**JAVA CORE BASIC**

1. Kỹ thuật lập trình căn bản
2. Download và cài đặt công cụ lập trình (Nếu là java thì dùng eclipse, hoặc myeclipse, hoặc net bean)
3. Các kiểu dữ liệu

* Nguyên thủy: byte, double, int, float, short, long, double, boolean, char
* Không nguyên thủy: String, Array, classes, interface

1. Các câu lệnh if, for, while, do-while, case switch… Cho ví dụ minh họa, sự khác nhau.

* If: điều kiện (if…else)

VD: if (x<1) {

System.out.println(x)

} else {

x += 1

System.out.println(x)}

* For: tạo vòng lặp

VD: for (x=0, x<10,x++) {

}

* While (điều kiện) {

}

* Do {

} while (điều kiện)

* Switch: tính năng giống if – else

VD:

String day = “Monday”;

Switch (day) = {

Case “Monday”: System.out.println(“it is Monday”);

Break;

Case “Tuesday”: System.out.println(“it is tuesday”);

Break;

Default: System.out.println(“it is not a day”);

}

1. Một số giải thuật làm việc với mảng: sắp xếp (các loại sắp xếp cơ bản), thêm, xóa 1 phần tử trong mảng, …

* Sắp xếp: kiểu nổi bọt, sắp xếp chọn, sắp xếp chèn, sắp xếp trộn, sắp xếp nhanh, shell sort

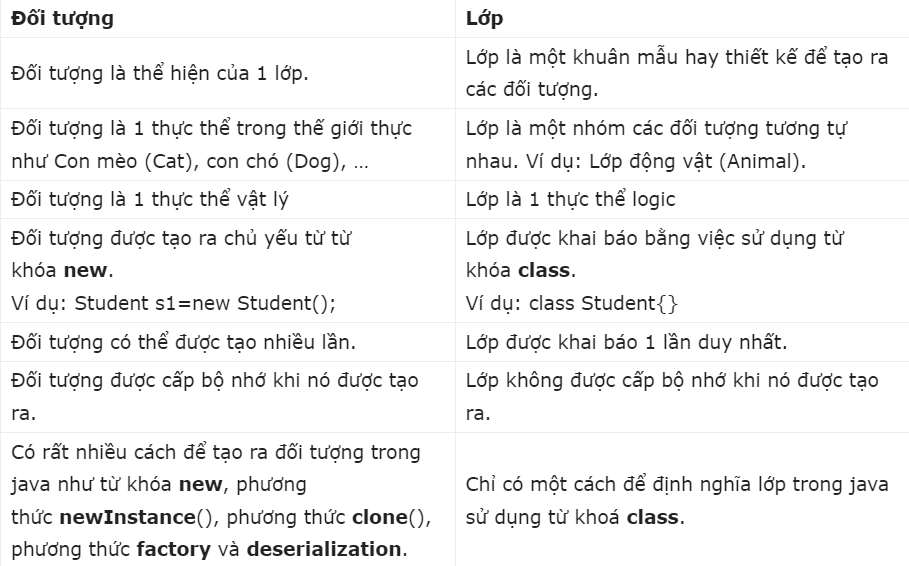
1. Streams Input/Output, làm việc với file mức cơ bản.

* Stream output: System.out.print(); hoặc : System.out.println();
* Stream input: Scanner scanner = new Scanner(System.in);

String name = scanner.nextLine();

int name = scanner.nextInt();

1. Lập trình hướng đối tượng (phần này cần chi tiết và rõ ràng, có ví dụ cụ thể, làm thành tài liệu)
2. Các khái niệm: Đối tượng là gì? Class là gì?

* Đối tượng(object): Đối tượng là một thực thể có trạng thái và hành vi. Nó có thể mang tính vật lý hoặc logic. Nếu chúng ta xem xét thực tế chúng ta có thể tìm thấy nhiều đồ vật xung quanh chúng ta: cái bàn, con chó, con người, v.v… Tất cả các đối tượng này đều có thuộc tính và hành vi. Nếu chúng ta xem xét một con chó, thuộc tính của nó sẽ là – tên, giống, màu sắc, và các hành vi là: sủa, chạy, ăn, … Nếu bạn so sánh các đối tượng trong phần mềm với một đối tượng trong thế giới thực, chúng sẽ có đặc điểm rất giống nhau: thuộc tính đối tượng trong phần mềm được lưu trữ trong trường (field) và hành vi được lưu trữ trong phương thức (method).
* Class: Chúng ta có thể xem lớp như một khuôn mẫu (template) của đối tượng (Object). Trong đó bao gồm dữ liệu của đối tượng (fields hay properties) và các phương thức(methods) tác động lên thành phần dữ liệu đó gọi là các phương thức của lớp.
* 

1. Các đặc điểm: (đa hình, thừa kế …), So sánh overloading và overriding, method overriding, abstract class, interface …

* Trừu tượng: Mọi thứ hoạt động trong class như 1 hộp đen, khi sử dụng đối tượng của class chúng ta chỉ thấy được đầu vào và đầu ra, còn lại xử lý bên trong như thế nào chúng ta k quan tâm
* Đóng gói: chúng ta sẽ đóng gói tất cả thuộc tính và hành vi của đối tượng thành 1 class. Với class này chúng ta chỉ cho thấy và sử dụng những gì chúng ta muốn cho thấy, từ đó bảo vệ được thực thể của đối tượng trước những thay đổi không mong muốn trực tiếp từ bên ngoài
* Kế thừa: giúp đối tượng tiến hóa thành những đối tượng chuyên biệt hơn nhưng không cần viết lại toàn bộ đối tượng chuyên biệt đó
* Đa hình:

+ overriding: chúng ta sẽ ghi đè hành vi của đối tượng chuyên biệt lên hành vi gốc của đối tượng được kế thừa

+ overloading: chúng ta sẽ định nghĩa nhiều hành vi cùng tên nhưng sẽ có các đầu vào khác nhau, dựa vào sự khác nhau của đầu vào mà đối tượng sẽ chọn đúng hành vi để xử lý tiếp

1. Cách khai báo lớp, thuộc tính, phương thức, constructor, cách phân chia các class?

-Cách khai báo lớp

<Phạm vi truy cập> class <Tên lớp> {     // Khai báo thuộc tính (field) hay biến     <Phạm vi truy cập> <kiểu dữ liệu> <field\_1>;     <Phạm vi truy cập> <kiểu dữ liệu> <field\_2>;     // Hàm khởi tạo không có tham số     <Phạm vi truy cập> <Tên lớp>(){         // Khởi tạo dữ liệu     }     // Hàm khởi tạo có tham số     <Phạm vi truy cập> <Tên lớp>(<Kiểu dữ liệu> <tên biến>[, <Kiểu dữ liệu> <tên biến>]){         // Khởi tạo dữ liệu     }     // Phương thức xử lý (method)     <Kiểu trả về> method\_1() {         // Logic xử lý     }     <Phạm vi truy cập> <Kiểu trả về> method\_2(<Kiểu dữ liệu> <tên biến>) {         // Logic xử lý     } }

Trong đó:

* class: là từ khóa để khai báo lớp trong Java.
* Tên lớp: là tên chúng ta đặt cho lớp (quy tắc đặt tên: viết hoa mỗi chữ cái đầu).
* field\_1, field\_2: các thuộc tính, các biến, hay các thành phần dữ liệu của lớp (quy tắc đặt tên: chữ cái đầu tiên viết thường, các chữ cái đầu tiên tiếp theo viết hoa).
* Hàm khởi tạo (constructor): hàm dùng để khởi tạo đối tượng lớp.
* method\_1, method\_2: là các phương thức thể hiện các thao tác xử lý, tác động lên các thành phần dữ liệu của lớp.

Thuộc tính: Để khai báo một thuộc tính trong class các bạn sử dụng cú pháp sau:

assess\_modifier option property\_type name;

Trong đó:

* assess\_modifier là phạm vi truy cập của thuộc tính. assess\_modifier sẽ là 1 trong 4 giá trị sau: public , protected , private , default hoặc bỏ trống
* option là loại của thuộc tính. option có thể là final hoặc static hoặc bỏ trống.
* property\_type là kiểu dữ liệu của thuộc tính.
* name là tên của thuộc tính.

Ví dụ: tạo một class có tên là Main và có các thuộc tính x và y

**public** **class** Main **{**

**int** x = 5;

**int** y = 3;

**}**

Phương thức: Phần khai báo phương thức tên bạn thấy ngoài tên phương thức là sayHello và code của phương thức trong khối {...} thì bạn còn thấy:

* **Phương thức tĩnh trong java:** là phương thức được khái báo có thêm từ khóa **static**. Nếu là phương thức tĩnh thì có thể gọi nó mà không cần tạo ra đối tượng của lớp chứa phương thức.
* **Phạm vi truy cập của phương thức:** xác định bằng các từ khóa như public, protected ... Xem thêm [phạm vi truy cập (Access modifier) Java](https://xuanthulab.net/pham-vi-truy-cap-phuong-thuc-khoi-tao-doi-tuong-java.html)
* **Kiểu dữ liệu trả về của phương thức**: Các phương thức cần chỉ rõ nó trả về dữ liệu kiểu như **int, String, double ...**. Từ khóa **void** cho biết phương thức này khi kết thúc không trả về dữ liệu gì.

Constructor trong java là một dạng đặc biệt của phương thức được sử dụng để khởi tạo các đối tượng.

Java Constructor được gọi tại thời điểm tạo đối tượng. Nó khởi tạo các giá trị để cung cấp dữ liệu cho các đối tượng, đó là lý do tại sao nó được gọi là constructor.

Khai báo của Constructor giống với khải báo của method (phương thức). Nó phải có **cùng tên với class** (lớp) và **không có giá trị trả về**.

Có 2 kiểu của constructor:

* Constructor mặc định (không có tham số truyền vào – **default constructors**).
* Constructor tham số (**parameterized constructors**).

1. Cho ví dụ minh họa chi tiết
2. Exception handling
3. Khai báo ngoại lệ
4. Cơ chế try-catch
5. Throw ngoại lệ
6. Chuyển tiếp ngoại lệ
7. Lập trình xử lý với database



1. Cài đặt oracle (10G or 11G)
2. Sử dụng oracle: các câu lệnh cơ bản, table, store procedure, function, trigger, sequence, partition, job, schedule…

- Các function:

<https://viblo.asia/p/cac-function-trong-mysql-phan-1-yMnKMnODZ7P>

<https://viblo.asia/p/cac-function-trong-mysql-phan-2-GrLZDvD35k0>

- Update & delete

UPDATE employees

SET hourly\_pay = 10.25, (cập nhật dữ liệu)

hire\_date = NULL

WHERE id = 6;

SELECT \* FROM employees;

DELETE FROM employees;

SELECT \* FROM employees; // lệnh sẽ xóa toàn bộ bảng

DELETE FROM employees

WHERE id = 6; //xóa toàn bộ dữ liệu của hang có thông tin id=6

SELECT \* FROM employees;

- Trích xuất dữ liệu:

SELECT \* FROM employees;

SELECT first\_name, last\_name

FROM employees;

SELECT \* FROM employees

WHERE last\_name = “Nguyen”;

Hoặc

WHERE hourly\_pay <=20;

Hoặc

WHERE hire\_date IS NOT NULL;

- Chèn dữ liệu mới vào cơ sở dữ liệu:

INSERT INTO employees

VALUES (1, “Tuan Kiet”, “Nguyen”, 25, “2023-01-02”);

SELECT \* FROM employees;

**Hoặc**

INSERT INTO employees

VALUES (1, “Tuan Kiet”, “Nguyen”, 25, “2023-01-02”),

(2, “Thiet”, “Pham”, 20, “2023-01-03”),

(3, “Spongebob”, “Squarepants”, 12, “2023-01-15”),

(4, “Patrick”, “Star”, 14, “2023-01-12”);

SELECT \* FROM employees;

**Hoặc**

INSERT INTO employees (id, first\_name, last\_name)

VALUES (1, “Tuan Kiet”, “Nguyen”);

SELECT \* FROM employees;

-TABLE: tạo bảng

CREATE TABLE employees (

Id INT,

First\_name VARCHAR(50),

Last\_name VARCHAR(50),

Hourly\_pay DECIMAL(5,2),

Hire\_date DATE

);

Trích xuất bảng:

SELECT \* FROM employees;

Đổi tên bảng:

RENAME TABLE employees TO worker;

Xóa bảng:

DROP TABLE worker;

Sửa đổi bảng:

ALTER TABLE worker

ADD phone\_number VARCHAR(15);

**Hoặc**

ALTER TABLE worker

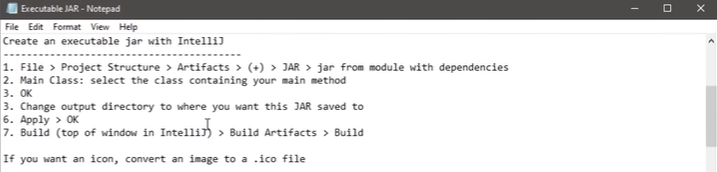
RENAME COLUMN phone\_number TO email;

MODIFY COLUMN email VARCHAR(100);

1. Kết nối với oracle, tự làm bài tập để test
2. Học cách làm quen và sử dụng file config trong các module: (sửa dụng 14, hoặc ở mức cơ bản có thể tự viết đọc file text)

<https://viblo.asia/p/java-vi-du-ve-thao-tac-voi-properties-file-gDVK2kGmZLj>

1. Đóng gói chương trình, run file trên windows, linux …



1. **Bài tập (*Về phần bài tập các tham số được lưu trữ trong các file cấu hình – cho phép truyền vào mà không cần build lại code, Viết trên console*)**
2. Viết chương trình nhập 1 số cho tới khi nhập đúng số cần nhập thì hiện thông báo thành công. Số lần nhập tối đa 5 lần nếu nhập sai thì sẽ dừng chương trình và hiện thông báo nhập lỗi.
3. Viết chương trình tính tiền điện, cho phép nhập số điện đã dùng trong tháng, giả thiết 100 số đầu tính giá 1000đ/1 số, kể từ 100 số tính giá 1500đ/số, kể từ 150 số tính giá 2000đ/số.
4. Sắp xếp mảng số nguyên 1 chiều gồm n phần tử cho trước được đọc ra từ file input.txt bằng phương pháp quick sort, giả thiết các số nguyên cách nhau bằng dấu cách hoặc xuống dòng
5. Viết chương trình console, thực hiện nhập thông tin n sinh viên (tên, giới tính, quê quán, tuổi) Sau khi nhập xong bấm enter để insert thông tin vào DB, giả thiết tên không được trùng nhau, id tự tăng. (primary key = id, unique = tên)

https://www.codejava.net/java-se/jdbc/jdbc-tutorial-sql-insert-select-update-and-delete-examples#ExecuteINSERT

**JAVA CORE ADVANCE**

1. Streams Input/output, làm việc với file mức nâng cao (tự nghiên cứu sử dụng các thư viện nâng cao trong Java), ưu nhược điểm.
2. Thread (chi tiết, thực hành ngay nếu học xong lý thuyết)

* Khái niệm

*Một Thread (luồng) là một hướng đi hay một đường dẫn đã được thực hiện, khi chương trình được thực hiện khi mà chương trình được thực thi. Nói chung, tất cả các chương trình có ít nhất một luồng, được biết như là luồng chính, nó được cung cấp bởi JVM (Java Vitural Machine) vào lúc bất đầu thực hiện chương trình.*

*Đa luồng (multithreading) trong java là một tiến trình thực hiện nhiều luồng đồng thời. Luồng (thread) về cơ bản là một tiến trình con (sub-process). Nó là đơn vị nhỏ nhất của tiến trình. Đa tiến trình (multiprocessing) và đa luồng (multithreading) cả hai được sử dụng để tạo ra hệ thống đa nhiệm (multitasking).*

* Khởi tạo, có mấy cách khởi tạo, ưu nhược điểm

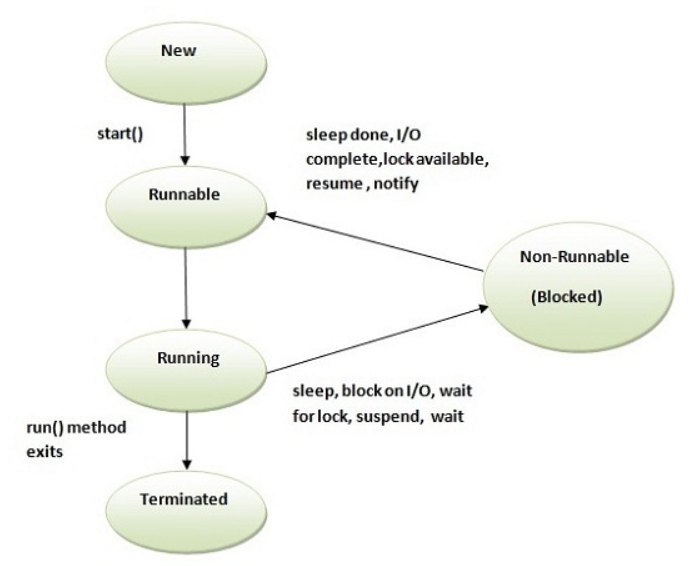
*2 cách khởi tạo thread hoặc multithread: dùng class thread hoặc subclass (cách 1) và dùng interface Runable*

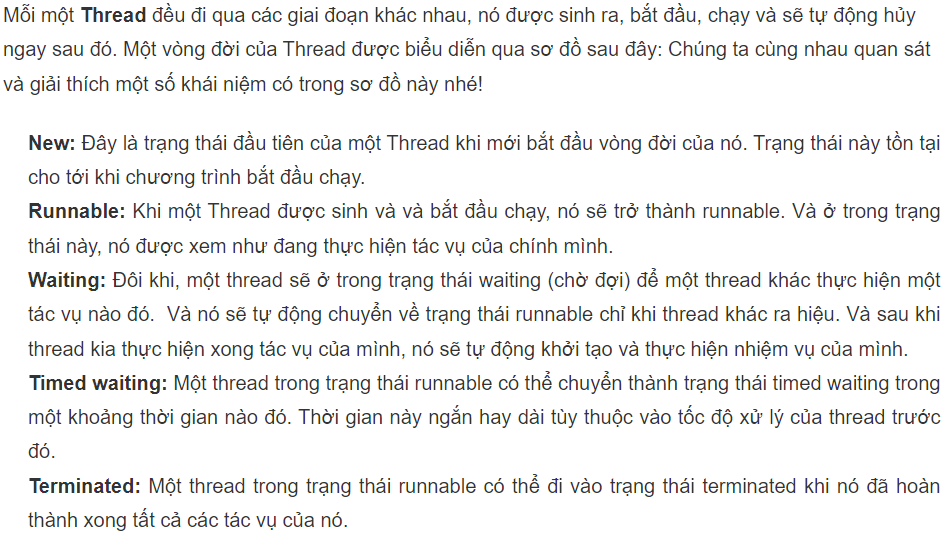
*Thực tế không có quy ước nào nói rằng phương pháp nào tốt nhất để khởi tạo và chạy****Thread****. Tuy nhiên việc sử dụng****Interface Runnable****có một cái lợi là giúp ta kế thừa từ 1 lớp khác ngoài lớp thread, vì nếu sử dụng Subclass thì chỉ có thể thừa kế một lớp****Thread****vì java không hỗ trợ đa thừa kế.*

* Thread Priority

Thứ tự ưu tiên của thread cho phép thread nào chạy trước

* Vòng đời của 1 thread, các trạng thái





* Dealock, khái niệm, ví dụ

***Deadlock trong java****hay bế tắc trong java là một phần của đa luồng. Deadlock có thể xảy ra trong trường hợp một luồng đang đợi một khóa đối tượng (object lock), được sử dụng bởi một luồng khác và luồng thứ hai đang chờ đợi một khóa đối tượng (object lock) được sử dụng bởi luồng thứ nhất. Vì cả hai luồng đang chờ đợi nhau để giải phóng khóa, tình trạng này được gọi là bế tắc (deadlock).*

* Timer, Time Task, Schedule … Khái niệm, cho ví dụ

1. Lập trình mạng
2. Socket: Lập trình socket cho phép bạn tạo, kết nối và truyền dữ liệu qua mạng giữa các thiết bị. Bạn có thể tạo ứng dụng máy khách và máy chủ sử dụng các giao thức như TCP hoặc UDP.
3. Tcp: khái niệm, cơ chế, ví dụ

*TCP (Transmission Control Protocol) là một trong hai giao thức chính được sử dụng trong mô hình TCP/IP (Transmission Control Protocol/Internet Protocol) để truyền dữ liệu qua mạng. Nó cung cấp một kênh truyền tin cậy và đảm bảo việc truyền dữ liệu giữa các máy tính trên mạng.*

*Cơ chế của TCP:*

1. *Thiết lập kết nối: Trước khi truyền dữ liệu, hai máy tính cần thiết lập một kết nối TCP. Quá trình này được gọi là "ba bước bắt tay" (three-way handshake). Bước đầu tiên, máy tính gửi một gói tin SYN (synchronize) đến máy tính đích. Máy tính đích sau đó gửi gói tin SYN-ACK (synchronize-acknowledgment) trở lại. Cuối cùng, máy tính gửi một gói tin ACK (acknowledgment) để xác nhận việc thiết lập kết nối. Sau quá trình này, kết nối TCP được thiết lập và sẵn sàng truyền dữ liệu.*
2. *Truyền dữ liệu và quản lý luồng: TCP chia dữ liệu thành các đoạn (segments) và truyền chúng qua mạng. Nó sử dụng cơ chế kiểm soát luồng (flow control) để đảm bảo rằng máy tính nhận dữ liệu mà nó có thể xử lý. TCP sử dụng cơ chế xác nhận (acknowledgment) để đảm bảo rằng các đoạn dữ liệu đã được nhận đúng thứ tự và không bị mất.*
3. *Quản lý độ tin cậy: TCP đảm bảo việc truyền dữ liệu tin cậy. Nếu một đoạn dữ liệu bị mất hoặc bị hỏng trong quá trình truyền, TCP sẽ yêu cầu máy tính gửi lại đoạn dữ liệu đó. Nếu một gói tin không được xác nhận sau một khoảng thời gian, TCP sẽ gửi lại gói tin đó..*
4. udp: khái niệm, cơ chế, ví dụ

*UDP (User Datagram Protocol) là một giao thức truyền dữ liệu không đảm bảo độ tin cậy trong mô hình TCP/IP. UDP cho phép truyền dữ liệu nhanh chóng và hiệu quả, nhưng không đảm bảo việc dữ liệu được gửi và nhận một cách chính xác và tin cậy như TCP.*

*Cơ chế của UDP:*

*1. Gửi dữ liệu: Khi sử dụng UDP, dữ liệu được chia thành các gói tin (datagrams) riêng lẻ và gửi đi mà không cần thiết lập kết nối trước như TCP. Mỗi gói tin UDP chứa đủ thông tin để xác định địa chỉ nguồn và đích cùng với dữ liệu gửi đi.*

*2. Không có xác nhận: UDP không sử dụng cơ chế xác nhận hoặc gửi lại dữ liệu bị mất. Điều này có nghĩa là nếu một gói tin UDP bị mất hoặc hỏng trong quá trình truyền, nó sẽ không được gửi lại và ứng dụng sẽ không nhận được thông báo về việc mất gói tin.*

*3. Không có kiểm soát luồng: UDP không có cơ chế kiểm soát luồng để điều chỉnh tốc độ truyền dữ liệu. Điều này có thể dẫn đến tình trạng quá tải hoặc mất mát dữ liệu nếu nguồn dữ liệu gửi quá nhanh hoặc đích không thể xử lý dữ liệu đến với tốc độ tương ứng.*

1. So sánh tcp và udp

1. Độ tin cậy: TCP đảm bảo truyền dữ liệu tin cậy. Nó sử dụng cơ chế xác nhận và gửi lại dữ liệu bị mất để đảm bảo rằng dữ liệu được gửi và nhận một cách chính xác. Trong khi đó, UDP không đảm bảo độ tin cậy. Nó không sử dụng cơ chế xác nhận và không gửi lại dữ liệu bị mất.

2. Cơ chế kết nối: TCP sử dụng cơ chế thiết lập kết nối trước khi truyền dữ liệu. Quá trình ba bước bắt tay (three-way handshake) được sử dụng để thiết lập kết nối. Ngược lại, UDP không yêu cầu thiết lập kết nối trước khi truyền dữ liệu. Nó cho phép gửi và nhận dữ liệu mà không cần thiết lập kết nối trước đó.

3. Kiểm soát luồng: TCP sử dụng cơ chế kiểm soát luồng để điều chỉnh tốc độ truyền dữ liệu giữa nguồn và đích. Điều này đảm bảo rằng nguồn gửi dữ liệu không quá tải đích và đảm bảo rằng dữ liệu không bị mất do quá tải. UDP không có cơ chế kiểm soát luồng, điều này có thể dẫn đến tình trạng quá tải hoặc mất mát dữ liệu nếu nguồn gửi quá nhanh hoặc đích không thể xử lý dữ liệu đến với tốc độ tương ứng.

4. Hiệu suất: UDP có hiệu suất cao hơn so với TCP. Do UDP không có các cơ chế như xác nhận, gửi lại hay kiểm soát luồng, nó có thể truyền dữ liệu nhanh chóng và hiệu quả hơn. Tuy nhiên, TCP đảm bảo độ tin cậy và chính xác, do đó phù hợp cho các ứng dụng yêu cầu tính tin cậy cao như truyền file, truyền dữ liệu quan trọng.

5. Ứng dụng: TCP thích hợp cho các ứng dụng như truyền file, trang web, email và các ứng dụng yêu cầu tính tin cậy và chính xác. UDP thích hợp cho các ứng dụng như streaming media, video call, trò chơi trực tuyến, nơi tốc độ truyền dữ liệu và phản hồi nhanh là quan trọng hơn độ tin cậy và kiểm soát luồng.

Tùy thuộc vào yêu cầu cụ thể của ứng dụng, lựa chọn giữa TCP và UDP sẽ phụ thuộc vào yêu cầu về độ tin cậy, tốc độ và tính ứng dụng của hệ thống.

1. MultiThread
2. Khái niệm

MultiThread là một khái niệm trong lập trình đa luồng (multithreading), cho phép chương trình thực hiện nhiều công việc đồng thời trong cùng một thời điểm. Mỗi thread đại diện cho một luồng thực thi riêng biệt, có thể chạy đồng thời với các thread khác. Sử dụng multithreading giúp tận dụng tài nguyên hệ thống hiệu quả, tăng cường hiệu năng và cải thiện trải nghiệm người dùng trong các ứng dụng đa nhiệm.

1. Tìm hiểu Queue là gì? các loại? cách làm việc với queue

Queue là một cấu trúc dữ liệu trong lập trình, tuân thủ nguyên tắc "First-In-First-Out" (FIFO). Nó cho phép chúng ta thêm phần tử vào cuối hàng đợi (enqueue) và lấy phần tử từ đầu hàng đợi (dequeue). Trong Java, Queue là một interface trong gói java.util và có những lớp cụ thể như LinkedList và PriorityQueue để triển khai.

Có hai loại chính của Queue trong Java:

1. Queue không giới hạn kích thước: Ví dụ LinkedList, cho phép chứa số lượng phần tử không giới hạn. Khi enqueue, phần tử được thêm vào cuối danh sách. Khi dequeue, phần tử được lấy ra từ đầu danh sách.

2. Queue có giới hạn kích thước: Ví dụ ArrayBlockingQueue, có kích thước tối đa xác định trước và chỉ chứa số lượng phần tử tối đa đó. Khi enqueue và danh sách đã đầy, quá trình thêm phần tử mới sẽ bị chặn (block) cho đến khi có vị trí trống.

Cách làm việc với Queue trong Java:

Để sử dụng Queue, trước tiên bạn cần import gói java.util và khởi tạo một đối tượng Queue từ một lớp triển khai cụ thể.

Sử dụng phương thức enqueue để thêm phần tử vào cuối hàng đợi.

Sử dụng phương thức dequeue để lấy phần tử từ đầu hàng đợi.

Có thể sử dụng các phương thức khác như peek để xem phần tử đầu tiên mà không xóa nó, size để lấy số lượng phần tử trong hàng đợi, và isEmpty để kiểm tra xem hàng đợi có rỗng hay không.

1. Tìm hiểu và làm ví dụ với Thread Pool: Khái niệm, cách sử dụng, ưu nhược điểm …

Thread Pool là một cơ chế quản lý các thread trong lập trình đa luồng. Nó giới hạn số lượng thread được tạo ra và sử dụng lại các thread đã hoàn thành công việc trước đó. Điều này giúp tránh tạo ra quá nhiều thread và giảm tải cho hệ thống.

Cách sử dụng Thread Pool trong Java:

Trước tiên, import gói java.util.concurrent để sử dụng ThreadPoolExecutor.

Khởi tạo một ThreadPoolExecutor với số lượng thread tối đa và các thiết lập khác như thời gian sống của các thread không hoạt động (idle thread).

Để thực thi một nhiệm vụ, sử dụng phương thức execute hoặc submit trên ThreadPoolExecutor và truyền một Runnable hoặc Callable object.

ThreadPoolExecutor sẽ quản lý việc chia sẻ và sử dụng lại các thread trong pool để thực thi các nhiệm vụ.

- Ưu điểm của Thread Pool:

Giảm tải hệ thống: Thread Poolgiúp giới hạn số lượng thread được tạo ra, tránh tạo ra quá nhiều thread và làm tăng tải hệ thống.

Tái sử dụng thread: Các thread đã hoàn thành công việc được sử dụng lại cho các nhiệm vụ mới, giảm thời gian tạo và hủy thread.

Điều chỉnh quy mô: Có thể điều chỉnh số lượng thread trong pool để phù hợp với yêu cầu và tải công việc.

Quản lý và giám sát: Thread Pool cung cấp các phương thức để quản lý và giám sát các thread, như kiểm tra trạng thái của pool, theo dõi tiến trình thực thi và xử lý các ngoại lệ.

- Nhược điểm của Thread Pool:

Đòi hỏi đánh đổi: Sử dụng Thread Pool đòi hỏi một số đánh đổi như tiêu tốn bộ nhớ để lưu trữ các thread trong pool và việc phân phối công việc giữa các thread.

Khó kiểm soát: Khi sử dụng Thread Pool, không thể kiểm soát trực tiếp các thread cụ thể, vì chúng được quản lý bởi ThreadPoolExecutor.

1. Thế nào là đồng bộ và bất đồng bộ, làm thế nào để đồng bộ khi nhiều thread cùng sử dụng 1 tài nguyên? Các vấn đề cần xử lý khi chương trình có nhiều thread?... (Cần đọc nhiều về multithreading)

Đồng bộ (synchronization): Đồng bộ hóa trong multithreading đảm bảo rằng các thread không can thiệp vào nhau khi truy cập và sửa đổi các tài nguyên chia sẻ. Điều này đảm bảo tính nhất quán và độ tin cậy của dữ liệu. Trong Java, từ khóa synchronized và các phương thức đồng bộ hóa (synchronized methods) được sử dụng để đảm bảo sự đồng bộ giữa các thread.

Bất đồng bộ (asynchronization): Bất đồng bộ trong multithreading cho phép các thread thực thi độc lập và không chờ đợi nhau. Các thread có thể thực hiện các tác vụ riêng biệt mà không cần đồng bộ hoặc chờ đợi nhau. Điều này có thể tăng hiệu suất và thời gian thực thi, nhưng đồng thời cũng có thể gây ra các vấn đề như đọc/ghi không nhất quán.

Để đồng bộ khi nhiều thread cùng sử dụng một tài nguyên, bạn có thể sử dụng các khối synchronized hoặc các phương thức đồng bộ hóa. Điều này đảm bảo rằng chỉ có một thread được phép truy cập và thay đổi tài nguyên chia sẻ vào một thời điểm. Bằng cách này, bạn có thể tránh các vấn đề như đọc/ghi xung đột và thiếu nhất quán dữ liệu.

- Một số vấn đề cần xử lý khi chương trình có nhiều thread:

Đọc/ghi không an toàn (race condition): Khi nhiều thread cùng đọc/ghi vào một tài nguyên chia sẻ mà không được đồng bộ, có thể xảy ra race condition, dẫn đến kết quả không chính xác hoặc không nhất quán.

Deadlock: Deadlock xảy ra khi các thread bị khóa vì chờ đợi tài nguyên mà không bao giờ được giải phóng bởi các thread khác. Điều này có thể xảy ra khi các thread sử dụng khối đồng bộ hóa theo

*Hướng dẫn: Học về synchronize, lock, unlock, cơ chế wait, notify …, Phần này tương đối phức tạp cần nghiên cứu và làm nhiều bài tập thì càng tốt.*

*Synchronize: Từ khóa "synchronized" trong Java được sử dụng để đồng bộ hóa truy cập vào các phương thức hoặc khối mã. Khi một phương thức hoặc khối mã được đánh dấu là synchronized, chỉ có một luồng được phép thực thi nó tại một thời điểm. Các luồng khác phải chờ đợi cho đến khi luồng hiện tại hoàn thành trước khi được thực thi.*

*Lock và Unlock: Trong Java, các cơ chế khóa và mở khóa được cung cấp bởi các đối tượng Lock. Locks cung cấp khả năng kiểm soát độc quyền truy cập vào một phần mã được bao quanh bởi khối lock và unlock. Khi một luồng gọi phương thức lock(), nó sẽ giữ khóa và truy cập vào phần mã nằm trong khối. Khi hoàn thành, luồng sẽ gọi phương thức unlock() để giải phóng khóa.*

*Wait và Notify: Trong Java, phương thức wait() và notify() được sử dụng để đồng bộ hóa và quản lý sự tương tác giữa các luồng. Khi một luồng gọi phương thức wait() trên một đối tượng, nó tạm dừng và chờ đợi cho đến khi một luồng khác gọi phương thức notify() hoặc notifyAll() trên cùng đối tượng đó để thông báo rằng có sự thay đổi quan trọng đã xảy ra. Sau đó, luồng được đánh thức và tiếp tục thực thi.*

*Các khái niệm trên được sử dụng để tạo ra các cơ chế đồng bộ hóa trong Java, nhằm đảm bảo an toàn và tính nhất quán khi truy cập và sử dụng tài nguyên chung trong môi trường đa luồng. Chúng giúp tránh các vấn đề như đua (race conditions), deadlock và starvation. Điều quan trọng là sử dụng chúng một cách đúng đắn và cân nhắc để đảm bảo tính tin cậy và hiệu suất của ứng dụng.*

1. Logger

* Mục đích, vai trò cho việc logging?

Logging là quá trình ghi lại các thông tin quan trọng, sự kiện hoặc trạng thái của một chương trình hoặc hệ thống vào một tệp tin gọi là log. Việc logging có các mục đích và vai trò quan trọng sau:

Giám sát và theo dõi: Logging cho phép giám sát và theo dõi hoạt động của một chương trình hoặc hệ thống. Bằng cách ghi lại các thông tin quan trọng và sự kiện, log có thể cung cấp một cái nhìn tổng quan về hiệu năng, sự cố, hoặc các hoạt động không mong muốn xảy ra trong hệ thống.

Phân tích và gỡ lỗi: Log cung cấp thông tin hữu ích để phân tích và gỡ lỗi khi có sự cố xảy ra trong chương trình hoặc hệ thống. Bằng cách kiểm tra các thông tin log, người phát triển hoặc nhà quản trị có thể tìm ra nguyên nhân của sự cố, xác định lỗi và tìm giải pháp để khắc phục.

Kiểm tra tuân thủ: Logging có thể được sử dụng để kiểm tra tuân thủ các quy tắc, chuẩn và yêu cầu của hệ thống. Thông qua việc ghi lại các hoạt động, log có thể đóng vai trò như một bằng chứng để xác minh rằng hệ thống hoạt động đúng như mong đợi và tuân thủ các quy tắc đã định.

* Tự tạo ra log, viết chương trình và ghi log, thư viện hỗ trợ log?

<https://poe.com/s/JPjy6gKzHC4kaF0YYPFa>

1. **Bài tập thực hành**
2. Viết 1 chương trình chạy real-time, ghi các số nguyên random ra file output.txt. Dừng chương trình khi gõ lệnh stop trên cửa sổ chương trình.
3. Viết 1 chương trình chạy real-time, cứ n giây in 1 số nguyên random ra mà hình. Dừng chương trình sau n phút.
4. Viết chương trình giải quyết bài toán producer & consumer với các yêu cầu sau:

* Có một message queue chứa các message, cấu trúc message là tùy chọn. Message queue có size giới hạn.
* Một thread đóng vai trò producer: producer định kỳ sẽ tạo ra một message và đưa vào message queue, nếu msgq đã full thì thread sẽ phải đợi cho tới khi msgq không ở trong trạng thái full và tiếp tục tạo message mới đưa vào queue.
* Một thread đóng vai trò consumer: mỗi lần sẽ lấy ra 1 message từ message queue và in message ra màn hình, nếu message queue đang empty thì sẽ phải đợi cho tới khi có message trong queue để xử lý.

1. Viết chương trình đơn giản gửi liên tục, mỗi lần 1 chuỗi tự sinh ngẫu nhiên (Random theo cấu trúc tự tạo) từ máy A, và gửi tới 1 máy tính B khác bằng socket, Máy B nhận được sẽ in ra màn hình kết quả nhận được. Yêu cầu xử lý ngoại lệ trong quá trình gửi nhận (có thể test bằng việc rút dây mạng trong quá trình gửi nhận), ghi ra log file nếu xảy ra lỗi, đưa vào file cấu hình các tham số: ip, port, connection time-out, send time-out, receive time-out … (bài toán có thể mở rộng ra: viết chương trình chát trên 2 máy tính có kết nối với nhau). Gợi ý: viết 2 module trên 2 máy theo cơ chế client – server và kết nối với nhau qua tcp.

**YÊU CẦU CHUNG**: THỰC HIỆN THEO TRÌNH TỰ, học để hiểu, tự tìm hiểu. Bắt đầu với 1 phần mới nên học khái niệm chắc và hiểu, đưa ra được ví dụ và các so sánh giữa các phương pháp; Về phần bài tập yêu cầu comment cho code đầy đủ, gọn gàng; làm thành tài liệu chi tiết.

**Tự học thêm các khái niệm, các kỹ thuật khác không nằm trong đề cương, quy trình phát triển phần mềm (quy trinh RUP, UML, Web service …), Phần Advance tuy nói là advance nhưng bài tập mới dừng ở mức độ làm để hiểu;**

**NÊN TẠO KẾ HOẠCH CỤ THỂ LÀ TỪ NGÀY NÀO ĐẾN NGÀY NÀO LÀM GÌ? ĐẠT ĐƯỢC GÌ? HẾT 1 PHẦN (BASIC, ADVANCE) CÓ THỂ GỬI LẠI REVIEW CŨNG ĐƯỢC (BAO GỒM CẢ TÀI LIỆU VÀ BÀI TẬP)**