# PHÂN TÍCH YÊU CẦU

Yêu cầu:

* Giao diện đồ họa(Swing)
* Chỉ cho phép đặt vé cho các suất chiếu trong 1 ngày
* Có thể lựa chọn sử dụng kết nối TCP hoặc UDP
* Tự tổ chức cách lưu trữ dữ liệu(không quan trọng về vấn đề bảo mật) và KHÔNG ĐƯỢC PHÉP sử dụng database
* Cho phép nhiều client truy cập vào server và đặt vé cùng lúc
* Một số chức năng của server
  + Có giao diện
  + Cho phép cấu hình các suất chiếu trong ngày(7-9h; 9:30-11:30h;… )
  + Cấu hình rạp chiếu phim(có bao nhiêu khu/khán đài chiếu phim, có bao nhiêu hàng, bao nhiêu ghế trên một hàng, giá vé trên từng khu khác nhau thế nào)
  + Xem tình trạng đặt chổ(các chổ còn trống, chổ đã đặt)
  + Nhiều client truy cập vào server cùng lần để xem và đặt vé
* Một số chức năng của user
  + Có giao diện
  + Chức năng kết nối tới server
  + Hiển thị các thông tin lấy được từ server(các suất chiếu; các khu vực/khán đài; tình trạng của một chổ ngồi)
  + Đặt vé cho một chổ ngồi còn trống

\*\* về phân chia khu vực/khán đài sẽ phân thành(ghế thường, ghế vip và ghế đôi)

\*\* phân biệt về màu sắc ghế đã đặt, ghế đang chọn và ghế còn trống

# MỘT SỐ NỘI DUNG DÙNG TRONG ĐỒ ÁN

3 trạng thái của một thread

* New: một luồng ở trạng thái này khi nó nhận được thời gian chờ của CPU
* Running : dấu hiệu của một luồng đang được thực thi
* Suspended : một luồng ở trạng thái suspended khi nó tạm thời không hoạt động hoặc đang đợi được thực thi
* Blocked : một luồng ở trạng thái blocked khi nó đang chờ tài nguyên để tiếp tục
* Terminated: nếu một luồng nhận được đấu hiệu ở trạng thái này dù bất cứ thời điểm nào thì nó sẽ dừng thực thi ngay lập tức

Cách cách để tạo một thread

* Kế thừa từ lớp **Thread**
* Implement interface **Runnable**
* Đa phần người ta sẽ chọn sử dụng implement interface Runnable bởi vì nó không yêu cầu người dùng phải tạo ra một lớp kế thừa từ Thread. Đặc biệt khi thiết kế phần mềm sử dụng đa kế thừa thì interface là lựa chọn duy nhất

Một số phương thức của lớp Thread thường được sử dụng

**suspend()** : phương thức này tạm dừng hoạt động của một luồng nào đó bằng cách ngừng cung cấp CPU của thread này. Muốn cung cấp lại CPU thì sử dụng phương thức resume(). Khi gọi phương thức suspend() thì luồng sẽ không dừng ngay lập tức mà chỉ dừng sau khi mà luồng đó trả tài nguyên về cho CPU

**resume()** :

**stop()** : kết thúc phương thức run() bằng cách ném ra một ngoại lệ ThreadDeath để kết thúc luồng 1 cách ép buộc. Trường hợp gọi stop cho một luồng khi nó đang nắm 1 tài nguyên mà luồng khác đang chờ thì sẽ xảy ra deadlock

**destroy()** : dừng hẳn 1 luồng

**isAlive()** : Kiểm tra xem một luồng có còn hoạt động hay không. Trả về true nếu luồng đã được start() và chưa bị rơi vào deadlock. Trả về false nếu luồng đang ở trạng thái new thread hoặc dead

**yield()** : khi gọi phương thức này thì một luồng sẽ bị ngừng cấp CPU và nhường cho luồng tiếp theo trong hàng chờ ready. Khác với suspend() thì luồng này sẽ chỉ bị ngừng cấp CPU trong đợt nay và vẫn sẽ được cấp trở lại trong đợt tiếp theo

**sleep(long)** : tạm ngừng luồng trong 1 khoảng thời gian

**join()** : thông báo rằng hãy chờ thread này hoàn thành rồi thread cha mới được tiếp tục chạy

**join(long)** : thread cha cần phải đợi bao lâu đó mới được phép tiếp tục chạy kể từ lúc gọi join() . nếu tham số trong phương thức này bằng 0 tức là đợi cho tới khi luồng này kết thúc thì tiếp tục chạy

**getName(), setName(String)** : trả về tên thread, thay đổi tên Thread

**getId()** : trả về Id của thread

**getState()** : trả về trạng thái hiện tại của thread

**currentThread()** : trả về tham chiếu của thread đang được thi hành

**getPriority()** : trả về độ ưu tiên của thread

**setPriority(int)** : set độ ưu tiên cho một thread

NET WORKING

Quá trình kết nối giữa máy chủ và máy khách

* Máy chủ khởi tạo một đối tượng ServerSocket với một cổng giao tiếp port
* Máy chủ gọi phương thức accept() của lớp ServerSocket. Phương thức này sẽ đợi cho tới khi có một máy khách kết nối đến máy chủ trên cổng đã cho
* Trong khi máy chủ đang chờ đợi, máy khách sẽ tạo một đối tượng Socket, xác định tên máy chủ(IP hoặc domain đều hợp lệ) và số cổng để kết nối
* Đối tượng Socket của máy khách sẽ cố gắng kết nối tới máy chủ với tên và số cổng đã chỉ định. Nếu thiết lập thành công, máy khách bây giờ đã có được một đối tượng Socket và đối tượng này có khả năng kết nối vưới máy chủ
* Ở phía máy chủ, phương thức accept() trả về một tham chiếu đến một socket mới trên máy chủ mà được kết nối với máy khách
* Sau khi đã đượck kết nối thì thông tin sẽ được truyền và nhận thông qua OutputStream và InputStream

Một số phương thức constructor của lớp ServerSocket

|  |  |
| --- | --- |
| Tên phương thức | Mô tả |
| Public ServerSocket(int port) throws IOException | Cố gắng tạo một socket được ràng buộc vào một port chỉ định. Ngoại lệ xảy ra khi port đã bị ràng buộc bởi một ứng dụng khác |
| Public ServerSocket(int port, int backlog) throws IOException | Tương tự với phương thức ở trên. Tham số backlog xác định xem có bao nhiêu máy khách đến để lưu trữ trong một hàng đợi |
| Public ServerSocket(int port, int backlog, InetAddress address) throws IOException | Tham số InetAddress chỉ định địa chỉ IP cục bộ để ràng buộc. Được sử dụng cho các máy chủ có thể có nhiều địa chỉ IP, cho phép máy chủ xác định xem IP nào có thể chấp nhận yêu cầu của máy khách |
| Public ServerSocket() throws IOException | Tạo ra một ServerSocket không kết nối. Khi sử dụng constructor này, sử dụng thêm phương thức **bind()** khi bạn đã sẵn sàng để ràng buộc socket của máy chủ |

MỘT SỐ PHƯƠNG THỨC KHÁC

|  |  |
| --- | --- |
| Tên phương thức | Mô tả |
| Public int getLocalPort() | Trả về cổng Socket của máy chủ lắng nghe. Phương thức này khá là hữu ích khi bạn truyền 0 là port cho một constructor và để cho máy chủ tìm thấy một cổng cho bạn |
| Public Socket accept() throw IOException | Chờ máy khách đến. Phương thức này ngăn chặn cho đến khi một máy trạm kết nối tới máy chủ trên cổng được chỉ định hoặc socket timeout, với thời gian được thiết lập bằng phương thức setSoTimeout(). Nếu không phương thức này sẽ bị xóa vô thời hạn |
| Public void setSoTimeout() | Thiết lập giá trị thời gian chờ cho bao lâu socket của máy chủ chờ khách hàng trong suốt quá trình chấp nhận – accept |
| Public void bind(SocketAddress host, int backlog) | Liên kết socket này với máy chủ và cổng được chỉ định trong đối tượng SocketAddress |

* Khi một server gọi accept thì phương thức này sẽ không phản hồi cho tới khi một client nào đó kết nối đến. Sau khi có client kết nối, ServerSocket tạo một Socket mới trên một cổng không xác định và trả về một tham chiếu đến Socket mới này

LỚP SOCKET là đại diện cho Socket mà cả máy khách và máy chủ sử dụng để liên lạc qua lại với nhau. Máy khách sẽ nhận 1 đối tượng Socket bằng cách khởi tạo trong khi máy chủ sẽ nhận 1 đối tượng Socket từ phương thức accept()

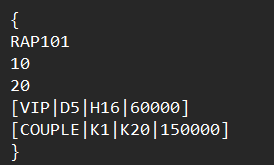
Một số phương thức constructor của lớp Socket

|  |  |
| --- | --- |
| Tên phương thức | Mô tả |
| Public Socket(String host, int port) throws UnknowHostException, IOException | Cố gắng tạo và kết nối tới máy chủ với cổng chỉ định. Nếu thành công máy khách và máy chủ sẽ kết nối với nhau |
| Public Socket(String host, int port) throws UnknowHostException, IOException | Giống hoàn toàn với phương thức trên ngoại trừ việc máy chủ được biểu hiện bởi một đối tượng InetAddress |
| Public Socket(String host, int port, InetAddress, int localPort) throws IOException | Kết nối đến máy chủ với cổng được chỉ định, tạo một socket trên máy chủ cục bộ tại địa chỉ và cổng được chỉ định |
| Public Socket(InetAddress host, int port, InetAddress localAddress, int localPort) throws IOException | Giống với phương thức ở trên trừ việc máy chủ được biểu hiện bởi 1 InetAddress thay vì một String |
| Public Socket() | Tạo một socket không liên lạc. Sau này khi có nhu cầu muốn liên lạc sẽ sử dụng phương thức connect() để kết nối tới máy chủ. |

# CẤU TRÚC DỮ LIỆU

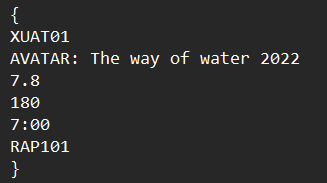
1. File lưu thông tin rạp phim

* Id rạp
* Số hàng
* Số cột
* Phân loại các khu



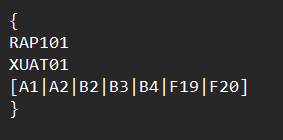
1. File lưu thông xuất chiếu

* Id xuất chiếu
* Tên film
* Rating
* Thời lượng
* Thời gian bắt đầu chiếu
* Rạp chiếu(id rạp)



1. File lưu tình trạng đặt vé của một suất chiếu

* Id xuất chiếu
* Id rạp chiếu
* Mảng 2 chiều lưu tình trạng đặt vé



Advanced :

1. Thông tin vé được đặt

* Id xuất chiếu
* Vị trí ngồi
* Mã đặt vé
* Khi người dùng khởi động ứng dụng và kết nối thành công tới server thì server sẽ gửi tới cho người dùng toàn bộ các thông tin xuất chiếu
* Khi người dùng chọn 1 xuất chiếu(gửi request tới server để server gửi lại toàn bộ nội dung về thông tin phim về đặt vé)
* Khi người dùng điền thông tin và đặt vé (gửi reuquest tới server cập nhật file lưu tình trạng đặt vé của một suất chiếu(file 3) + thêm một file lưu thông tin vé được đặt(advanced)