1. Constants là gì ?

* Là một biến có giá trị **không thể thay đổi khi nó đã được gán** . Java không có hỗ trợ tích hợp cho các hằng số.
* Cú pháp :

Final float pi = 3.14f;

1. Variables là gì ?

* Là biến được đặt cho một vị trí trong bộ nhớ
* Giá trị của 1 biến có thể thay đổi trong quá trình thực thi
* Tất cả các biến được khai báo trước khi sử dụng
* Biến có 3 thành phần : DataType, DataName, Value

1. Data type là gì ?

* Là kiểu dữ liệu chia làm 2 loại
* Kiểu dữ liệu nguyên thủy: số nguyên (byte, short, int, long), số thực(float, double), boolean (true/false), char(ký tự)
* Kiểu dữ liệu không nguyên thủy : string, array, object

1. Operations ?

* Java cung cấp một tập hợp các toán tử phong phú để thao tác với các biến. Chúng ta có thể chia tất cả các toán tử Java thành các nhóm sau:
* Toán tử số học ( +, -, \*, /, %, ++, --)
* Toán tử chuyển nhượng
* Toán tử so sánh ( ==, !=, >, <, >=, <=)
* Toán tử logic (&&, ||, !)
* Toán tử bitwise

1. Conditions ?

* If, if else (câu lệnh điều kiện if xảy ra nếu đúng)
* Switch casa (Sử dụng switch câu lệnh để chọn một trong nhiều khối mã sẽ được thực thi.)
  + Các switchbiểu thức được tính một lần.
  + Giá trị của biểu thức được so sánh với giá trị của mỗi biểu thức case.
  + Nếu có sự trùng khớp, khối mã được liên kết sẽ được thực thi.
  + Các breakvà defaulttừ khóa là không bắt buộc, và sẽ được mô tả sau trong chương này

6. Function và method ?

* Function : là một khối mã được viết để thực hiện một số tác vụ cụ thể. Chúng ta có thể xác định một hàm bằng cách sử dụng từ khóa *hàm* , theo sau là *Tên* và các *tham số* tùy chọn . Thân hàm được bọc trong nẹp Xoăn.
* Cú pháp : function functionname ( tham số ) { // nội dung }
* Method : là một thuộc tính của một đối tượng có chứa định nghĩa hàm. Phương thức là các hàm được lưu trữ dưới dạng thuộc tính đối tượng. Phương thức đối tượng có thể được truy cập bằng cú pháp sau:

Đối tượng = {

methodName : function () {

//Nội dung

}

}

Object.methodName.

7. Arraylist và Linkedlist ?

|  |  |  |
| --- | --- | --- |
|  | **ArrayList** | **LinkedList** |
| Cấu trúc dữ liệu (Structure) | ArrayList sử dụng **mảng động**để lưu trữ các phần tử. ArrayList  là một cấu trúc dữ liệu dựa trên chỉ mục (index), trong đó mỗi phần tử (element) được liên kết với một chỉ mục. | LinkedList sử dụng **danh sách liên kết (Doubly Linked List)**để lưu trữ các phần tử. Các phần tử trong LinkedList được gọi là node, mỗi node cần lưu trữ 3 thông tin: tham chiếu phần tử trước nó, giá trị của phần tử và một tham chiếu tới phần tử kế tiếp. |
| Thao tác thêm và xóa (Insertion And Removal) | Thao tác thêm và xóa phần tử với ArrayList là chậm bởi vì nó sử dụng nội bộ mảng. Bởi vì sau khi thêm hoặc xóa các phần tử cần sắp xếp lại. Độ phức tạp: **O(n).** | Thao tác thêm và xóa phần tử với LinkedList nhanh hơn ArrayList. Bởi vì nó không cần sắp xếp lại các phần tử sau khi thêm hoặc xóa. Nó chỉ cần cập nhật lại tham chiếu tới phần tử phía trước và sau nó. Độ phức tạp: **O(1).** |
| Thao tác tìm kiếm hoặc truy xuất phần tử (Retrieval) | Truy xuất phần tử trong ArrayList nhanh hơn LinkedList. Bởi vì các phần tử trong ArrayList được lưu trữ dựa trên chỉ mục (index). Độ phức tạp:  **O(1).** | Thao tác truy xuất phần tử trong LinkedList chậm hơn nhiều so với ArrayList. Bởi vì, nó phải duyệt qua lần lượt các phần tử từ đầu tiên cho đến cuối cùng. Độ phức tạp:  **O(n).** |
| Truy xuất ngẫu nhiên (Random Access) | ArrayList có thể truy xuất ngẫu nhiên phần tử. | LinkedList không thể truy xuất ngẫu nhiên. Nó phải duyệt qua tất cả các phần tử từ đầu tiên đến phần tử cuối cùng để tìm phần tử. |
| Trường hợp sử dụng (Usage) | ArrayList chỉ có thể hoạt động như một list vì nó chỉ implements giao tiếp **List**. | LinkedList có thể hoạt động như một **ArrayList**, stack (hàng đợi), queue (hàng đợi), **Singly Linked List and Doubly Linked List** vì nó implements các giao tiếp List và Deque. |
| Sử dụng bộ nhớ | ArrayList yêu cầu **ít bộ nhớ hơn** so với LinkedList. Bởi vì ArrayList chỉ lưu trữ dữ liệu (data) của nó và chỉ mục (index). | LinkedList yêu cầu **nhiều bộ nhớ hơn** so với ArrayList. Bởi vì LinkedList lưu giữ thông tin của nó và tham chiếu tới phần tử trước và sau nó. |
| Khi nào sử dụng? | ArrayList **tốt hơn trong việc lưu trữ và truy xuất dữ liệu (get).** | LinkedList**tốt hơn trong việc thao tác dữ liệu (thêm/ xóa).** |

8. Stack và Queue ?

8.1 Stack

* Stack hay còn gọi là ngăn xếp, là 1 loại cấu trúc dữ liệu trừu tượng hoạt động theo nguyên lý "Vào Sau Ra Trước" - Last In First Out(LIFO).
* Một số thao tác :
* push: đẩy 1 phần thử vào đầu ngăn xếp.
* pop: xóa 1 phần tử đầu khỏi ngăn xếp.
* peek: lấy 1 phần tử đầu của ngăn xếp mà không xóa phần tử đó.
* isFull: kiểm tra ngăn xếp đã đầy chưa.
* isEmpty: kiểm tra ngăn xếp có rỗng không.
* **Chú ý:** Tất cả các thao tác với ngăn xếp chỉ tác động đến phần tử ở đầu ngăn xếp.

8.2 Queue

* Queue hay còn gọi là hàng đợi, là 1 loại cấu trúc dữ liệu trừu tượng hoạt động theo nguyên lý "Vào Trước Ra Trước" - First In First Out(FIFO).
* Một số thao tác:
* enqueue: thêm 1 phần tử vào hàng đợi.
* dequeue: xóa 1 phần tử khỏi hàng đợi.
* peek: lấy 1 phần tử đầu của hàng đợi mà không xóa nó.
* isFull: kiểm tra hàng đợi đã đầy chưa.
* isEmpty: kiểm tra hàng đợi có rỗng không.
* **Chú ý:** Tất cả các thao tác với hàng đợi sẽ tác động đến phần tử đầu và cuối của hàng đợi.

10. My SQL ( Structured Query Language )

10.1 Trên cơ sở dữ liệu ( database )

create database: tạo cơ sở dữ liệu

create database HoSoSinhViencreate database HoSoSinhVien

on (name=’HSSV\_data’, filename=’c:HSSV\_data.mdf’)

log on (name=’HSSV\_log’, filename=’c:HSSV\_log.ldf’)

drop database: xóa cơ sở dữ liệu

drop database HoSoSinhVien

alter database: sửa các thông tin của cơ sở dữ liệu

alter database HoSoSinhVienmodify name = HSSV

exec sp\_dboption: sửa các thông tin của cơ sở dữ liệu

exec sp\_dboption HoSoSinhVien, ‘read only’, ‘true’exec sp\_dboption HoSoSinhVien, ‘autoshrink’, ‘true’

exec sp\_dboption HoSoSinhVien, ‘single\_user’

dbcc: điều khiển cơ sở dữ liệu

dbcc shrinkdatabase(HoSoSinhVien, 10)

10.2 Trên bảng (table)

**create table: tạo bảng**

Tạo bảng với các cột

create table SinhVien (MaSV int,TenSV nchar(50))

Tạo với các ràng buộc

create table SinhVien (MaSV int primary key,TenSV nvarchar(50) not null,QueQuan int references DiaPhuong(MaDP))

#### drop table: xóa bảng

drop table SinhVien

#### alter table..add: thêm cột

Thêm cột bằng cách dùng lệnh alter table

alter table SinhVienadd QueQuan int

#### alter table..drop column: xóa cột

alter table SinhVien drop column QueQuan

#### alter table..alter column: thêm thuộc tính not null

alter table SinhVienalter column TenSV nchar(50) not null

#### alter table..add primary key: thêm khóa chính

alter table SinhVienalter column MaSV int not null

go

alter table SinhVien

add primary key (MaSV)

#### alter table..add foreign key: thêm ràng buộc khóa ngoài

alter table SinhVien add foreign key (QueQuan) references DiaPhuong(MaDP)

#### alter table..add default: thêm ràng buộc mặc định

alter table SinhVien add default 'khong ten' for TenSV

#### exec sp\_help: xem thông tin bảng

exec sp\_help SinhVien

## **10.3) Trên bản ghi (record)**

insert into SinhVien (MaSV, TenSV, QueQuan)values (1, N’Nguyễn Văn A’, 1)

#### insert..select: thêm các bản ghi từ bảng khác vào bảng

insert into DocGiaselect MaSV, TenSV, QueQuan from SinhVien

insert into DocGia

select MaGV, TenGV, QueQuan from GiaoVien

#### select..into: đưa kết quả lựa chọn vào bảng mới

select MaSV, TenSV, Dieminto SinhVienKha

from SinhVien

where Diem > 7.0

#### delete: xóa các bản ghi từ bảng

delete from SinhViendelete from SinhVien where MaSV=1234

#### truncate: xóa toàn bộ bản ghi của bảng

truncate table SinhVien

#### update: sửa các bản ghi trong bảng

update SinhVienset NhomTruong = 1

where MaSV < 4

update SinhVien

set NhomTruong = 4

where (MaSV > 3 and MaSV < 8)

**10.4) Truy vấn (query)**

Hỗ trợ truy vấn: distinct, top, as, identity

Phép toán tập hợp: in, like, between

Các hàm tổng nhóm: sum, max, min, avg

### **10.4.1) Truy vấn đơn giản**

#### select \*: Hiện tất cả bảng

select \*from SinhVien

#### select: Hiện một số cột

select TenSV, DiemTBfrom SinhVien

#### select..where: Hiện một số dòng / bản ghi

select TenSV, DiemTBfrom SinhVien

where DiemTB > 6.0

#### select..order by: Hiện và sắp xếp theo điểm rồi theo tên

select TenSV, DiemTBfrom SinhVien

order by DiemTB desc, TenSV asc // asc sắp sếp tăng dần, desc là giảm dần

#### select..distinct: Hiện danh sách giá trị không trùng lặp

#### select distinct QueQuanfrom SinhVien

#### select..top: Hiện các dòng đầu tiên trong bảng

select top 3 TenSV, DiemTBfrom SinhVien

order by DiemTB desc, TenSV asc

### **10.4.2) Truy vấn lồng nhau (nested query)**

#### select..where (select)

Hiện tất cả những người trong bảng nhân viên có lương bằng lương lớn nhất của những người có trong công ty:

select TenNV, Luongfrom NhanVien

where Luong = (select max(Luong) from NhanVien)

#### select..where (in)

Hiện tất cả những người trong bảng nhân viên có lương lớn nhất hoặc lớn nhì của những người có trong công ty:

select TenNV, Luongfrom NhanVien

where Luong in (select top 2 Luong from NhanVien order by Luong)

Câu lệnh select trong sẽ tạo ra một tập hai giá trị (top 2) đó là lương lớn nhất và lương lớn nhì. Và câu lệnh select thứ nhất sẽ chọn ra những người mà lương nằm trong tập lớn nhất và lớn nhì

#### select..where (in sub)

Hiện ra tất cả những người có lương lớn nhất phòng của anh ta (không phải lớn nhất trong công ty mà lớn nhất trong phòng hoặc đơn vị mà anh ta thuộc về)

select nv1.TenNV, nv1.Luongfrom NhanVien as nv1

where nv1.Luong = (select max(Luong) from NhanVien where Phong=nv1.Phong)

Câu lệnh select trong sẽ trả về giá trị lương lớn nhất nhưng không phải lớn nhất trong toàn công ty mà lớn nhất trong phòng của nv1. Sau đó câu lệnh select ngoài cùng sẽ xác định xem nv1 có được chọn không bằng cách kiểm tra lương anh ta với lương lớn nhất của phòng anh ta.

### **10.4.3) Truy vấn tổng nhóm (subtotal query / grouping query)**

#### select..group by: Thống kê theo tiêu chí

Hiện ra số lượng các nhân viên ứng với từng quê

select QueQuan, count(\*)from NhanVien

group by QueQuan

Đếm số nam và số nữ trong công ty

select GioiTinh, count(\*)from NhanVien

group by GioiTinh

Tính tổng thu nhập theo từng phòng

select Phong, sum(Luong)from NhanVien

group by Phong

#### select..having: Hiện ra một số nhóm phù hợp

Chỉ đếm số lượng người ở Hải Phòng và số lượng người ở Hà nội

select QueQuan, count(\*)from NhanVien

group by QueQuan

having (QueQuan = ‘HP’, QueQuan = ‘HN’)

Chỉ hiện ra những phòng nào có tổng thu nhập lớn hơn 500000

select Phong, sum(Luong)from NhanVien

group by Phong

having sum(Luong) > 5000000

Chỉ hiện ra những tỉnh nào có số lượng người lớn hơn 10

select QueQuan, count(\*)from NhanVien

group by QueQuan

having count(\*) > 10

### **10.4.4) Truy vấn liên bảng (cross table query / joining query)**

#### select..inner join: ghép các cặp bản ghi thỏa mãn điều kiện

Ghép bảng nhân viên và hiện ra tên nhân viên và tên địa phương

select NhanVien.TenNV, DiaPhuong.TenDP

from NhanVien

inner join DiaPhuong on NhanVien.QueQuan = DiaPhuong.MaDP

#### select..left outer join: lấy tất cả phía trái và ghép (nếu có) với phải

Lấy tất cả những nhân viên kể cả những nhân viên có quê quán không hợp lệ (nghĩa là mã quê quán không có trong bảng địa phương)

select NhanVien.TenNV, DiaPhuong.TenDP

from NhanVien

left outer join DiaPhuong on NhanVien.QueQuan = DiaPhuong.MaDP

#### select..right outer join: lấy tất cả phía phải và ghép (nếu có) với phía trái

Lấy tất cả những địa phương ghép với nhân viên, các địa phương không hợp lệ sẽ được ghép với bộ dữ liệu rỗng. Không hiện ra các nhân viên không có mã quê quán phù hợp

select NhanVien.TenNV, DiaPhuong.TenDP

from NhanVien

right outer join DiaPhuong on NhanVien.QueQuan = DiaPhuong.MaDP

#### select..full outer join: lấy từ hai phía và ghép nếu có

Lấy tất cả những nhân viên (nếu không có quê quán phù hợp thì ghép với bộ dữ liệu rỗng) và tất cả những địa phương kể cả không có nhân viên.

select NhanVien.TenNV, DiaPhuong.TenDP

from NhanVien

right outer join DiaPhuong on NhanVien.QueQuan = DiaPhuong.MaDP

#### select..cross join: trả về tất cả các cặp có thể ghép

Ghép từng nhân viên với tất cả các địa phương. Như vậy nếu có m nhân viên và có n địa phương thì bảng đích sẽ có m\*n dòng. n dòng đầu cho nhân viên thứ nhất ghép với các địa phương. n dòng sau cho nhân viên thứ hai ghép với các địa phương. và tiếp tục như thế tới nhân viên thứ m.

select NhanVien.TenNV, DiaPhuong.TenDP

from NhanVien

cross join DiaPhuong

11. OOP (Object Oriented Programming)

## 11.1 **Lập trình hướng đối tượng (OOP) là gì? OOP được dùng để làm gì?**

- Là một phương pháp lập trình dựa trên khái niệm về lớp và đối tượng.

- Mục tiêu của OOP là tối ưu việc quản lý source code, giúp tăng khả năng tái sử dụng

## 11.2 **Đối tượng (Object) và Lớp (Class) trong OOP là gì?**

### **11.2.1 Đối tượng (Object)**

Đối tượng trong OOP bao gồm 2 thành phần chính:

* Thuộc tính (Attribute): là những thông tin, đặc điểm của đối tượng
* Phương thức (Method): là những hành vi mà đối tượng có thể thực hiện

### 11.2.2 **Lớp (Class)**

* Lớp là sự trừu tượng hóa của đối tượng. Những đối tượng có những đặc tính tương tự nhau sẽ được tập hợp thành một lớp. Lớp cũng sẽ bao gồm 2 thông tin là thuộc tính và phương thức.
* Một đối tượng sẽ được xem là một thực thể của lớp.

11.3 Ưu điểm của lập trình hướng đối tượng (OOP)

* OOP mô hình hóa những thứ phức tạp dưới dạng cấu trúc đơn giản.
* Code OOP có thể sử dụng lại, giúp tiết kiệm tài nguyên.
* Giúp sửa lỗi dễ dàng hơn. So với việc tìm lỗi ở nhiều vị trí trong code thì tìm lỗi trong các lớp (được cấu trúc từ trước) đơn giản và ít mất thời gian hơn.
* Có tính bảo mật cao, bảo vệ thông tin thông qua đóng gói.
* Dễ mở rộng dự án.

## 11.4 **4 đặc tính cơ bản của OOP**

### 11.4.1 **Tính đóng gói (Encapsulation)**

* Cho phép che giấu thông tin và những tính chất xử lý bên trong của đối tượng.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifier** | **Trong lớp** | **Trong package** | **Chỉ các lớp con trong cùng package** | **Ngoài package** |
| **Private** | Y | N | N | N |
| **Default** | Y | Y | N | N |
| **Protected** | Y | Y | Y | N |
| **Public** | Y | Y | Y | Y |

### 14.4.2. **Tính kế thừa (Inheritance)**

15. Mô hình MVC

**MVC** là từ viết tắt bởi 3 từ **Model – View – Controller**. Đây là mô hình thiết kế sử dụng trong kỹ thuật phần mềm.



\*Mô hình MVC và các thành phần bên trong của MVC.

1. Model (M)

* + - * Là bộ phận có chức năng lưu trữ toàn bộ dữ liệu của ứng dụng.
      * Là bộ phận cầu nối giữa Controller và View.
      * Model thể hiện dưới hình thức là một CSDL hay 1 file XML bình thường

2. **View (V):**

* Là phần giao diện (theme) dành cho người sử dụng.
* Nơi mà người dùng có thể lấy được thông tin dữ liệu của MVC thông qua truy vấn hoặc sử dụng thông qua website
* Các ứng dụng web thì View là một phần của hệ thống, nơi các thành phần HTML sẽ được tạo ra
* View không quan hệ trực tiếp với Controller mà chỉ hiển thị yêu cầu chuyển cho controller

**3. Controller (C):**

* Bộ phận có nhiệm vụ xử lý các yêu cầu người dùng đưa đến thông qua view.
* Từ đó, C đưa ra dữ liệu phù hợp với người dùng.
* Bên cạnh đó, Controller còn có chức năng kết nối với model.

**16. HTTP**

**16.1 HTTP (request, response, header, body)**

**16.1.1 Http Request**

* **Là thông tin của client gửi lên server yêu cầu server làm 1 cái gì đó**

**16.1.2 Http Response**

* **Là kết quả mà server trả về cho client**

**16.1.3 Http Header**

* **Cho phép client và server chuyển thông tin bổ sung với một yêu cầu hay phần hồi HTTP**