Day 1

03-09-2021

Unix Training

Unix is an Operating system (OS) developed in Bell laboratories.

Unix is multi-tasking, multi-user, high secure and very high performance os

In Unix OS we can customize according to own requirement.

Unix is Open Source.

Unix is command base OS.

Unix flavors

Sun : Solaris

IBM : AIX

Apple : MAC

HP : HP-UX

Google : Android

Unix

And

Linux :it is GUI base

Shell : It is a command or instruction given to kernel through System call to execute some task.

Cd

Shell

1. Read line from standard input device (each of these line is called pipe line).
2. Splits the command into tokens.
3. Checks the token to see If it is a keyword.
4. Check the first word alias.
5. Then execute the command.

Unix Commands

1. echo
2. print
3. date --date ‘Date Format’: check other formats.
4. date +%d-%M-%Y : check other formats.

**File system :**

**pwd** : current directory path.

**ls** : it display all files and folder present in current directory.

mkdir folderName : This command is use to create the folder or directory

mkdir folderName folderName folderName : this command is use to create more than one folder.

rmdir folderName : This command is use to delete the folder.

cd folderName : this command is use to move inside a folder.

mv oldFolderName newFolderName : This command is use to change the folder.

folderName start with pre-fix . consider as hidden folder.

mkdir .folderName

ls : this command is use to display. Non hidden folder.

ls –a : This command I use to show normal as well as hidden folder.

Mv oldFolderName .folderName : This command is use to hide the folder.

cd .. : this command is use to move to parent directory of current directory.

cd ~ : this command is use to move to root directory.

Unix file system

Creating the file in Unix

1. cat command :
   1. creating file using cat command

cat > filename.txt : it create the new file with allow to enter the contents inside a file.

* 1. Reading the content from file using cat command

Cat < filename.txt or cat filename

* 1. Cat >> filename : this command is use to append the content in existing file.

1. Touch filename : This command is use to create the empty file

Touch filename filename filename

1. Using echo

echo “Welcome to Unix Training “ > filename

head and tail command :

head –n filename : this command is use to display the top n number line from a file

tail –n filename : this command is use to display the bottom n number of line from a file

wc : word count :

wc –l filename : number of lines

wc –w filename : number of words

wc –c filename : number of character

cp : copy the content from one file to another file.

cp sourcefilename destinationfilename.

Delete the file

rm filename

rm –I filename : it ask the confirmation to delete the file.

ls folderName : This command is use to display directory details.

rmdir folderName This command is use to delete the folder if folder is empty.

rm –r folderName This command is use to delete the folder doesn’t matter folder empty or contains set of files.

Filter command : flat file : in a file data entered by using delimiter ie space , \_, tab etc.

Employee.txt file with few records

cut -c 1 Employee.txt : 1 index position

cut -c 3 Employee.txt : 3 index position

cut -c 3-6 Employee.txt : 3 to 6 range index position

cut -d ' ' -f2 Employee.txt : -d delimiter ‘ ’ space f2 2nd columns

cut -d ',' -f1 Manager.txt : -d delimiter ‘,’

paste Employee.txt : display the content

paste –s Employee.txt : serial format

paste Employee.txt Manager.txt both file contents.

Tr ( translate command)

This command is use to translate data from one format to another format.

tr SET1 SET2 < Employee.txt

search SET1 content and replace by SET2 contents in file.

tr abcde ABCDE < Employee.txt

a-z A-Z < Employee.txt

tr ' ' , < Employee.txt

tr [:lower:] [:upper:] < Employee.txt : please work more pre-defined keywords to translate the content from file.

sort Employee.txt : ascending order as 1st field

sort –r Employee.txt : descending order

sort -k2 Manager.txt k2 send field

sort -k2 -t ',' Manager.txt -t delimiter ‘,’ by default space consider.

EmployeeInfo.txt

Id,Name,Salary,Desg, City, DeptId

4,Raj,19000,Developer,Bangalore,10

2,Raju,18000,Developer,Delhi,20

5,Ramesh,15000,Developer,Bangalore,20

8,Ajay,12000,Developer,Delhi,10

9,Vikash,19000,Developer,Bangalore,20

6,Ram,12000,Developer,Delhi,20

7,Vijay,18000,Developer,Bangalore,10

1. Sort by Id
2. Sort by city
3. Sort by City and sub sort by DeptId

Note : column name not sort(ignore).

GREP Commands GREP Global Regular Express Print

Grep command actually search then the content in file base upon the pattern.

The grep command mainly divided into three parts.

1st grep command followed by pattern to search in file and then filename or contents.

grep 'e' grepdemo.txt : display the line where e character present

grep –i 'e' grepdemo.txt : ignore case sensitive

grep -c -i 'E' grepdemo.txt : it display the number of line where E character present.

grep -n -i 'e' grepdemo.txt : it display the line which contains e character as well as it display the line numbers.

Please work more GREP Commands

SED : The SED command Unix stands for stream editing commands.

Using SED command we can modify the contents of the file.

SED command in Unix basically use search and replace the text or contents from a file.

Pipe commands

Pip e command is use to combine more than command

So first command execute it provide the output and that is input for another command.

echo “Welcome to Unix Training “ |

SED command

1. Read a line from input stream
2. Execute the command on a line
3. Display the result on output stream.

In SED command we use s and g

S : substitution

G : means many occurrence.

sed s/Raj/'Raj Deep'/g seddemo.txt

sed s/EE/i/ seddemo.txt

sed s/ee/i/g seddemo.txt

sed s/ee/i/ seddemo.txt

sed '2 s/R/r/g' seddemo.txt

sed -i s/Raj/'Raj Deep'/gi seddemo.txt : this command search, replace and update in same file.

sed s/'Hor r you'/'How are You'/g a.txt

SED commands

Day 2

06-09-2021

**Awk : awk is one of the most powerful tool in Unix use for processing the rows and column in the file base upon the delimiter.**

**Syntax**

**Awk ‘BEGIN {start\_action} {action} END {stop\_action}’ filename**

**Start action : initialization**

**Action : looping**

**End action : at the last**

**awk -F ',' '{print $0}' Employee.txt : display all rows**

**awk -F ',' '{print $1}' Employee.txt : display specific fields from a file**

**awk -F ',' '{print $2}' Employee.txt : display specific fields from a file**

**NF : Number of Fields**

**NR : Number of records**

**GREP : grep is useful if you want to quickly search data from line or file that match our pattern. It also return some other simple information like matching number, match count, and file names**

**SED : it is use when you want to make any changes in a file base upon the pattern match. If allow you to easily match part of line, makes modification and print out result or update in file.**

**AWK : it is entire programming language. Build around reading any type of file (CSV (comma separated values) style file, processing records and optionally print out the result.**

**File access permission**

**File**

**Types of users**

1. **File owner**
2. **Group owner**
3. **Others**

**U -> it represent the owner**

**G -> it represent group of owner**

**O-> it represents any other user but not belong to user group.**

**Permission for regular file**

1. **Open file : Read mode**
2. **Write file or modify : write mode**
3. **Execute : Read and Execute**

**ls –l filename : this command is use to display the properties of a file.**

1. **First field display - : means it is a file**
2. **The owner has over the file. rw**
3. **The group has over the file. r**
4. **Other people has over the file r**
5. **Fifth : number of links for that folder or file. 1**
6. **User that own the file or directory**
7. **Group details.**
8. **Size of the file in byte**
9. **Time when last modification happen**
10. **File name**

**chmod : change mode**

**chmod <permission> <filename>**

**code**

**a : all**

**u : user**

**g : group**

**o : other**

**+ : add**

* **: remove**

**R : read , w : write , x : execute**

**0 🡪 no permission**

**1 🡪 execute permission**

**2 -> write permission**

**4 🡪 read permission**

**4 + 2 +1 : 7 (read, write and execute)**

**4+0+1 : 5 (read, no write and execute)**

**4+0+0 : 4 (read, no write and no execute)**

**chmod 754 fileName**

**User 🡪 read, write and execute**

**Group 🡪 read, no write and execute**

**Other 🡪 read, not write and execute**

**Chmod 777 filename**

Shell scripting

A shell is a program that takes command typed by user and send the instruction to kernel through system call and Kernel run the command.

A shell is a program that acts as the interface between you and the Unix or Linux operating system.

Types of Shell

1. The Bourne Shell
2. Sh : Posix
3. Ssh : Korne shell
4. Bash :
5. C shell
   1. csh
   2. Tops tcsh

sudo apt install shellName

shell programming

we can do shell programming using two ways

1. You can type a sequence of command and allow the shell to execute them interactively.
2. You can store those commands in a file and you can run the file.

VI Editor

Open the file with vi editor

vi filename.txt

1. Command mode
2. Insert mode
3. Exit mode

Open the open the file to move from command mode to insert mode we have write i

Then write the contents

To come out from insert mode to command mode we have to click esc key.

To save the write :w

To quite the file :q

To save and quite : wq!

**Shell scripting**

You create the file and write one or more command and save the file with extension .sh

Variable : variable is a name which hold some value and the value can change during the execution of a program.

2 types

1. pre-defined variable : This provide the OS details. All pre-defend variable are in upper case. When we are using those variable we have to start with $ symbol.
2. User-defined variable

Taking the value through keyboards

1. Command line arguments.

sh ./filename firstargument secondarguemnt thirdargument

$1 : first parameter

$2 : second parameter

$3 : third parameter

$0 : script file name

$\* : number of arguments

$# : all arguments .

1. Using read property

-gt

-gte

-lt

-lte

-eq

-ne

Day 3

07-09-2021

Oracle SQL and Pl SQL

C, C++, Java, Python etc.

Input read a,b

Process sum = a+b

Output display sum

File handling programs

API : Application Programming interface.

Store the data permanently

1. File base system

File base system limitation

1. We can store same data again and again ie data redundancy(duplicate records).
2. Data is not a secure. (read or write mode).
3. CRUD Operation is very complex (Create, Read, Update and Delete).
4. Consistency ( format of the file, type of file etc).

Data : raw facts.

Information : processed data or meaningful data.

Database : it is used to store the data in table format using column and row.

DBMS : Database Management System. It is a software which help to store the data in a table format.

Excel DBMS.

Employee

Id Name Salary

100 Raj 12000

101 Raju 14000

102 Mahesh 16000

103 Rajesh 18000

Database Model

1. Hierarchical model: two tables or files are connected to each other from top to bottom like parent and child relationship.

1. Network model
2. Relational model : logical relationship between two tables.

Trainer\_Students\_Details

TId TName Tech Sid SName age

100 Raj Java 1 Seeta 21

100 Raj Java 2 Reeta 22

100 Raj Java 3 Meeta 23

100 Raj Java 4 Veeta 24

Trainer

PK

TId TName Tech

100 Raj Java

101 Raju Python

Students

PK FK

SId SName Age TSId

1 Reeta 21 100

2 Meeta 22 100

3 Veeta 23 101

4 Leeta 24 101

Database Model

RDBMS : Relational Database management system.

Dr. EF Codd’s Rules : 12 rules 0 to 11 rules.

In DBMS : Table column row

In RDBMS : Relation attribute or fields tuple

RDBMS :

Database Name Company

MySQL Sun Micro system (Oracle) : 8.x

Oracle Oracle

Db2 IBM

Sql Server 2020 Micro soft

Application Layer to interact with the Database Server

Console (Command prompt) or GUI

SQL : Structure Query Language :

SQL divided into 5 sub category.

1. DRL or DQL (Data Retrieval/Query Language)
2. DDL (Data Definition Language)
3. DML (Data Manipulation Language)
4. DCL (Data Control Language)
5. TCL (Transactional Control Language).

80 to 85% query is common in all RDBMS database.

MySQL : by default username in MySQL : root

show databases : to display all database available in MySQL Database.

use databaseName : to switch or move inside a database.

create database databaseName: This command is use to create the database

show tables; This command is use to display all tables present in that database.

Oracle : by default oracle username and password

scott and tiger.

show databases

in Oracle database the login name itself is database consider.

scott is database in my machine for oracle sever.

Oracle database provide 1 pre-defined table ie tab. This table help to display all pre-defined or user-defined table name available in your account.

This query is use to find all table available in our account.

select \* from tab;

select \* from tableName;

employees

departments

job\_history

jobs

locations

select \* from tableName;

RDBMS is case insensitive but values are case sensitive.

Day 4

08-09-2021

Select \* from tab; This command is use to display all pre-defined or user-defined table present in you’re a account.

Select \* from tableName;

Select \* from employees;

Select \* from departments

Select \* from locations;

Select \* from employees;

DRL or DQL : Data query or retrieval language

Select clause

select columnname, columnName,columnName from tableName;

desc employees; This command is use to provide the table structure.

Column alias : temporary name for the column

Select columnName as columnAlias from tableName;

select employee\_id as emp\_id,first\_name as fname,salary as emp\_salary from employees;

or

select employee\_id emp\_id,first\_name fname,salary emp\_salary from employees;

maths operation on column if column contains numerical value.

select employee\_id,first\_name,salary,salary+500 as gross\_salary from employees;

|| This operator is use to combine two columns as one column.

select first\_name||last\_name from employees;

Display the output as

Employee\_id, Full\_name, Basic\_Salary,HRA, DA, PF, Gross\_Salary

Salary is basic salary

Hra is 10% on salary

Da is 5% on salary

Pf is 7% on salary

Gross salary is hra + da –pf +salary

Where clause : where clause is use to filter the data.

Relational operators

select \* from tableName where columname RO value;

>

>=

<

<=

=

!= or <>

select \* from employees where salary > 15000;

select \* from employees where employee\_id < 150;

select \* from employees where salary = 17000;

select \* from employees where hire\_date='28-SEP-97';

select salary from employees where first\_name = 'Steven';

between operator (range selection)

select \* from tableName where columnName between minValue and maxValue

select \* from employees where salary between 5000 and 10000

select \* from employees where employee\_id between 110 and 120

oracle by default date format consider as dd-mon-yy

We can use between operator for date column also.

In operator (in operator is use to check more than one specific value).

Select \* from employees where employee\_id in(110,116,112,118);

Select \* from employees where job\_id in(‘IT\_PROG’,’ST\_MAN’,’PU\_CLERK’);

Like operator

Select \* from employees where first\_name=’Steven’

Select \* from employees where first\_name like ‘Steven’;

% zero or 1 or many

select \* from employees where first\_name like '%n' : end n character

select \* from employees where first\_name like 'A%' : start with A character

select \* from employees where first\_name like '%e%' : contains e character

select \* from employees where first\_name like '\_e%' : 1st character can be anything and 2nd must be 2 character.

not null : column doesn’t contains any value.

select first\_name,salary,commission\_pct from employees where commission\_pct is null;

and, or, not

and means both the condition must be true

or means any one condition must be true

not if condition true it make it false and vice-versa.

Select \* from employees where employee\_id=100 and first\_name like ‘Steven’;

Select \* from employees where job\_id like ‘IT\_PROG’ and salary > 10000;

Select \* from employees where job\_id like ‘IT\_PROG’ or salary > 10000;

Select \* from employees where first\_name not like ‘Steven’;

Select \* from employees where job\_id not like ‘IT\_PROG’;

Select \* from emplyees where salary not between 5000 and 10000

Select \* from employees where hire\_date not in(’01-mar-90’)

Select \* from employees where commission\_pct is not null

Order by clause : this clause is use to do the sorting may be ascending or descending order.

select employee\_id,first\_name,salary from employees order by salary asc;

select employee\_id,first\_name,salary from employees order by salary desc;

select employee\_id,first\_name,salary from employees order by salary;

we can do multi sorting

select employee\_id,first\_name,salary from employees order by salary asc;

select employee\_id,first\_name,salary from employees order by salary desc;

select employee\_id,first\_name,job\_id,salary from employees order by job\_id, salary desc;

Join :

Create table table1(srno int, name varchar(10), salary float)

Create table table2(accno int, name varchar(10), amount float)

Join is use to retrieve the records from one or more than one table with out without conditions.

1. Cartesian product :

select srno,salary, accno, amount from table1, table2;

output is m \*n

select srno,salary, table1.name,accno, amount from table1, table2;

select srno,salary, table2.name,accno, amount from table1, table2;

select srno,salary, table1.name,table2.name,accno, amount from table1, table2;

table alias

select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1, table2 t2;

Equi – join : display common records present in both the tables.

Using where clause

select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1, table2 t2 where t1.name = t2.name;

equi join also known as inner join.

On clause

select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1 inner join table2 t2 on t1.name = t2.name;

select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1 join table2 t2 on t1.name = t2.name;

outer join

left outer join : common + left side or first table remaining records

where clause :

select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1, table2 t2 where t1.name = t2.name(+);

on clause

select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1 left outer join table2 t2 on t1.name = t2.name;

right outer join : common + right side or second table remaining records.

Where clause

select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1, table2 t2 where t1.name(+) = t2.name;

on clause

select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1 right outer join table2 t2 on t1.name = t2.name;

full join : common + left and right side both table remaining records.

select t1.srno,t1.name,t1.salary,t2.accno,t2.name,t2.amount from table1 t1, table2 t2 where t1.name(+) = t2.name;

Employees 10 employees

6 employee working in projects.

Projects 5 projects

3 project started.

Equi join and inner join : display all employee details in project with project details.

Left outer join : display all employee details with the people working in project as well as not working in projects.

Right outer join : display project details with employee details if any employee working in that project or not.

Equi join

select emp.first\_name,emp.salary,dept.department\_name from employees emp,departments dept where emp.department\_id=dept.department\_id;

left outer join

select emp.first\_name,emp.salary,dept.department\_name from employees emp,departments dept where emp.department\_id=dept.department\_id(+);

right outer join

select emp.first\_name,emp.salary,dept.department\_name from employees emp,departments dept where emp.department\_id(+)=dept.department\_id;

self join : joining the same table itself is known as self join

while join concept it is not mandatory column name must be same but it contains same type of values with data types.

select emp1.first\_name,emp1.job\_id,emp2.first\_name from employees emp1, employees emp2 where emp1.manager\_id= emp2.employee\_id;

Reports

Self join

Lex is a Ad\_VP Report to Steven, He/She is Pred.

Self join with equi join : makes sure records not repeat.

Lex is a Administration Vice President Report to Steven, He/She is President.

Alexzander is a Programmer Report to Steven, Report to He/She is Administration Vice President.

Oracle functions

Function is to write set of instruction to perform specific task.

Function mainly divided into two types.

1. Pre-defined function
2. User-defined function : Pl SQL

Oracle provide lot of pre-defined function

2 types

1. Single row function : This function apply the function functionality to each records individually and return every row of tables.
2. Multi row function : This function apply the function functionality to all table records or base upon the group and it return only one single output.

Every function take one or more than one parameter and return the value.

String function

select first\_name,upper(first\_name) from employees;

select \* from dual; This function is use to check the function functionality.

select upper('raj'), lower('RAJ'), initcap('raj') from dual;

select substr('rajdeep',2),substr('rajdeep',2,5) from dual;

select length('raj deep') from dual;

number function

select round(345.678,0) from dual;

select round(345.678,1) from dual;

select round(345.678,2) from dual;

select round(345.678,3) from dual;

select round(344.678,-1) from dual;

select round(344.678,-2) from dual;

select round(644.678,-3) from dual;

select trunc(345.678,0) from dual;

select trunc(345.678,1) from dual;

select trunc(345.678,2) from dual;

select trunc(345.678,3) from dual;

select trunc(344.678,-1) from dual;

select trunc(344.678,-2) from dual;

select trunc(644.678,-3) from dual;

select ceil(10.1), ceil(10.9), floor(10.1), floor(10.9) from dual;

date function

select sysdate from dual;

select ceil(10.1), ceil(10.9), floor(10.1), floor(10.9) from dual;

select sysdate from dual;

select add\_months(sysdate, 2) from dual;

select next\_day(sysdate,'Mon') from dual;

select months\_between(sysdate, '01-jan-21') from dual;

1. display all employee first\_name, salary, numberofyearexp from employees
2. display only those employee details whose year of experience is > 20 employee first\_name, salary, numberofyearexp from employees

DDL and DML

Day 5

09-09-2021

Multi row functions

Multi row function also known as aggregate functions.

Sum, max, min, avg and count

Multi row function combine more than one records base upon the groups. By default the whole table itself is consider as one group. We can make sub group base upon some property or attribute.

select sum(salary) from employees;

select max(salary) from employees;

select min(salary) from employees;

select avg(salary) from employees;

select count(manager\_id) from employees

select count(commission\_pct) from employees;

select count(employee\_id) from employees;

select count(\*) from employees;

select first\_name,salary,commission\_pct from employees;

select first\_name,salary,nvl(commission\_pct,0) from employees;

select count(nvl(commission\_pct,0)) from employees;

group by clause

select sum(salary) from employees group by job\_id;

select sum(salary) from employees group by department\_id;

select sum(salary) from employees group by manager\_id

select job\_id, sum(salary) from employees group by job\_id;

select job\_id, sum(salary) from employees group by job\_id;

select department\_id,sum(salary) from employees group by department\_id;

select department\_id,sum(salary),max(salary) from employees group by department\_id;

having clause : having is like a where clause but having clause must be use after group by clause.

Where clause condition apply for individual records and having clause condition apply for group of records.

select department\_id,sum(salary)from employees group by department\_id having sum(salary) > 50000

select department\_id,sum(salary)from employees group by department\_id having sum(salary) > 50000

select department\_id,sum(salary)from employees where department\_id <> 100 group by department\_id having sum(salary) > 50000

select department\_id,sum(salary)from employees where department\_id <> 100 group by department\_id having sum(salary) > 50000 order by department\_id asc;

sub query

query within another query is known as sub query.

Syntax

Outer query (inner query)

Outer query (inner Query (Inner Inner query))

Find the person name whose salary is greater than average salary of all employee working in organization or working in some particular department.

In sub query inner query will execute and it generate the output or result and that output is input for outer query.

Inner query can return single row or multiple row(make sure query must be retrieve single column).

Single row sub query : aggregate operator or condition with primary key.

Relational operator >, <, >=, <=, =, <> or !=

Multi row sub query: in,

relational operator with any

relational operator with all

Inner query if return more than one records I you want to apply relational operator with any or all.

First we have to check inner query maximum and minimum condition values.

Max 🡪12000

Min 🡪 6900

>all : greater than maximum salary of inner query.

select first\_name,salary from employees where salary >all (select salary from employees where department\_id=100);

>any : greater than minimum salary of inner query

select first\_name,salary from employees where salary >any (select salary from employees where department\_id=100);

exists

select \* from employees where EXISTS (select \* from departments where department\_id=1000)

if inner query return records then outer query display the records.

DDL and DML :

Data definition language

Syntax to create the table

select avg(salary) from employees;

select avg(salary) from employees where department\_id=100

select first\_name,salary from employees where salary > 6461

select first\_name salary from employees where salary > (select avg(salary) from employees)

select first\_name from employees where job\_id in (select job\_id from employees where department\_id=90)

select first\_name from employees where department\_id in (select department\_id from departments where location\_id=1400)

desc departments

select \* from departments

select first\_name,salary from employees where salary >any (select salary from employees where department\_id=100);

select \* from employees where EXISTS (select \* from departments where department\_id=1000)

Create table

Syntax

Create table tableName(columnName datatype, columnName datatype)

create table employee(id int, name varchar(10),salary float,dob date)

create table employee(id number(2), name varchar2(10),salary number(10,2),dob date)

DML : Data Manipulation language

DDL :

drop table name; It remove table with all records.

Alter command

1. add new column to existing table.

alter table employee add desg varchar(2)

1. drop column from existing table

alter table employee drop column dob

1. modify the column data type size.

alter table employee modify desg varchar(10)

DML

Insert

Update :

Update tableName set columnName = value;

update employee set salary = 45000

Above query update all records but if you want to update with condition then we have to use where clause.

update employee set salary = 35000 where id =100;

update employee set salary = 37000 where name like 'Ravi';

both the condition must be satisfy.

update employee set salary = 40000 where id = 102 and name like =’Ramesh’;

if any condition satisfied

update employee set salary = 40000 where id = 102 or name like =’Ramesh’;

if we want to update more than value with one or more condition

update employee set salary = 42000, name = ‘Raj Deep’ where id = 100;

delete query :

delete from tableName; : it delete all records.

delete from employee

Delete with specific conditions.

delete from employee where id = 100

delete from employee where id = 101 and name like ‘Raj’

delete from employee where id = 101 or name like ‘Raj’

This is query is part of DDL

truncate table tableName; it remove all records but maintain the table structure.

delete, drop and truncate difference

truncate and drop is a part of DDL

drop : it remove table structure as well as all records.

Truncate : it remove all records but maintain the table structure.

With truncate we can’t use where clause.

Truncate we can’t use rollback (TCL command).

delete is part of DML

delete remove all the records if we fire query without conditions.

We can use where clause.

If we deleted record using delete query we can rollback with TCL concept (rollback and commit).

13-09-2021

**Keys :**

A key can be apply for attribute (column or fields) or combination of more than one fields that is used to identity unique records.

Some keys are concept and some key we can apply in table

empId,fname,lname,age,phonenumber, accno, typeofaccount, amount, pancard, cusid, custName, custName,mgrid, mname, projected, typeofproject, clientid, client,startdate, enddate etc.

Super key : an attribute or a combination of more than one attribute that is used to identify the records uniquely.

Empid

Pancard,

Accno

Custid,

Mgrid

Clientid

stdId

empid, fname

empid,fname,lname

empid,accnumber

empid,pancardId

empid,pancardid,accnumber

Candidate key : it can be defined as minimum super key or irreducible super key is known as Candidate key.

Empid PK

Pancard,

Accno

Custid,

Mgrid

Clientid

stdId

Primary key : primary key is a type of candidate key that is use by the database designer for unique identification of each row in the table.

In single table we can make only one column can be primary key.

If the column is a primary it doesn’t allow duplicate as well as doesn’t allow null value.

Unique key : in table we can make more than one column as unique key. Unique key doesn’t allow duplicate but it can allow by default null value.

Foreign key : A foreign key is an attribute which is use to link or refer to primary key of same table or different table.

If column is a foreign key it allow only those value which present in primary key.

Foreign key can allow duplicate value.

It can allow null value.

In single table we can create more than one column as foreign key.

Composite primary key :if we use multiple column or attribute which is use to create the primary key that key is know as composite primary key.

Both the column independently allow duplicate and null value. but the combination must be unique ness.

Alternate key : alternate key can be any of the candidate key except primary key.

Surrogate key : This key is use to insert unique records for the primary key.

Ie mysql auto\_increment and in Oracle database we have to use sequence.

Constraints :

Constraints is use to restrict the use or programmer not to insert invalid data.

Some constraints we can use in table level.

Some constraints we can use in column level

Some constraints we can use column level and table level.

EmployeeDetails

PK not null >21 25000 to 50000 phnumber default GWG

Empid FName LName Age Salary unique

**Relationship :**

4 types of relationship

One – primary key and many – foreign key

1. one – to – many Trainer and Student
2. many – to –one Employee – Project
3. one – to – one Person – Passport -🡪 shared primary key.
4. many – to – many Students --- Technologies.

Trainer primary table

PK

TId TName Tech

Student secondary table

PK FK

SId SName Age TSId

Students

PK

SId SName

1. Raj
2. Ravi
3. Ramesh

Technologies

PK

TId Tech

100 C

101 C++

102 Java

103 Python

Student\_tech

CPK

SId TId

1 100

1 102

2 102

3 100

3 101

3 102

3 103

PL SQL : Procedure Language or Programming Language on SQL.

Using SQL we can execute only one query at time. May the query retrieve or effect more than one records.

If we want to execute more than one query with terms and conditions we have to depends upon the other programing language like Pro \* C or Java or Python.

Pl SQL block

2 types of block

1. Anonymous block
2. Named block
   1. Function
   2. Procedure or stored procedure.

Anonymous block

keywords

declare optional

variable declaration

begin

sql and non sql code.

exception option

end;

begin

end;

variable : Pl SQL provide variable concept which help to store the value and value can change during the execution of a program.

Scalar variable : this variable is used to store only single value.

datatype variableName;

datatype variableName := value;

pl sql program to declare the variable and display the value

declare

n number(10);

m number(10) := 100;

fname varchar(10) := 'Raj Deep';

dob date :='20-mar-21';

begin

dbms\_output.put\_line(n);

dbms\_output.put\_line(m);

dbms\_output.put\_line('The value of n is '||n);

dbms\_output.put\_line('The value of m is '||m);

dbms\_output.put\_line('Name is '||fname);

dbms\_output.put\_line('DOB is '||dob);

end;

select query we can’t use direct inside Pl SQL block

so if we want to use select clause in pl sql we have to take the help of into clause and make sure the query retrieve only one records ie with primary key.

Select tablecolumnName1, tablecolumnName2 …. Into variableName1,variableName2 from tableName where primary key conditions.

While retrieving records from user-defined table or pre-defined table in Pl SQL we have to keep the track the column contains what type of data types as well as size of data types.

Variable anchor type

Syntax

variableName tableName.columnName%type

if statement in pl sql

if else

if else if

switch using case

looping

Scalar data type : it is used to store only one value.

number, string, date etc.

Pl SQL Composite data type: it is use to store more than one value of same or different types.

2 types

1. Collection : same type of values.
2. Records : different type of values.

Collection is type of composite data type which is use to store same type values and we can access using index position.

1. Varray : The varray is in short we can say array. The number of elements going to store in varray must be known at the time of its declaration.
2. Table : it is also known as a associate array. It is use to store the information the form key-value pairs. Each key should be unique index. The data type of key can be integer or string type.

varray Syntax

type arraytypename is varray(5) of varchar(10);

type is a keyword, arraytrrypename is a user-defined data type , is keyword varray is a pre-defined object with size of keyword and varchar datatype with size.

variablename arraytypename;

table syntax

type tableRefName is table of varchar(10) index by varchar(10);

names tableRefName;

records : it is a type of composite data type which is used to store different type of values(more than different type of scalar values).

Syntax to create the reference

type record\_ref\_name is record(

variableName datatype,

variableName datatype

);

record\_variableName record\_ref\_name

declare

v\_id number(10):=100;

type emp\_record\_ref is record(

v\_name varchar(10),

v\_salary number(10,2)

);

emp\_record emp\_record\_ref;

begin

select first\_name,salary into emp\_record.v\_name,emp\_record.v\_salary from employees where employee\_id = v\_id;

dbms\_output.put\_line('Name is '|| emp\_record.v\_name);

dbms\_output.put\_line('Salary is '|| emp\_record.v\_salary);

end;

rowtype attribute : it is use to copy all columnname as well as datype with size in records variables.

Syntax

record\_name tableName%rowtype;

in this syntax columname consider as record member name if we access through records variable name.

Cursor : A cursor is s temporary memory created when we execute any sql query within a PL SQL block. Cursor hold the information about DML and Select query. A cursor can hold more than one row but we can process only one row at time. The set of row where cursor hold is known as active set.

2 types of cursor

1. Implicit cursor
2. Explicit cursor

Implicit cursor start with sql followed by pre-defined attribute ie

%found

%notfound

%rowcount

%open

Etc

Explicit cursor : This type of cursor mainly use with select query.

Syntax to create the Explicit cursor

cursor cursor\_name is select clause

4 steps

1. Create the cursor using syntax in declare section
2. Then open the cursor
3. Then fetch the records
4. Then close the cursor

Exception Handling :

Exception is a type of error which occurs when unexpected things happened during the execution of a programs. If any exception generate program terminate abnormally. So Pl SQL provided great features ie exception handling so rather than program terminate abnormally we can display appropriate message to display that type of exception generated.

Syntax

declare

begin

sql and pl sql code

exception

when ex\_name1 then

when ex\_name2 then

when others then

end;

In Pl SQL Exception message divided into three parts

Name of type of exception

Error code

A message

In PL SQL exception are divided into three parts

Pre-defined exception with pre-defined error code.

Zero\_divide

Too-many\_rows

Record\_not\_found

User-defined exception with pre-defined error code.

Primary key with error code -1

User-defined exception with error code within some range.

The error code range must be -20000 to -20999.

15-09-2021

Anonymous block

declare

begin

exception

end;

PL SQL Named block

1. Stored procedure or procedure
2. Function

Store procedure or procedure

create procedure procedureName

as

begin

set of statement ie sql or non sql

end procedureName;

after created procedure successfully we have to call this procedure using procedure name.

calling can be through another procedure or anonymous block.

Procedure with passing parameter mode (by default mode is in mode consider)

1. In parameter : pass value or read only
2. Out parameter : return value or write only
3. In out parameter : pass and return read and write both.

Note : don’t provide size for parameter variable dynamically allocate the memory.

Function : function is also known as named block.

Function must be return one value using return keyword mandatory.

Procedure may or may return with out parameter

But function must be return one value mandatory with return keyword.

Syntax to create the function

create or replace function functionName

return datatype

localvarible declaration;

begin

return value;

end functionName;

Trigger : trigger is a type of special store procedure which get call automatically or fire when we execute any DML(Insert/Delete/Update) operation on table.

Syntax

Create or replace trigger triggerName

Before/after

Insert/delete/update

on tableName

for each row

begin

end triggerName;

oracle database provide sequence concept which help to auto generate the number one by one

create sequence sequenceName

sequenceName.nextval-🡪 increment the value by previous value.

sequenceName.currvalue 🡪 it insert or display current value.

trigger provided pre-defined property :old and :new

package : package is a collection of variable, records, cursor, function and procedure. Package is when two procedure or function etc have same name different purpose.

Raj and 123

HDFC withdraw

HSBC withdraw

In Pl SQL package divided into 2 parts

Package header : declaration of all member

Package body : implementation of member

16-09-2021

Pointer

Dynamic memory allocation

malloc and calloc

Cobol

Fortan

Pascal

C is a basic structure programming language developed in 1972.

So we can use C for general purpose as well as specific purpose.

Structure of C programming language

Pre processing instruction

Global declaration -🡪 operation

Pre-defined function()

{

Declaration section

Set of Statements

}

User-defined functions() {

}

GCC (GNU C Compiler).

Connect the unix terminal

Create the folder as

mkdir CPrograms

cd CPrograms

vi welcome.c

create the program

then compile the program using command as

gcc programname.c

after compile by default one file create ie. a.out or a.exe

to run the program we have use the command as

a.exe -🡪window

./a.out 🡪 unix

If you want to create our out or exe file

Then run the command as

gcc welcome.c -o welcome

Then run the program using command

./welcome

Data types : Data type is a type of data which tells what type of data it can hold.

1. Basic data types : integer, float, char and double.
2. Derived data types : array pointer, structure and union.

Variable declaration syntax

datatype variablename;

datatype variableName= value;

Taking the value through keyboards using scanf() functions

Operator :

Arithmetic Operator : +, -, \*, /, %

Logical operator &&, ||, !

Conditional operator : >, >=, <, <=, ==, !=

Assignment operator : =

Increment and decrement operator : ++, --

Bitwise operator : &, ! etc

Ternary operator : condition ? true:false;

If statement

1. Simple if

If(condition) {

}

1. If else

if(condition) {

}else {

}

1. Nested if : If within another if.

If(condition) {

If(condition) {

}else {

}

}else {

If(condition){

}

}

1. If else if or if ladder

If(condition) {

}else if(condition) {

}else if (condition) {

}else {

}

switch(variable) {

case label: block1;

break;

case label: block1;

break;

case label: block1;

break;

default : wrong block

break;

}

switch, case, break and default are keywords.

Variable must be type of int or char family.

Looping

It is use to execute set of statement again and again till the condition become false.

Three types of loop

while loop

do while loop

for loop

any type of loop

initialization : start and end position

condition : make sure it must be true

increment and decrement

int a;

array : array is used to store the same type of value.

syntax

datatype arrayName[size];

int abc[10];

We can retrieve the array value using index position start form 0.

function : function is use to write the set of instruction to perform a specific task.

With help of function we can do re-usability the code.

returnType functionName(parameterList) {

function body;

}

Void ; void means no return type

Int, float, char, double , array return type.

4 types

1. Function no passing parameter and no return type.
2. Function with passing parameter but no return type.
3. Function passing parameter and return value
4. Function with no passing parameter but return the value

Pointer : Pointer is a normal variable which is used to store the address of another variable.

Syntax for pointer variable declaration

datatype \*pointerVariable;

so using pointer variable can do some maths operation on address with help of pointer variable.

17-09-2021

Maths operation on address through pointer.

Return more than value using pointer ie call by value and call by reference.

Call by value

1. A copy of value is passed to function
2. If we do any changes inside the function for parameter variable it doesn’t reflect in another or main function.
3. In call by value concept separate memory get created for actual and formal arguments.

Call by reference

1. An address of value is passed to function.
2. Changes made inside a function is reflected outside the function also.
3. In this concept actual and format arguments will be created in same memory location.

C String :

Combination of more than one character is known as string. In C language every string end with null character \n.

Structure :

Structure is a user-defined data type which is used to store different type of values.

Syntax

struct structureName {

datatype variableName;

datatype varableName;

};

This structure syntax we can do inside main function or outside.

The variable which is part of structure is known as structure member. So we can’t access structure member directly.

To access structure member we have to create the structure variable.

Malloc and calloc : dynamic memory allocation.

Array or structure of array is known fixed memory size.

In C language we can create dynamic memory using malloc and calloc function or library

Syntax

variableName = (datatype\*)malloc(n\*sizeof(datatype))

enum

static keyword

Java : deep and shallow copy

20-09-2021

C

OOPs language

object : it is any real world entity.

Properties or state--🡪 have -🡪 variable / fields

Person

Behaviour -🡪do/does ---🡪 function / methods

Bank

Animal

Pen

Computer

Table

class : it is a blue print of object or template of object.

C++

C++ = C+ OOPs concept

C++ is a partial object oriented programming language.

C++ it is a platform dependent programming language.

Java

Initial Name of Java is Oak. 1991

In Nov 1995 rename from Oak to Java.

Java is developed by James gosling and team.

It was belong to sun micro system now part of Oracle.

Version 1.0 Java 16.

Java was open source but not now it is commercial.

Java 8.

Syntax

class className {

fields;

methods;

}

className must be follow pascal naming rules.

If class name contains one world then first letter must be in upper case.

If it contains more than one world each world first letter upper case.

Test

Employee

EmployeeInfo

ManagerDetails

Demo.java

class Demo {

public static void main(String args[]) {

System.out.println("Welcome to Java");

}

}

You have to save the program with name as

Demo.java

To compile the program we have to use the command as

javac Demo.java : After compile successfully to run the program we have to use the command as

java Demo

Welcome program

class Demo {

public static void main(String args[]) {

System.out.println("Welcome to Java");

}

}

Welcome message with print, println and printf

class Demo {

public static void main(String args[]) {

System.out.println("Welcome to Java");

System.out.print("Welcome to Java");

System.out.printf("Welcome to Java");

}

}

**Data Types :**

Types of data types

2 types

1. Primitive data types : it is used to store only value :

8 types

1. byte 1 byte
2. short 2 byte
3. int 4 byte
4. long : without decimal number 8 byte
5. float 4 byte
6. double : with decimal number 8 byte
7. char : any single character 2 byte
8. boolean : true or false 1 bit
9. Non primitive or reference data types : it is used to store value as well as another data type reference.

4 types

1. Array
2. class : pre-defined or user-defined class
3. interface : pre-defined or user-defined interface
4. enum

variable declaration and display the value of variable

class Demo {

public static void main(String args[]) {

int a=10;

System.out.println(a);

System.out.println("The value of a is "+a);

}

}

Type casting :

Converting one data type to another data type is known as type casting.

2 types

1. implicit type casting
2. explicit type casting

int family

----------------------------🡪 implicit type casting ----------------🡪

byte short int long

🡨-----------------------explicit type casting ---------------------------

(type)variabeName;

class Demo {

public static void main(String args[]) {

byte a=10;

short b =a;

System.out.println(a);

System.out.println(b);

short c =20;

byte d = (byte)c; // explicit type casting

System.out.println(c);

System.out.println(d);

}

}

int to float or double family

by default every decimal number is consider as double and double by default size is 8 byte so we can’t store the double value in float.

Example

class Demo {

public static void main(String args[]) {

int a=10;

float b=a;

System.out.println(a);

System.out.println(b);

//float c = (float)20.20;

float c = 20.20f;

int d = (int)c;

System.out.println(c);

System.out.println(d);

}

}

Operator :

instanceOf

if statement

if else

nested if

if else if

switch statement : variable type can be string type.

Looping

While loop

Do while loop

For loop

Taking the value through keyboards in Java

1. Scanner class
2. BufferedReader
3. DataInputStream
4. Command line argument

Scanner is a pre-defined class part of util package.

Syntax to create the scanner class object.

Scanner obj = new Scanner(System.in);

Scanner class contains lot of pre defined methods

nextByte();

nextShort();

nextInt();

nextLong();

nextDouble();

nextFloat();

nextBoolean();

String name = obj.next(); it is use to receive only one world

String fname = obj.nextLine() it is use to receive more than one world. This method terminator is enter key.

//import java.util.\*;

import java.util.Scanner;

class Demo {

public static void main(String args[]) {

String msg = "Welcome to Java Training";

System.out.println(msg);

Scanner obj = new Scanner(System.in);

System.out.println("Enter the id");

int id = obj.nextInt();

obj.nextLine(); // it use to hold the enter the key

System.out.println("Enter the name");

String name = obj.nextLine();

System.out.println("id is "+id);

System.out.println("name is "+name);

}

}

Reference data type

Array : array is used to store more than one value of same data types.

Syntax

datatype arrayName[];

int abc[10]; C or C++

int abc[]; Java

array declaration in java

class Demo {

public static void main(String args[]) {

int abc[];

int []abc1;

int abc2 [];

int[] abc3;

}

}

for each loop and enhanced loop

for(datatype variableName : arrayName) {

}

class Demo {

public static void main(String args[]) {

int num[]={10,20,30,40,50,60,100,140,10,56,78,999};

System.out.println(num[0]);

System.out.println(num[1]);

System.out.println("Size of array is "+num.length);

System.out.println("Using for loop");

for(int i=0;i<num.length;i++) {

System.out.println(num[i]);

}

System.out.println("Using for each loop or enhanced loop");

for(int n : num) {

System.out.println(n);

}

}

}

Creating memory size for array

datatype arrayName[]=new datatype[size];

int num[]=new int[10];

21-09-2021

OOPs (Object oriented Programming system).

object : any real world entity

properties or state 🡪 have

Person

Behaviour 🡪do/does

Car

Bank

Animal

class : class is a blue print of object or template of object or user-defined data types which help to create the object.

Object creation syntax

className objectRefName = new className();

objetRefName.method()

objetRefName.variablename= value;

Car class with object concept

class Car {

int wheel;

String color;

float price;

void start() {

System.out.println("Car Start");

}

void appliedGear() {

System.out.println("Applied Gear");

}

void moving() {

System.out.println("Car is moving");

}

void stop() {

System.out.println("Car Stop");

}

}

class CarTest {

public static void main(String args[]) {

System.out.println("Main method");

Car innova = new Car(); // heap memory

innova.start();

innova.stop();

}

}

Type of variable or fields

In java variable are divided into three types

1. Instance variable :
   1. The variable which declare outside class but inside a method is known as instance variable.
   2. All instance variable hold default value with respective their data types. like int family 0, float family 0.0, char space, Boolean false, String null.
   3. We can use all instance variable directly inside a all methods but method must be part of same class and it must be non-static method.
2. Local variable
   1. The variable which declare inside a method is known as local variable.
   2. Local variable doesn’t hold default value we have to initialize.
   3. The scope of the variable within that block where it declare.
3. Static variable

Instance variable and local variable example

class Car {

int wheel;

String color;

float price;

void start() {

int temp=0;

System.out.println("Car Start");

System.out.println("Wheel"+wheel);

System.out.println("Color"+color);

System.out.println("price"+price);

System.out.println("temp "+temp);

}

void stop() {

String msg="Welcome";

System.out.println("Car Stop");

System.out.println("Wheel"+wheel);

System.out.println("Color"+color);

System.out.println("price"+price);

System.out.println("msg "+msg);

}

}

class CarTest {

public static void main(String args[]) {

System.out.println("Main method");

Car innova = new Car(); // heap memory

innova.start();

innova.stop();

}

}

Constructor : it is a type of special method which help to create the memory.

Pts

1. Constructor have same name as class itself.
2. Constructor doesn’t contains return type not even void also.
3. Constructor no need to call it will call automatically when we create the object.

Constructor example

class Car {

Car() {

System.out.println("Object created...");

}

void display() {

System.out.println("display method");

}

}

class CarTest {

public static void main(String args[]) {

Car innova = new Car(); // heap memory

innova.display();

innova.display();

innova.display();

innova.display();

Car swift = new Car();

swift.display();

}

}

In the life of the object if we want to perform any task only one time that type of task we have to write inside a constructor.

If we want to perform a task more than one time that type of code we have to write inside methods.

If we doesn’t write any constructor in class by default java compiler provide default constructor. Default constructor is always empty constructor.

If we write explicitly empty or parameterized then there is no default constructor.

Parameterized constructor

class Abc {

int a,b,sum;

Abc() {

a=10;

b=20;

}

Abc(int x, int y) {

a=x;

b=y;

}

void setValue(int x, int y) {

a=x;

b=y;

}

void addNumber() {

sum = a+b;

}

void display() {

System.out.println("Sum is "+sum);

}

}

class Demo {

public static void main(String args[]) {

Abc obj1 = new Abc(); obj1.display(); // sum =0

Abc obj2 = new Abc(); obj2.addNumber(); obj2.display(); // sum =30

Abc obj3 = new Abc(); obj3.addNumber(); obj3.display(); // sum =30

Abc obj4 = new Abc(100,200); obj4.addNumber(); obj4.display(); // sum = 300

Abc obj5 = new Abc(); obj5.setValue(1,2); obj5.setValue(3,4); obj5.addNumber(); obj5.display(); // sum = 7

}

}

Encapsulation : binding or wrapping data and code in a single unit is known as Encapsulation.

Class

class Employee {

String name;

float salary;

void display() {

System.out.println("Name is "+name);

System.out.println("Salary is "+salary);

}

}

class Demo {

public static void main(String args[]) {

Employee emp1 = new Employee();

emp1.salary = 12000;

emp1.name = "Raj";

emp1.display();

}

}

When instance variable and local variable have same name then local variable hide the visibility of instance variable. But if you want to refer the instance variable inside a method we have to use this.instancevaraibleName.

Another encapsulation example

class Employee {

private String name;

private float salary;

void setValue(String name, float salary) {

this.name = name;

//this.salary = salary;

if(salary <0) {

this.salary = 8000;

}else {

this.salary = salary;

}

}

void display() {

System.out.println("Name is "+name);

System.out.println("Salary is "+salary);

}

}

class Demo {

public static void main(String args[]) {

Employee emp1 = new Employee();

//emp1.salary = -12000;

//emp1.name = "Raj";

emp1.setValue("Raj",-12000);

emp1.display();

}

}

Polymorphism : one name many forms or many implementation.

2 types

Compile time run time

Static binding dynamic binding

Early binding late binding

Ex : Ex

Method overloading Method overriding

Method Overloading : The method have same name but different parameter list (type of parameter of list or number of parameter list must be different).

Inheritance : Inheritance is use to inherits or acquire properties and behaviour of old class to new class.

class OldClass { super class, base class or parent class

properties

behaviour

}

class NewClass extends OldClass{ sub class, derived class or child class

properties

behaviour

}

With the help of super class objet we can access only its own properties and behaviour. But with the help of sub class object we can access its own as well as super class properties and behaviour.

Simple example of inheritance

class A {

void dis1() {

System.out.println("A class dis1 method");

}

}

class B extends A{

void dis2() {

System.out.println("B class dis2 method");

}

}

class Demo {

public static void main(String args[]) {

A obj1 = new A();

B obj2 = new B();

obj1.dis1();

obj2.dis2();

obj2.dis1();

}

}

Types of inheritance

1. Single inheritance : one super class and one sub class

class A { }

class B extends A { }

1. Multilevel inheritance : one super class and n number of sub classes connected one by one

class A { }

class B extends A {}

class C extends B{}

class D extends C {}

1. Hierarchical inheritance : one super class and n number of sub classes directly connected to super class.

class A { }

class B extends A {}

class C extends A {}

class D extends A {}

1. Multiple inheritance : more than one super class and one sub class.

class A { }

class B { }

class C extends A,B { } : but Java doesn’t support this type of inheritance using class. it can support using interface.

Oops relationship :

1. Is a relationship : Inheritance : using classes or interfaces.
2. Has a relationship : inside one class we are creating another class object.

Manager/ Developer Is a Employee

class Manager extends Employee {

}

class Employee {

Address add = new Address();

}

class Developer extends Employee{

}

class Address {

}

Super class must be generic and sub class must be specific.

Has a relationship

1. Association
2. Aggregation
3. Composition

class A {

B obj zero or 1 or many

}

class B {

A obj zero or 1 or many

}

To make association we have to create either side one or many object.

class Manager {

Address ladd, padd 1 or many

}

class Address {

}

It is a type of association but it is known as week association. So it is known as aggregation

class Student {

StudentHistory sh 1 or many

}

class StudentHistory {

}

It is a type of association but it is known as strong association. So it is known as composition.

Hierarchical inheritance example

class Bike {

void speed() {

System.out.println("60km/hr");

}

}

class Honda extends Bike{

void color() {

System.out.println("Red");

}

}

class Pulsar extends Bike{

void color() {

System.out.println("Black");

}

}

class Demo {

public static void main(String args[]) {

Honda hh = new Honda(); hh.color(); hh.speed();

Pulsar pu = new Pulsar(); pu.color(); pu.speed();

}

}

Method overriding

The method have same name and same method signature (ie type of parameter list, number of parameter list and return type must be same).

Annotation : annotation is known as a meta-data. Meta-data means data about data.

Java provide lot pre-defined annotation. All annotation start with @ followed by name of the annotation.

Few annotation we can use in class level or method level or property level.

@Override

Method overriding and super.methodName() example

class Bike {

void speed() {

System.out.println("60km/hr");

}

}

class Honda extends Bike{ // re-usability

void color() {

System.out.println("Red");

}

}

class Pulsar extends Bike{

@Override

void speed() { // method overriding

System.out.println("90km/hr");

}

void color() {

System.out.println("Black");

}

}

class Tvs extends Bike{

@Override

void speed() {

super.speed(); // calling super class speed() method. so merge the code

System.out.println("20km/hr");

}

void color() {

System.out.println("gray");

}

}

class Demo {

public static void main(String args[]) {

Honda hh = new Honda(); hh.color(); hh.speed();

Pulsar pu = new Pulsar(); pu.color(); pu.speed();

Tvs tv = new Tvs(); tv.color(); tv.speed();

}

}

abstract :

1. abstract is a keyword we can use with method and class but not with variable.
2. abstract method : the method without body or incomplete method or without curly braces

syntax

abstract returnName methodName(parameterList);

1. if class contains any abstract method then we have to declare the class as a abstract class

syntax

abstract class classname {

}

1. which ever class extends abstract class that class must be provide the body for all abstract methods mandatory. That class can ignore if that class itself is a abstract class.
2. abstract class we can’t create the object.
3. Abstract class can contains normal as well as abstract methods. It can contains zero or 1 or many abstract methods.
4. Abstract class can contains default constructor as well as we can write parametrized constructor. (constructor use to do initialization).

Abstract example

abstract class Bike {

int wheel;

Bike() {

wheel =4;

}

abstract void speed();

}

abstract class Honda extends Bike{

void color() {

System.out.println("Red");

}

}

class Demo {

public static void main(String args[]) {

//Honda hh = new Honda(); hh.color(); hh.speed();

}

}

static keyword

1. static keyword we can use with variable and method but not with class(if class is inner class we can use static keyword).
2. static variable : if variable is a static we can assign the value for that variable with help of class name. Even though we can assign the value for that variable through object also.
3. If method is static we can call that method with the help of class name. Even though we can call static method with help of object also.
4. Inside a non static method we can access static as well as non static variable directly.
5. But inside static method we can access only static variable directly we can’t access non static variable if you want to access we have to create the object of that class mandatory.

Static example

class Abc {

int a;

static int b;

void dis1() {

System.out.println("Non static method");

System.out.println("a "+a);

System.out.println("b "+b);

}

static void dis2() {

System.out.println("static method");

Abc obj1 = new Abc();

System.out.println("a "+obj1. a);

System.out.println("b "+b);

}

}

class Demo {

public static void main(String args[]) {

Abc.b=10;

Abc obj1 = new Abc();

obj1.b=20;

obj1.a=30;

Abc.dis2();

obj1.dis1();

Abc.dis2();

}

}

Static memory and heap memory

Every class we will get only one static memory. Static memory always belong to class. heap memory always belong to object. Number of object creation equal to number of heap memory will create.



class Abc {

int a;

static int b;

void dis1() {

System.out.println("Non static method");

System.out.println("a "+a);

System.out.println("b "+b);

}

}

class Demo {

public static void main(String args[]) {

Abc obj1 = new Abc();

Abc obj2 = new Abc();

obj1.a=10;

obj1.b=20;

Abc.b=30;

obj2.a=40;

Abc.b=50;

obj2.b=60;

obj1.dis1(); // a=10 b=60

obj2.dis1(); //a=40 b=60

}

}

Static is like a global to all objects.

Your id, name, salary must be instance variable, working in one project, under one client, under manager they must be static.

final keyword

1. final keyword we can use with variable, method and class.
2. final variable : to declare constant variable we have to use final keyword with variable.
   1. final int A=10;
3. final method : if method is final we can’t override that method but we can use it.
   1. final void dis() {

}

1. final class : if class is final we can’t inherits that class or can’t extends.
   1. final class classname {

}

Final example

final class A {

final void dis1() {

System.out.println("dis1 method");

}

}

class B extends A {

/\*void dis1() {

System.out.println("dis1 method override by B class");

}\*/

}

class Demo {

public static void main(String args[]) {

final int A=10;

System.out.println(A);

//A=20;

B obj1 = new B();

obj1.dis1();

}

}

interface : interface is known as 100% pure abstract class till java7 version.

It is also known as reference data types.

Syntax

Interface interfaceName{

fields;

methods;

}

In interface all fields are public static and final by default.

And all methods are public and abstract by default.

interface A {

public static final int A=10;

int B=20;

public abstract void dis1();

void dis2();

}

Like class one interface can extends another interface. Interface can extends more than one interface.

Class always implements interface. Class can implements more than one interface.

Access specifiers rules while overriding the methods

Super class / interface Sub

public public

protected public

protected

default (nothing) public

protected

default (nothing)

private we can’t override

class and interface example

interface A {

int M=10;

void dis1();

}

interface B {

int N=20;

void dis2();

}

interface C extends A,B{ // using interface we can achieve multiple inheritance.

int O=30;

void dis3();

}

class D implements A,B {

public void dis1() {

System.out.println("A interface method");

}

public void dis2() {

System.out.println("B interface method");

}

}

class Demo {

public static void main(String args[]) {

D obj1 = new D();

obj1.dis1();

obj1.dis2();

}

}

Difference between abstract class and interface

1. interface contains only constant or final variable abstract class can contains normal as well as final variable.
2. Interface contains all abstract methods but abstract class can contains normal as well as abstract methods.
3. Interface can extends more than one interface abstract class can extends only one class.
4. Normal class or abstract class can implements more than one interface but interface can’t extends or implements to class.

Common points

1. Which ever class extends abstract class or implements interface that class must be provide the body for all abstract method belong to that abstract class or interface.
2. We can’t create the object of interface as well as abstract class.

this, this(), super and super()

class A {

int n=10;

}

class B extends A{

int n=20;

void dis1() {

int n=30;

System.out.println("local variable n is "+n);

System.out.println("instance variable n is "+this.n);

System.out.println("super variable n is "+super.n);

}

}

class Demo {

public static void main(String args[]) {

B obj1 = new B();

obj1.dis1();

}

}

**this()** : this() parameter is use to invoke same class constructor or it is use to achieve constructor chaining

this() or this(parameter) we have to use inside a constructor only and it must be first statement inside a constructor.

This() example

class Employee {

Employee() {

this(10); // calling int parameterized constructor

System.out.println("()");

}

Employee(int id) {

this(11,"Ramesh");

System.out.println("(int)");

}

Employee(int id, String name) {

System.out.println("(int,string)");

}

}

class Demo {

public static void main(String args[]) {

Employee emp1 = new Employee();

//Employee emp2 = new Employee(10);

//Employee emp3 = new Employee(11,"Raju");

}

}

Another this()

class Employee {

int id;

String name;

float salary;

Employee() {

this.id = 123;

this.name = "Unknown";

this.salary = 8000;

}

Employee(int id) {

this();

this.id = id;

}

Employee(int id, String name) {

this(id);

this.name = name;

}

Employee(int id, String name, float salary) {

this(id,name);

this.salary = salary;

}

void dis() {

System.out.println("Id is "+id);

System.out.println("Name is "+name);

System.out.println("Salary is "+salary);

}

}

class Demo {

public static void main(String args[]) {

Employee emp1 = new Employee(); emp1.dis();

Employee emp2 = new Employee(100); emp2.dis();

Employee emp3 = new Employee(101,"Ramesh"); emp3.dis();

Employee emp4 = new Employee(102,"Raju",15000); emp4.dis();

}

}

super() :

By default every sub class constructor contains super() as the first statement. It use to call the super class constructor or constructor chaining from sub class to super class. it call always super class empty constructor if we want to call parameterized constructor we can use super(parameter).

super() example

class Employee {

int id;

Employee() {

System.out.println("Employee class object created..");

}

Employee(int id) {

this.id = id;

System.out.println("Employee class object created..with parameter");

}

}

class Manager extends Employee {

Manager() {

super(10);

System.out.println("Manager class object created..");

}

}

class Demo {

public static void main(String args[]) {

Manager mgr = new Manager();

}

}

Run time polymorphism with object creation

1st Example

class A {

void dis1() {

System.out.println("dis1 method part of A class");

}

}

class B extends A {

void dis1() {

System.out.println("dis1 method part Override by B class");

}

void dis2() {

System.out.println("dis2 method part of B class");

}

}

class Demo {

public static void main(String args[]) {

A obj1 = new A(); obj1.dis1();

B obj2 = new B(); obj2.dis1(); obj2.dis2();

//B obj3 = new A(); //creating super class object and sub class reference not possible.

A obj4 = new B(); // creating sub class object and super class reference possible

// with the help of these reference we can call only those method belong to super class

// method overrided at run time.

obj4.dis1(); //obj4.dis2();

}

}

2nd Example

abstract class A {

void dis1();

}

class B extends A {

void dis1() {

System.out.println("dis1 method part Override by B class");

}

void dis2() {

System.out.println("dis2 method part of B class");

}

}

class Demo {

public static void main(String args[]) {

A obj1 = new B(); //creating sub class object and abstract class reference possible.

obj1.dis1();

}

}

**3rd Example**

interface A {

void dis1();

}

class B implements A {

public void dis1() {

System.out.println("dis1 method part Override by B class");

}

void dis2() {

System.out.println("dis2 method part of B class");

}

}

class Demo {

public static void main(String args[]) {

A obj1 = new B(); //creating sub class object and interface reference possible.

obj1.dis1();

}

}

Interface always use to provide the specification and class provide implementation.

Init block and static block

class A {

A() {

System.out.println("A class constructor");

}

{

System.out.println("init block");

}

static {

System.out.println("Static block");

}

void dis1() {

System.out.println("dis1 method");

}

}

class Demo {

public static void main(String args[]) {

A obj1 =new A();

obj1.dis1();

obj1.dis1();

A obj2 = new A();

}

}

package : package is a known as collection of classes and interfaces.

Package mainly divided into two types

1. User-defined package
2. Pre-defined package.

When we write two classes or interface which have same name but different purpose.

Education

School College pg

Attendance Attedance Attendance

Package is like a directory or folder.

Syntax to create the user-defined package

package packageName;

package com;

class Test {

public static void main(String args[]) {

System.out.println("This is simple package program");

}

}



Access specifiers : it is use to expose the visibility of class, method, variable or constructor.

4 types

private : we can use this access specifier with instance variable, static variable, non static method, static method, constructor but can’t use with local variable and class.

scope : within a same class.

default : we can use with all.

scope : within a same package.

protected : we can use this access specifier with instance variable, static variable, non static method, static method, constructor but can’t use with local variable and class.

scope : within a same package other package if it sub class.

public : we can use this access specifier with instance variable, static variable, non static method, static method, constructor and class but can’t use with local variable.

scope : same package as well as other package