**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

|  |  |
| --- | --- |
| Kiểu dữ liệu | Kích thước |
| Boolelan | 1 |
| Char | 2 |
| Byte | 1 |
| Short | 2 |
| Int | 4 |
| Long | 8 |
| Float | 4 |
| Double | 8 |

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

-Mệnh đề if dùng để kiểm tra giá trị dạng boolean của điều kiện , điều kiện kiểm tra của mệnh đề if trả về giá trị true hoặc false.

Có kiểu mệnh đề if else trong java : if , if-else , if-else-if

-for dùng để lặp lại một chương trình , một phần chương trình trong một sô lần cho trước

Cấu trúc cơ bản của câu lệnh for :

for ( khởi tạo biến , kiểm tra điều kiện biến , tăng giảm biến ) ,

Đối với mảng hay collection trong java , có thể dùng for-each để lặp qua các phần tử :

For ( type var : array )

Ta có thể gán nhãn vòng lặp for , điều này là một cách đặt tên cho vòng lặp , giúp chủ động thực hiện các câu lệnh break , continue dễ dàng

Nhãn vòng lặp :: for ( khởi tạo biến , kiểm tra điều kiện biến , tăng giảm biến ) ,

* Khác nhau giữa while , do-while :

Vòng lặp while luôn kiểm tra điều kiện tại đầu vòng lặp trước khi thực thi đoạn mã trong khi do-while , chương trình trong do-while sẽ luôn được chạy 1 lần trước khi kiểm tra điều kiện

* Case switch : dùng để thực hiện 1 hoặc nhiều khối lệnh từ điều kiện
* Continue / break : hai từ khóa này được dùng trong vòng lặp :

+ , continue : khi gặp từ khóa này, vòng lặp hiện tại sẽ bị hủy bỏ , chương trình sẽ tiếp tục khởi tạo 1 vòng lặp mới để tiếp tục

+ break : khi găp từ khóa này , chương trình sẽ thoát khỏi vòng lặp , và tiếp tục với các câu lệnh khác sau vòng lặp.

1. Một số giải thuật làm việc với mảng: sắp xếp (các loại sắp xếp cơ bản), thêm, xóa 1 phần tử trong mảng, …
2. Streams Input/Out put, làm việc với file mức cơ bản.
3. Lập trình hướng đối tượng (phần này cần chi tiết và rõ ràng, có ví dụ cụ thể, làm thành tài liệu)
4. Các khái niệm: Đối tượng là gì? Class là gì?

Đối tượng là một thực thể có trạng thái , hành vi .

Class là một “ khuôn mãu “ để khởi tạo ra các đối tượng có thuộc tính chung , lớp sẽ quy định các thuộc tính và phương thức của các đối tượng được tạo từ lớp đó.

Tên public class được đặt trùng với tên file .java trong project . 1 file chỉ chứa duy nhất 1 public class , nếu class không phải public , 1 file có thể chứa nhiều class

* Access modifiers : từ khóa được dùng trước lớp , thuộc tính , phương thức nhằm chỉ định phạm vi truy cập của thuộc tính, phương thức , lớp .

Access modifiers : private ,public , default , protected.

+ , class

|  |  |
| --- | --- |
| Modifier | Decription |
| Public | Class có thể được truy cập từ tất cả các class khác trong project |
| Default | Đây là trường hợp trong class khi mình không chỉ đinh 1 modifier cho class . Class chỉ có thể truy cập trong cùng 1 package . |

+, attribute , method , constructor :

|  |  |
| --- | --- |
| Modifiers | Decription |
| Private | Chỉ có thể truy cập trong cùng 1 class |
| Public | Có thể truy cập mọi nơi |
| Default | Chỉ có thể truy cập trong cùng 1 package |
| Protected | Giống default , tuy nhiên có thể truy cập được tại những subclass. |

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

Oop có 4 tính chất cơ bản : đóng gói , kế thừa , đa hình , trừu tượng

-đóng gói : là việc che giấu những thông tinh thuộc tính của đối tượng , tùy chỉnh việc truy cập, sửa đối của các đối tượng khác tới nó .

Đạt được tính đóng gói thông qua access modifiers : private và cung cấp các hàm public getter / setter

Các hàm getter/setter sẽ cung cấp tùy chỉnh về truy cấp của đối tượng : write only , read only đồng thời cũng là nơi kiểm tra điều kiện trước khi việc thay đổi , đọc ghi thuộc tính diễn ra.

+, Kế thừa : một lớp có thể có toàn bộ những thuộc tính ,phương thức , constructor của 1 lớp khác => tái sử dụng code

Đạt được sự kế thừa trong java bằng từ khóa “extends” :

class ten\_lóp\_con extends ten\_lop\_cha

Trong java chỉ hỗ trợ đơn kế thừa : một lớp chỉ kế thừa từ 1 lớp khác

+ đa hình : một phương thức có thể được thực thi theo nhiều cách khác nhau.

Đạt được tính đa hình : overloading / override

|  |  |
| --- | --- |
| Overloading | Override |
| +, method signature của hai phương thức khác nhau  +, là việc định nghĩa thêm hành vi cho phương thức.  +, được thực hiện trong cùng 1 class  +, được quyết định tại thời điểm compiler time | +method signuture giống hệt nhau  + là việc định nghĩa lại hành vi method  + được dùng trong hai class có quan hệ kế thừa  + được quyết định thực hiện tại run time |

+ , Trừu tượng : là việc che dấu những thông tin quan trọng , chỉ hiển thị những thông tin quan trọng, che dấu những thông tin quan trọng với người dùng .

Tính trừu tượng giúp ltv tập trung vào những chức năng cốt lỗi, loại bỏ tính phức tạp của đối tượng , tăng tính mở rộng của hệ thống.

Đạt được tính trừu tượng dùng abstract , interface

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

Lớp được khai báo bằng từ khóa class , chúng ta có thể chỉ định các access modifier cho class : public , default .

Tên public class trùng với tên file .java. Một file .java chứa duy nhất 1 public class . 1 file .java có thể có nhiều defautl class.

1. Cho ví dụ minh họa chi tiết
2. Exception handling

Ngoại lệ là một sự kiện bất thường xảy ra trong thời gian runtime , làm gián đoạn luồng hoạt động bình thường của chương trình .

1. Khai báo ngoại lệ

Có 3 loại exception :

+,Checked Exception : lỗi xảy ra lúc Compile time

+,Unchecked Exception : Các ngoại lệ không đc kiểm tra tại thời điểm biên dịch, đc kiểm tra tại thời gian chạy

+Error : lỗi không thể phục hồi

Throwable là lớp gốc của hệ thống phân cấp ngoại lệ. Nó được kế thừa bởi 2 lớp con là Exception và Error

1. Cơ chế try-catch

* Là cơ chế để xử lí ngoại lệ

Chi tiết :

|  |  |
| --- | --- |
| Key word | Decription |
| Try | Chứa đoạn mã chứa ngoại lệ |
| Catch | Chứa đoạn mã dùng để xử lí ngoại lệ |
| Finally | Chứa đoạn mã luôn được thực thi |
| Throw | Ném ngoại lệ |
| Throws | Khai báo ngoại lệ |

1. Throw ngoại lệ
2. Chuyển tiếp ngoại lệ
3. Lập trình xử lý với database
4. Cài đặt oracle (10G or 11G)
5. Sử dụng oracle: các câu lệnh cơ bản, table, store procedure, function, trigger, sequence, partition, job, schedule…
6. Kết nối với oracle, tự làm bài tập để test
7. 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)
8. Đóng gói chương trình, run file trên windows, linux …

git : <https://github.com/doanphung0402/javacore_xtel.git>

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.

**JAVA CORE ADVANCE**

1. Streams Input/out put, làm việc với file mức nâng cao (tự nghiên cứu sử dụng các thư viện nâng cao trong Java), ưu nhược điểm.

1. Thread (chi tiết, thực hành ngay nếu học xong lý thuyết)

* Khái niệm
  + - Một thread là một quy trình con nhẹ , là đơn vị nhỏ nhất trong java , mỗi thread thực thi 1 công việc riêng biệt
    - Tại một thời điểm chỉ có duy nhất một luồng thực thi.
    - Java chính là một ngôn ngữ lập trình đa phân luồng (**multithreaded**) => có thể chia nhỏ nhiều hoạt động riêng biệt bên trong 1 ứng dụng lớn.
    - Các thread có thể chạy song song và biệt lập lẫn nhau.

Khi một luồng được thực thi, nó sẽ không thể được thực thi lại => lỗi IllgalThreadStateException.

Mỗi thread chạy trong 1 callstack ;

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

Có hai cách tạo 1 thread :

* + extends Thread
  + Implement Runable

|  |  |
| --- | --- |
| extends Thread | Implements runable |
| Kế thừa lớp Thread , cung cấp nhiều hàm tạo và phương thức để thao tác trên 1 thread.  Chỉ có thể kế thừa duy nhất 1 class Thread .  Luồng khởi chạy bởi class extends Thread cần khởi tạo đối tượng và gọi hàm start() để khởi chạy => mỗi luồng gắn duy nhất với 1 đối tượng , không thể cập nhật chung tài nguyên trong mỗi luồng | Triển khai interface runable , có duy nhất phương thức run() để khởi chạy luồng .  Tận dụng được tính chất đa kế thừa của interface trong java,=> cho phép chia sẻ 1 đối tượng cho nhiều Thread.  Dùng hàm tạo của Thread class khởi tạo luồng :   * Thread(String name) * Thread (Runable runable ,String name) * Thread(Runable runable) * Thread()   Hàm tạo nhận vào obj Runable => dùng chung 1 obj cho 1 nhóm luồng => dùng khi cằn cập nhật chung 1 tài nguyên cho 1 nhóm luồng . |

* Dùng runable khi :

+ tạo tính loosely-coupled trong ứng dụng

+ cần cập nhật chung 1 tài nguyên cho đối tượng

+ logic bớt phức tạp

* Thread Priority

Thread Scheduler trong java nhận nhiệm vụ điều phối hoạt động giữa các luồng,

Có hai yếu tố để Thread Scheduler quyết định khi nào 1 luồng đc thực thi : time of arrival , Priority

* + - * Priority : Mức độ ưu tiên của luồng , có giá trị từ 1-> 10 , priority càng cao mức độ ưu tiên được chạy càng lớn
      * Time of arrival : nếu hai luồng cùng mức độ ưu tiên => time of arrival của luồng sẽ được ưu tiên lựu chọn.

Priority có 3 constant : MIN\_PRIORITY =1 , MAX\_PRIORITY =10 , NORM\_PRIORITY =5 (default) .

Mặc định priority của thread là 5 , của main thread là 10

Dùng các phương thức getPriority/setPriority để lấy hoặc set độ ưu tiên cho thread.

* Vòng đời của 1 thread, các trạng thái

Các luồng có 5 trạng thái : new state , active state , waiting/blocked state ,timed waiting state , terminated state.

* New state : trạng thái đầu tiên khi khởi tọa của thread , tồn tại cho tới kho chương trình bắt đầu chạy , tại thời điểm này thread có thể chạy tại bất cứ khi nào đc gọi
* Active state : khi thread đang thực hiện tác vụ của mình .

+, runable state : thread được hệ thống cung cấp tài nguyên và thời gian , sẵn sàng để chạy

+, run state : thread chuyển từ runable xang running , thread thực thi công việc cua mình trong thời gian mà tiến trình cung cấp .

* Waiting / Blocked state : trạng thái chờ của 1 thread , thread vào trạng thái waiting hay block , hay là nó sẽ đợi vô hạn để chờ 1 tác vụ hay tài nguyên nào đó. Và khi hệ thống ycau nó sẽ chuyển về trạng thái runable .

Khác nhau : waiting của thread do bản thân thread tụ áp đặt để đạt được sự đồng bộ với các luồng khác.

Block : do cơ chế của các tiến trình áp đặt giàm giải quyết việc chia sẻ tài nguyên găng trong hệ thống.

* Time waiting : là trạng thái waiting của luồng , thread sẽ về trạng thái waiting trong một khoảng thời gian xác định và sẽ chờ thông báo của những thread khác để chuyển về trạng thái runable .
* Terminated : là trạng thái cuối vòng đời của thread , khi nó hoàn thành xong tất cả các tác vụ của mình .
* 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 multi threading)

*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)**