 - a11 > 365) {
```

```
            int leapOff = getLeapMonthOffset(a11, timeZone);
            int leapMonth = leapOff - 2;
            if (leapMonth < 0) {
                leapMonth += 12;
            }
            if (lunarLeap != 0 && lunarMonth != leapMonth) {
                System.out.println("Invalid input!");
                return new int[]{0, 0, 0};
            } else if (lunarLeap != 0 || off >= leapOff) {
                off += 1;
            }
        }
        int monthStart = getNewMoonDay(k+off, timeZone);
        return jdToDate(monthStart+lunarDay-1);
    }
}
```