**SJAVA 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

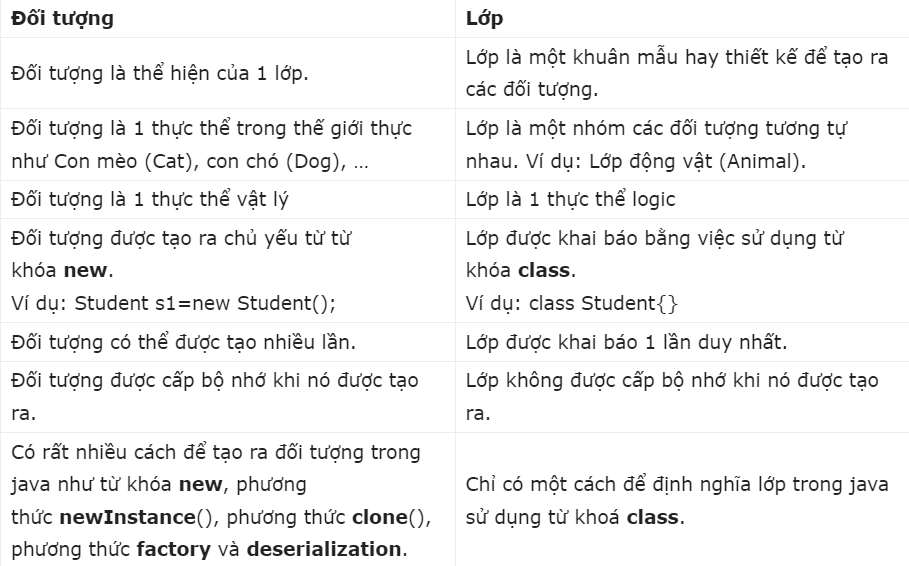
1. Streams Input/Output, làm việc với file mức cơ bản.

* 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
8. Cài đặt oracle (10G or 11G)
9. Sử dụng oracle: các câu lệnh cơ bản, table, store procedure, function, trigger, sequence, partition, job, schedule…
10. Kết nối với oracle, tự làm bài tập để test
11. 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)
12. Đóng gói chương trình, run file trên windows, linux …
13. **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*)**
14. 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.
15. 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.
16. 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
17. 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.

**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
* Khởi tạo, có mấy cách khởi tạo, ưu nhược điểm
* Thread Priority
* Vòng đời của 1 thread, các trạng thái
* Dealock, khái niệm, ví dụ
* Timer, Time Task, Schedule … Khái niệm, cho ví dụ

1. Lập trình mạng
2. socket
3. Tcp: khái niệm, cơ chế, ví dụ
4. udp: khái niệm, cơ chế, ví dụ
5. So sánh tcp và udp
6. MultiThread
7. Khái niệm
8. Tìm hiểu Queue là gì? các loại? cách làm việc với queue
9. Tìm hiểu và làm ví dụ với Thread Pool: Khái niệm, cách sử dụng, ưu nhược điểm …
10. 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)

*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.*

1. Logger1 ngày

* Mục đích, vai trò cho việc logging?
* Tự tạo ra log, viết chương trình và ghi log, thư viện hỗ trợ log?

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)**