**Day 1 : 10/16/20**

emp.txt

Id, name, salary

store employee details

pipe command

ls | mkdir Abc

command | command | command | command

**Day 2 : 10/19/20**

\* Unix Command line

Given a file, say employee.csv, containing data in the following format

empid,initials,department,leave\_available

1000,KK,CTO\_OFFICE,12

2001,MA,HEXAVARSIT,8

1001,IK,CTO\_OFFICE,9

2002,UM,HEXAVARSIT,7

1002,SK,CTO\_OFFICE,13

2003,VR,HEXAVARSIT,6

2004,JK,HEXAVARSIT,6

\* Give an unix command which returns the record with the maximum leave\_available

\* In the above case, the output should be

1002,SK,CTO\_OFFICE,13

$ --- > bash shell

% ---🡪 C shell

echo $SHELL

/bin/csh

/bin/bash

cat /etc/shells

\* Give an unix command which returns one row per department - department, sum of leave\_available of all employees in that department

\* In the above case, the output should be

CTO\_OFFICE,34

HEXAVARSIT,2­­7

**Day 3 :**

simple if

case (switch statement)

1: create the file : customer.txt

If file present don’t create else create it.

2: add customer info cusid,custname,age

3: display all customer details.

4: display specific customer details using custId

5:

**3 days (HTML/CSS/JavaScript)**

https://[www.google.com](http://www.google.com) URL : Uniform resource locator

http : hyper text transfer protocol (secure)

www: world wide web

google : domain

com : commercial

------http/https(req)--------------------🡪

Client Server

🡨------http/https(res)------------------

Html/html5

Css/Css3

JavaScript (JS)

Validation using

JavaScript

HTML5 Features

HTML : hypertext markup Language which help to create web pages.

Web pages help to display the contents in different format like bold, italic, clips etc

HTML is use to create static as well as dynamic web page.

HTML provided lot of pre-defined tags (elements)

Syntax

<tagName> opening tag

</tagName> closing tag

<tagName/> self closing tag

Html tags

1. Html
2. Head : meta – data (data about data)
3. Body : actual contents to display on web page.
4. P (paragraph tag)
5. Br (break tag) : it doesn’t contains closing tag
6. Heading tag (6 types start from h1(largest) to h6(smallest))
7. b (bold)
8. i (italics)
9. u (underline)

Attribute : Attribute is known as properties of a tags.

Syntax

<tagName name1=”value1” name2=’value2’ name3=value3></tagName>

P and heading tag attribute

<p align=’left/right/center/justify’></p>

<h1 align=’left/right/center/justify’></h1>

Font tag : This tag is use change color, face(style) and size.

Image tag :

<img src=”imageName.jpeg/gif/” />

Hyperlink tag : This tag is use to connect external web page as well as point

to specific para(contents) within a same web page.

2 types

1. External hyperlink
2. Internal hyperlink (bookmark)

External hyperlink

<a href=”pathOfPage”>Text</a>

a – anchor

href – hyper reference

Add the images

**List tags:**

It is use to display the items

1. UnOrder List
2. Order List
3. Definition list

**Table Tag :**

Employee Details

|  |  |  |
| --- | --- | --- |
| **Id** | **Name** | **Salary** |
| 100 | Ravi | 12000 |
| 101 | Seeta | 14000 |
| 102 | Meeta | 16000 |

Form Tag :

Syntax

<input type=”text/password/radio/checkbox/button/submit/reset/file” />

Query Param :

url?key=value&key=value&key=value

Get : By default every form method.

Data will send through url using query param concept.

We can send maximum 256 character data.

Performance fast.

Post : we have write method=”post” in form tag

Data send through body part of http protocol.

Data secure and we can send huge data.

Performance slow compare to get methods.

Get/Post/Put/Delete : But html /html5 doesn’t support put and delete method.

Get : Get the Resources ie employee, customer, product

Post : Create the Resource ie employee, customer, product

Put : Update the Resources ie employee, customer, product

Delete : Delete the Resources ie employee, customer, product

**Day 4 :**

Html 4

.xhtml (document type definition )

<!doctype public url=”…………………………..dtd”>

Html 5

<!DOCTYPE html>

VSCode (Visual Studio) : UI training (angular / react )

.net

Web Page

3 parts

1. Contents : HTML/HTML5 (Basic Action using submit and reset)
2. Presentation : Look and Feel : CSS/CSS3
3. Action : JavaScript

CSS : Cascading Style sheet :

With the help of CSS we can apply good look and feel (formatting style for the contents).

We can achieve **separation of concern** : Contents separate and formatting style.

CSS 3 types

1. Inline CSS
2. Internal CSS or embedded CSS
3. External CSS

Inline CSS

Syntax

<tagName style=”property:value;property:value;”></tagName>

Internal CSS or embedded CSS

Syntax

In head

<style>

Selector {property:value;property:value;}

</style>

Types of Selectors

1. \* universal selector

\*{color:green;}

1. Specific selector

tagName {property:value}

1. Multi selector

tagName, tagName {property:value}

1. Class selector (Local class selector)

tagName.className {property:value}

1. Global class selector

.className {property:value}

1. Id selector

#idName{property:value}

Class selector Vs Id selector

Class : groups of the tag

Id : unique for that tag

DOM : Document Object Model

**JavaScript:**

JavaScript was object based interpreter scripting language. Which help to do client side validation as well as to create dynamic effect for the web page.

JavaScript using ES5 Features

ECMAScript : It is a concept. And JavaScript is a implementation of ES5.

Object Based Vs Object Oriented

Class, object, Encapsulation, Inheritance, Polymorphism and abstraction.

JavaScript is object oriented interpreter scripting language from ES6 onwards.

All browser doesn’t support ES6 all features.

Interpreter

Compiler

Both are translator : converting one format to another format.

Compiler convert all code at time where interpreter convert line by line.

Syntax (script tags)

<script type=”text/JavaScript”> opening tag

</script> closing tag

This tag must be in between head or body or within any html tags.

Variable : variable is a name which hold some value.

Value will change during the execution of program.

JavaScript is loosely data types.

int a=10; //C or C++ or Java

To declare the variable in JS we have keyword as **var**

Syntax

var variableName;

DataTypes

number

string

boolean

object reference

undefined

etc

primitive data types : it is use to store only value.

Syntax to create the memory of pre-defined or user-defined objects

var referenceName = new ObjectName();

var obj = new Date()

var obj = new String();

var obj = new Number();

Operators:

Arithmetic Operator :

+, -, \*, /, %(remainder)

Relation Operator

>, >=, <, <= , ==, !=

Logical Operator

&& || !

Assignment Operator

=

Increment and decrement operator

++ , --

++ increment the value by 1

--decrement the value by 1

Pre-increment and post increment

Pre-decrement and post decrement

++n; increment and store or assign

n++; assign or store and then increment

Ternary operator

Condition ? true : false;

===

if statement

1. Simple if

if(condition){

true block

}

1. If else

if(conditions) {

true block

}else {

false block

}

1. Nested if

if(condition) {

if(conditions) {

true block

}else {

False block

}

}else {

if(conditions) {

}

}

1. If ladder or if else if

if(conditions) {

}else if(conditions) {

}else if(conditions) {

}else {

}

switch statement

syntax

switch(variableName) {

case value:block1;

break;

case value:block2;

break;

case value:block3;

break;

default: wrongblock;

break;

}

switch, case, break and default are keywords.

Looping : it is use to execute the statement again and again till the conditions becomes false.

While loop

Do while loop

For loop

**functions :**

set of instruction to perform a specific task.

1. Pre-defined function
2. alert(“Msg”); This function is use to display alert or popup message on web page.
3. prompt(); This function is use to receive the value through keywords.
4. eval() : It is use to convert string to number
5. confirm : This popup contains two button ok and cancel

If user click ok then it return true else return false.

1. User-defined function

Do

1: Add , 2 :Sub : alert

: prompt

Switch statement

1 : Addition of two number

Alert to receive the value of a and b

eval : convert string to number

2: Subtraction of two number

Alert to receive the value of a and b

Confirm (to do the another operation)

While

**HTML5 : Few forms**

**CSS**

**JavaScript**

**Day 4 :**

user-defined function

syntax

function functionName(parameterList) {

}

1. function no parameter and no return type

2. function passing parameter and no return type

3. function no passing parameter but return type

4. function passing parameter and return type

ES6 React JS

Events : Events is interaction between user and html components.

It is delegation model.

Types of Events

In JavaScript all events start with on\*

like

onClick

onDblClick

onMouseOver

onMouseOut

onKeyUp

onKeyDown

onSubmit

onBlur

onFocus

onChange

etc

Event is provide bridge between html and JavaScript

DOM : document object model

DOM is a api(application programming interface)

which provide set of method and properties which help to read, write and

update html contents dynamically.

JavaScript objects :

In JavaScript object are divided into two types

pre-defined objects.

object : any real world entity.

JavaScript internally follow object hierarchy.

object ---> property (state) ---> have - varialbes

behaviour --------->do/does - functions

object

property (variables)

behaviour (functions)

object

property

behaviour

object

BOM : Browser Object Model

DOM : Document Object Model

user-defined objects.

Interval Functions

setInterval()

setTimeout()

clearInterval()

Day 5 :

**Box Model :**

div{

    border-style: groove;

    border-width: 1;

    border-color: red;

    width: 200px;

    height: 300px;

}

div{

    border: 2px solid red;

    width: 200px;

    height: 300px;

}

Position : Relative ; we can use top, left, right and bottom properties for that tag

Position : Absolute : We can’t top, left, right, bottom this property

But user can decided where you want to display that tag base upon the event.

Position : Fixed

**Database**

Copy OJDBC jar from here: H:\common\Trainee\_2020

<< OLE Object: Picture (Device Independent Bitmap) >>

The userids are train1-20  the password for **train1**is ‘**pitchers1**’  for

train2 the password is ‘pitchers2’, etc.

   The tnsnames entry you’ll need is below.

INDIATRN =

  (DESCRIPTION =

    (ADDRESS = (PROTOCOL = TCP)(HOST = [indiaoracletraining-dev-oracle-rds.c2zuiscfni9y.us-east-1.rds.amazonaws.com](http://indiaoracletraining-dev-oracle-rds.c2zuiscfni9y.us-east-1.rds.amazonaws.com/))(PORT = 1521))

    (CONNECT\_DATA = (SID = INDIATRN)))

Username: train1

**Database : (Oracle ) -- > SQL / PLSQL Using Oracle 5 days**

**SQL -- > 3 days**

**PlSQL 🡪 2 SQL**

**Data : Raw facts**

**Information : Processed Data or Meaningful data.**

**Database : Storing the data and information in proper or common (tables) format.**

**Input : keyboard, initialization , file, database or through network**

**Process : business rules.**

**Output : console, storing in file, database, through send to network.**

1. **File**
2. **Database**

**File base system**

1. **We can store data permanently.**
2. **We can store huge data.**
3. **We can apply security for data (read, write or execute)**

**Limitation of file base system**

1. **Data redundancy (means storing same data again and again) (Duplicate records)**
2. **Data inconsistency (format of the file).(txt, doc, xsl, pdf, etc)**

**.txt (delimiter for each records as next line or / or , or tab space)**

**.txt(delimiter for each data**

**Emp.txt**

**Id,name,salary**

**IdNameSalary**

1. **Security (Read or Write mode (Execute shell scripting or batch file))**

**Database : Tables Format (Table format)**

**DBMS : Database Management System : It is a software which help to store the Data in the table format like columns and rows.**

**Excel Sheet. DBMS**

**VB (Visual Basic ) with Excel**

**Database Model :**

**HDBMS : Hierarchal Model**

**NDBMS :**

**RDBMS : Relation DBMS (Database management system)**

**1970**

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

**The Database which support all 12 rules which provided by EF Codd’s rules**

**RDBMS : Relational Database Management System**

**RDBMS Databases**

**Database Name Company**

**MySQL Sun MicroSystem (Oracle)**

**Oracle Oracle**

**SQL Server Microsoft**

**2019, 2020**

**Db2 IBM**

**SQL : Structure Query Language**

**SQL Developer tools**

**Commands (CLI )Command line interface.**

**80% to 90%**

**SQL : Five types (Sub types )**

1. **DQL(Data Query Language) or DRL(Data Retrieval Language) (View Records)**
   1. **Select Clause** 
      1. **Select all records**
      2. **Select specific records**
      3. **Select with where clause**
      4. **Select with group by clause**
      5. **Select with having clause**
      6. **Select with order by**
      7. **Select with join**
      8. **Select with sub query**
      9. **Select with union operator**
2. **DDL (Data Definition Language) (table structure )**
   1. **Create**
   2. **Drop**
   3. **Alter**
   4. **Rename**
   5. **Truncate**
3. **DML (Data Manipulation Language) (work on data )**
   1. **Insert**
   2. **Delete**
   3. **Update**
4. **TCL (Transactional Control Language)**
   1. **Commit**
   2. **Rollback**
   3. **Savepoint**
5. **DCL (Data Control Language)** 
   1. **Create the user**
   2. **Grant the permission to the user**
   3. **Revoke the permission from the user**

**View**

**Index**

**Sequence**

**SQL :**

**PlSQL :**

**MySQL/ SQL Server / Db2 --🡪**

**username : root**

**password : \*\*\*\*\*\***

**Oracle :**

**Username : scott**

**Password : tiger**

**Sql server**

**Username : sys**

**Password : \*\*\*\***

**NoSQL**

**MongoDB**

**MySQL/ SQL Server / Db2 --🡪**

**show databases; This command is use do display the database.**

**use databaseName; This command is use to move the specific database.**

**Database : database is a collection of tables or views etc.**

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

**create database databaseName; This command is use to create the database.**

**Above four command are invalid in oracle database.**

**username itself is a database in oracle database.**

**system**

**scott**

**admin**

**oracle**

**hr**

**select \* from dual;**

**select \* from tab; : tab is a pre-defined table provided by oracle databases which help to display all pre-defined as well as user-defined tables table in that account.**

**Oracle eXpress Edition : small database. It support basic feature of oracle database.**

**Oracle enterprise edition : it support all features of oracles.**

**ORDBMS**

**Day 7 : 27-10-20**

**DBMS : Database Management system**

**RDBMS : Relational Database Management System**

**DBMS 🡪 Table, Columns , Rows**

**RDBMS -🡪 Relation, Attribute , Records (Tuple)**

**Trainer\_Student**

**TId TName Tech Sid SName Age**

**1 Raj Java 101 Seeta 21**

**1 Raj Java 102 Reeta 22**

**1 Raj Java 103 Meeta 23**

**2 Ravi C/C++ 104 Keeta 24**

**2 Ravi C/C++ 104 Teeta 25**

**Trainer**

**PK(Primary Key): doesn’t allow duplicate as well as null value.**

**TId TName Tech**

**1 Raj Java**

**2 Raj Java**

**3 Ravi C/C++**

**Students**

**PK FK(Foreign Key)**

**Sid SName Age TS\_Id**

**100 Seeta 21 1**

**101 Meeta 22 1**

**102 Keeta 23 2**

**103 Teeta 24 3**

**104 Leeta 26 null**

**FK is use to refer the primary of same table or different table. FK allow only those records(values) which present in primary key. It allow null value.**

**SQL : Structure Query Language**

1. **DRL or DQL (Data Query or Retrieval Language)**

**MySQL/SQL Server / Db2**

**Create database mydb;**

**Use mydb;**

**Show tables;**

**Scott : is database**

**Select \* from tab;**

**Select \* from tableName;**

**This command is use to display all records from that table**

**Table structure**

**desc tableName;**

**Filter the columns**

**Select columnName, columnName, columnName from tableName;**

**Columns alias**

**Select columnName as columnAlias, columnName columnAlias from tableName**

**select employee\_id as id,first\_name||' '||last\_name Full\_Name,salary,salary+5000 as Gross\_Salary from employees;**

**Employee\_Id,Full\_Name, BasicSalary, HRA, DA,PF,Gross\_Salary**

**Salary = 10%**

**DA = 5%**

**PF = 7%**

**GrossSalary = Salary + HRA + DA – PF**

**Filter records using Where clause**

1. **Relational Operator**

**>, >=, <, <=, = (equal), != or <>**

**Syntax**

**Select \* from tableName where columName RO value;**

**Select columName, columnName from tableName where columName RO value**

**Select \* from employees where salary > 15000;**

**Select employee\_id,first\_name from employees where salary > 15000;**

**Select employee\_id,salary from employees where first\_name =’Steven’;**

**select first\_name,job\_id,department\_id from employees where department\_id <> 90;**

**MySQL default date format is yyyy-mm-dd**

**Oracle default date format is dd-mon-yy**

**select first\_name,salary from employees where hire\_date ='14-Jun-97';**

**select first\_name,salary from employees where hire\_date > '31-Mar-95';**

1. **Between operator (Numerical and Date value)**

**Filter the range of records**

**Select \* from tableName where columnName between min and max**

**select first\_name from employees where employee\_id between 100 and 150;**

**select first\_name,salary from employees where salary between 10000 and 17000;**

1. **In operator (more then on specific)**

**Select \* from tableName where columName in(v1,v2,v3)**

**Value number then write directly if value are alphanumeric or date then write in single quote.**

**select first\_name from employees where employee\_id in(100,110,105,150,1000);**

**select first\_name,job\_id from employees where job\_id in('ST\_MAN','IT\_PROG');**

1. **Like operator (Specific only one records)**

**Select \* from tableName where columnName like value;**

**select employee\_id,salary from employees where first\_name like 'Steven';**

**select employee\_id,salary from employees where first\_name = 'Steven';**

**select employee\_id,first\_name,salary from employees where first\_name like 'S%';**

**Start with S Character**

**select employee\_id,first\_name,salary from employees where first\_name like '%n';**

**end with s character**

**select employee\_id,first\_name,salary from employees where first\_name like '%r%';**

**contains r character**

**% zero or one or many**

**select employee\_id,first\_name,salary from employees where first\_name like 'S\_e%';**

**Start with S, then 2nd character may be anything, 3rd may be e then one or more any character.**

**\_ any one single character**

1. **is null :**

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

1. **logical and, or, not**
2. **select \* from tableName where columnName op value and columnName op value (both the conditions must be true)**
3. **select \* from tableName where columnName op value or columnName op value (any condition must be true then we will get the records)**

**select first\_name,salary from employees where salary > 10000 and department\_id=90;**

**select first\_name,salary from employees where salary > 10000 or department\_id=90;**

1. **not**
2. **not between min and max**
3. **not in (v1,v2,v3)**
4. **not like value**
5. **is not null**

**Order by clause ( it is use to display the records in ascending or descending order)**

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

**select first\_name, salary from employees order by salary;**

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

**Multi sort (Order by clause )**

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

**Functions**

**Function is use to write the set of instruction to perform a specific task.**

**Oracle**

**Dual**

**Select \* from dual;**

**2 types**

**All function takes one or more than one parameter and return the value. ( it must return the values).**

1. **Pre-defined function**

**Two types**

1. **Single row function : The output or return type apply for every records individually depends upon the function.**
2. **Character function**

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

**select upper('raj') from dual;**

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

**select concat(first\_name,' ',last\_name) from employees; (it takes only two parameter)**

**select concat(first\_name,' ',last\_name) from employees;(In MySQL database it takes n parameter)**

**nested function**

**select concat(first\_name,concat(' ',last\_name)) from employees;**

**(nested function, first inner function execute, inner function output is input for outer functions).**

**or**

**select first\_name|| ‘ ‘ ||last\_name from employees**

**( it doesn’t support in MySQL)**

**Select substr(content,start,numberOfChar) from dual**

**select substr('rajdeepkumar',4) from dual;**

**select substr('rajdeepkumar',4,4) from dual;**

**select length('Raj Deep') from dual;**

**select trim(‘ Raj Deep ‘) from dual; help to remove the space before and after**

**select length(trim(' Raj Deep ')) from dual;**

**Number function**

**round()**

**trunc()**

**Date function :**

**Sysdate**

**Months\_between(newDate,oldDate)**

**select months\_between(sysdate,'01-Jan-20') from dual;**

**add\_months(sysdate,3)**

**select add\_months(sysdate,4) from dual;**

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

**select last\_day(sysdate) from dual;**

1. **Display all employees first\_name and numberOfYearOfExp from employees tables.**

**Hire\_date (**

**20 to 30 year**

**Remove decimal number but trunc**

1. **YearOfExp more than 25 years**

1. **Multi row function or aggregate function : The output or return type apply for group of records depending upon the group by clause. (if we doesn’t use group by clause we will get only one output for multi row function if the table contains one or more than one records).**

**Day 8 : 10/28/2020**

**Conversion functions**

**Oracle Database Data types**

**number(10) -🡪 maximum number of digits is 10**

**number(10,2)-🡪 total number is 10, before decimal 8 and after decimal 2**

**int ---🡪**

**float -🡪**

**char 🡪 it is use to store single character**

**varchar(10)-🡪 number of character**

**varchar2(10) --🡪number of character**

**date -🡪 it is use to store date and time information**

**timestamp -🡪 to store time**

**etc**

**to\_char : converting date to varchar/varchar2**

**to\_char(dateDetails,’Date Format’)**

**select to\_char(sysdate,'YYYY') from dual;**

**select to\_char(sysdate,'DD-MM-YYYY') from dual;**

**select to\_char(sysdate,'DD-MON-YYYY') from dual;**

**select to\_char(sysdate,'DD-MON-YYYY HH:MI:SS') from dual;**

**select to\_char(sysdate,'DAY DD MONTH YEAR') from dual;**

**to\_char : converting number to char**

**to\_date :**

**to\_number**

**nvl : null value**

**nvl(null,1) : If value is null replace by second parameter**

**nvl(‘Raj’,1) : if value not null contains same value.**

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

**nvl2(first\_parameter,second\_parameter,third\_parameter)**

**if first\_parameter is a not null then we will get second parameter**

**if first\_parameter is equal to null then we will get third parameter.**

**Case (This function is equal to switch statement in programming language)**

**Syntax**

**Case expr when comparision\_expression then return\_value**

**when comparision\_expression then return\_value**

**when comparision\_expression then return\_value**

**else**

**return\_value end**

**Job\_ID :**

**select first\_name,salary,Job\_id,case job\_id when 'AC\_MGR' then salary\*0.10 when 'SH\_CLERK' then salary\*0.05 else salary\*0.01 end as Bonus from employees;**

**Decode : decode is a short cut of case function**

**Select first\_name,salary,Job\_id,decode(job\_id,’AC\_MGR’,salary\*0.10,’SH\_CLERK’,salary\*0.05,salary\*0.01) as bonus from employees**

**Rather display 0 u have to display the message as no commission. (hint you have to take conversion function help).**

**nvl2**

**Multi – row functions or aggregate functions**

1. **sum()**
2. **max()**
3. **min()**
4. **avg()**
5. **count()**

**select sum(salary) from employees;**

**select sum(salary) as Total\_Salary from employees;**

**select max(salary) from employees;**

**select min(salary) from employees;**

**select sum(salary) as total,min(salary) as min\_sal,max(salary) as max\_sal, avg(salary) as avg\_salary from employees;**

**select count(commission\_pct) from employees;**

**select count(manager\_id) from employees;**

**select count(employee\_id) from employees;**

**select count(\*) from employees;**

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

**Sub group**

**Group by Clause**

**Employees table -🡪 FK Columns (Job\_Id, Department\_id, Manager\_Id)**

**Employee\_Id 🡪 PK**

**Group by Department Id**

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

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

**aggregate function with where clause and group by**

**select department\_id,sum(salary) from employees where department\_id is not null group by department\_id;**

**Having clause**

**Having clause must be after group by where clause must be before group by**

**Having clause is use to check the condition on group of records where clause is use to check the condition on single records**

**Where clause (Single row function)**

**select department\_id,sum(salary) from employees where first\_name like initcap('steven') group by department\_id;**

**Having clause (Multi row function)**

**select department\_id,sum(salary) from employees where department\_id is not null group by department\_id having sum(salary) > 50000;**

**select department\_id,sum(salary) from employees where department\_id is not null group by department\_id having sum(salary) > 50000 order by department\_id desc;**

**Single row and Multi row assignment Question**

Single row function12:24

1. Display first name, salary, and round the salary to thousands.12:24

2. Display the first word in job title.12:24

3. Display the length of first name for employees where last name contain character ‘b’ after 3rd position.12:24

4. Display the number of days between system date and 1st January 1995.12:25

5. Display how many employees joined in each month of the current year.12:25

Multi row functions12:26

1. Display average salary of employees in each department who have commission percentage.12:26

2. Display job ID for jobs with average salary more than 10000.12:26

3. Display years in which more than 10 employees joined.12:27

4. Display departments where any manager is managing more than 5 employees.12:27

5. Display department id and number of employees in the department.

**DDL and DML**

**Data Definition Language: Working Table Structure (Create, Drop and Modify).**

**Create**

**Drop**

**Truncate**

**Alter**

**Syntax to create the Table**

**Create table tableName(columnName dataTyp1, columnName dataType2…..columnName dataTypen);**

**Table 🡪EmpDetails**

**Id Name Salary**

**Number(10), varchar2(10) number(10,2)**

**create table empdetails(id number(10), name varchar2(10), salary number(10,2));**

**desc empdetails**

**DML: : Data Manipulation Language (Insert, Delete and Update)**

**Insert query**

**Insert into tableName(c1,c2,c3) values(v1,v2,v3);**

**insert into empdetails(id,name,salary) values(1,'Raj',12000);**

**insert into empdetails(id,name) values(2,'Ravi');**

**insert into empdetails values(5,'Ajay',null);**

**insert into empdetails(id) values(3);**

**insert into empdetails values(6,null,null);**

**insert into empdetails values(4,'Ramesh',14000);**

**insert into empdetails(name,salary,id) values(‘Dinesh’,14000,7);**

**Update Query**

**Syntax**

**Update tableName set columnName = value;**

**Update empdetails set salary = 25000;**

**Update query with where clause**

**Update tableName set columnName = value where columnName = value;**

**update empdetails set salary = 15000 where id=2;**

**update empdetails set name = 'Balaji' where id=3;**

**update empdetails set salary = 18000 where id in(3,5,6);**

**update empdetails set name = 'Ramesh' where id =7;**

**update one value base upon more than one condition satisfies**

**update one value base upon any one condition satisfies**

**update empdetails set salary=20000 where name like '%e%' and id=4;**

**update empdetails set salary=22000 where name like '%e%' or id=2;**

**update more than one value depending upon one conditions.**

**update empdetails set name='Raj Deep',salary=salary+2000 where id=1;**

**Delete Query**

**Syntax**

**Delete from tableName;**

**All records delete from table**

**Delete with where clause**

**Delete from tableName where columnName = value;**

**delete from empdetails where id=7;**

**delete from empdetails where salary > 20000;**

**delete from empdetails where name like 'Raj';**

**Drop table**

**Delete from tableName (without where clause)**

**All records are deleted but table structure present in database.**

**Drop table tableName; (All records as well as table structure remove from database).**

**drop table empdetails;**

**Truncate table tableName; All records are deleted but table structure available in database.**

|  |  |  |
| --- | --- | --- |
| **Truncate** | **Delete** | **Drop** |
| **DDL** | **DML** | **DDL** |
| **All records deleted** | **All records deleted** | **All records deleted** |
| **Table structure available** | **Table structure available** | **Table structure removed** |
| **We can’t use where clause** | **We can use where clause** | **We can’t use where clause** |
| **We can’t use rollback (undo)** | **We can use rollback (Undo)** | **--** |

**Alter command**

**Add column**

**Modify column**

**Drop column**

**alter table empdetails add desg varchar2(2);**

**alter table empdetails modify desg varchar2(10);**

**alter table empdetails drop column salary;**

**select \* from empdetails;**

**Rename tableName**

**alter table empdetails rename to employeedetails;**

**alter table employeedetails rename column desg to designation;**

**Join :**

**De-Normalization**

**Join is use to retrieve more than one columns from more than one table with or without conditions.**

**Create table table1(srno int, name varchar2(10), salary float)**

**Insert into table1 values(1,’Raj’,12000);**

**Insert into table1 values(2,’Reeta’,14000);**

**Create table table2(accno int, name varchar2(10), amount float)**

**Insert into table2 values(111,’Raj’,500);**

**Insert into table2 values(222,’Seeta’,1000);**

**Cartesian Product : m \* n**

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

**2 \* 2 = 4**

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

**but with common columns**

**Table alias**

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

**select \* from table1, table2;**

**Equi – Join**

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

**Outer Join**

**Left outer join**

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

**Right outer join**

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

**Inner Join**

**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**

**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**

**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 Outer join**

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

**Employee**

**10**

**5 working on project**

**5 on bench**

**Project**

**10**

**4 active**

**6 not started yet**

**Equi – Join Or Inner Join**

**N table it require n-1 min conditions**

**2 table it require 1 condition**

**3 table it require 2 condition (1st – 2nd table and 1-3rd table or 2 – 3rd table)**

**Using where clause – Equi – Join**

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

**Using on clause – Inner join**

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

**Self – Join : Joining the same table itself is known as self join.**

**When doing join it not mandatory column name must be same but the value as well as their data types must be same.**

**TCL :**

1. **User-defined function (Pl SQL )(Stored Procedure and functions)**

**Day 9 : 10/29/20**

**Sub Query**

**Query within another query is known as sub query**

**Syntax**

**Outer Query (Inner Query)**

**First Inner Query will execute and Inner query output is input for Outer Query.**

**Outer Query (Inner Query (Inner Inner Query ))**

**Inner Query must be return only one column (means not more than one or \* ).**

**Inner Query**

**Some query retrieve only one record or result (If inner query using Aggregate function (Without Group By Clause) or where clause with primary key).**

**Some Query retrieve more than one records**

**Sub Query mainly divided into two types**

1. **Single row sub query**
2. **Multi row sub query**

**Syntax of Single row sub query**

**Outer query RO/Like (Inner Query)**

**Syntax of Multi row sub query**

**Outer query in / any ro / all ro (Inner Query)**

**Find person name or details whose salary is > avg salary of all employees.**

**select avg(salary) from employees;**

**select first\_name from employees where salary > 6461;**

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

**select first\_name from employees where salary > (select max(salary) from employees where department\_id=90);**

**Multi – Row functions**

**In Operator**

**Find the person details whose job\_id is equal to those employee working in a department 90.**

**Find the manager\_name(first\_name) who manages more than 4 employees**

**select count(\*) from employees group by manager\_id;**

**select first\_name from employees where employee\_id in (select manager\_id from employees group by manager\_id having count(\*) > 4);**

**Find first\_name whose salary is greater than any of the employee working department 60**

**Inner Query**

**select salary from employees where department\_id=60**

**min\_salary = 4200**

**max\_salary = 9000**

**any**

**>any ( > min\_salary of inner query) ie > 4200**

**select first\_name from employees where salary >any (select salary from employees where department\_id=60);**

**select first\_name,salary from employees where salary >any (select salary from employees where department\_id=60);**

**select first\_name,salary from employees where salary > (select min(salary) from employees where department\_id=60) ;**

**Find first\_name whose salary is greater than all of the employee working department 60**

**all**

**>all ( > max\_salary of inner query) ie > 9000**

**select first\_name,salary from employees where salary >all (select salary from employees where department\_id=60);**

**select first\_name,salary from employees where salary > (select max(salary) from employees where department\_id=60);**

**exists**

**not exists**

**outer Query where exists (inner query)**

**If Inner query return true then outer query execute**

**If inner query return false then outer query doesn’t execute.**

**select first\_name from employees where exists (select \* from employees where department\_id=100);**

**Keys :**

**EmpId,EmpName,Age,Salary,PhNumber,EmailId, jobId,JobDesc,accnoNumber,typeOfAccount,custId,customername,addId,city,state,pincodde,projectId,projectName,startDate,endDate etc**

**Etc**

**Super key : all set combination are known as Super set**

**EmpId**

**AccNumber**

**custId**

**AddId**

**porjectId**

**phNumber**

**EmailId**

**JobId**

**EmpId,EmpName**

**JobId,JobDesc**

**projectId,projectName**

**EmpId,phNumber**

**EmpId,phnumber,FirstName**

**EmpId,Phnmber,FirstName,jobId**

**Candidate keys :**

**EmpId,Phnmber,FirstName,jobId : Super Key**

**EmpId,Phnmber,FirstName :Super key**

**EmpId,Phnmber : Super key**

**EmpId :Super key**

**PhNumber :Super**

**FirstName : not a super key**

**JobId :Super key**

**EmpId,PhNumber, JobId are Candidate keys.**

**Primary key**

**Primary key :**

**EmpId PK**

**AccNumber CK**

**custId CK**

**AddId CK**

**porjectId CK**

**phNumber CK**

**EmailId CK**

**JobId CK**

**In single table only one column as a primary key**

**It doesn’t allow duplicate as well as null value.**

**Unique Key :**

**EmpId PK**

**AccNumber UniqueKey**

**custId UniqueKey**

**AddId UniqueKey**

**porjectId UniqueKey**

**phNumber UniqueKey**

**EmailId UniqueKey**

**JobId UniqueKey**

**In Single table we can create more than one column as unique key**

**It doesn’t allow duplicate but it can allow null value.**

**Foreign key :**

**It is use to refer the primary key of same table or different table**

**It allow duplicate records which present in primary key column. It allow null value.**

**Composite Key: If we create primary key(combination of more than one column). Both column independently allow duplicate but combination must be unique.**

**Alternate key : They are type of candidate or unique key**

**Secondary key :**

**Surrogate key : it help to generate auto increment number.**

**Constraints :Constraints is use to restrict the user to insert invalid data like duplicate records, -ve value. Etc**

1. **Not null**
2. **Default**
3. **Check**
4. **Unique**
5. **Primary key**
6. **Foreign key**
7. **Composite primary key**

**EmployeeDetails**

**PK not null null >21 male|female, 250000 and 50000, Default GWG**

**EmpId ,FName,LName,Age ,Gender,Salary,PhNumber,DOB, Company**

**create table employeedetails(**

**empid int primary key,**

**fname varchar2(10) not null,**

**lname varchar2(10),**

**age int check(age>21),**

**gender varchar2(10) check(gender in('male','female')),**

**salary number(10) check(salary between 25000 and 50000),**

**phnumber number(10) unique,**

**dob date not null,**

**company varchar(10) default 'GWG');**

**insert into employeedetails values(100,'Raj','Deep',24,'male',34000,9900,'30-Jan-85','Wipro');**

**insert into employeedetails values(101,'Raj','Deep',24,'male',34000,9901,'30-Jan-85','Wipro');**

**insert into employeedetails values(102,'Raj','Deep',24,'male',50000,9902,'30-Jan-85','Wipro');**

**insert into employeedetails values(103,'Seeta',null,22,'female',45000,9903,'10-Feb-90','IBM');**

**insert into employeedetails values(104,'Reeta',null,22,'female',45000,9904,'10-Feb-92','TCS');**

**insert into employeedetails(empid,fname,lname,age,gender,salary,phnumber,dob) values(105,'Leeta',null,24,'female',30000,9906,'10-Mar-92');**

**Relationship :**

**ER-Model : Entity Relationship Diagram**

**4 types**

1. **One – To – Many (PK-🡪FK) : Trainer – Student**
2. **Many – To – One (FK🡨-PK) : Employee – Department /Project**
3. **One – To – One (PK-🡪 or 🡨 PF) Shared Primary key Person -🡪PanCard/Passport**
4. **Many – To – Many (PK----🡪FK and FK🡨--PK)**

**One – to – Many bi-directional**

**Students <---🡪SkillSet**

**Many – to – Many**

**Trainer**

**PK**

**TId TName Tech (Primary table)**

**1 Raj Java**

**2 Raju .net**

**3 Ravi C**

**PK FK**

**Student**

**Sid SName Age TSId (Secondary Table )**

**101 Seeta 21 1**

**102 Meeta 22 1**

**103 Keeta 23 2**

**104 Leeta 24 2**

**105 Veeta 25 null**

**Trainer**

**create table trainer(tid int primary key,tname varchar2(10) not null, tech varchar(10) not null);**

* **Column level constraints**

**create table trainer(tid int,tname varchar2(10) not null, tech varchar(10) not null,**

**constraints t\_pk primary key(tid));**

* **Table level constraints**

**insert into trainer values(1,'Raj','Java');**

**create table student(stid int primary key,sname varchar2(10) not null, age int, tsid int references trainer(tid));**

* **Column level constraints**

**create table student(sid int,sname varchar2(10) not null, age int, tsid int,**

**constraints s\_pk primary key(sid),**

**constraints ts\_fk foreign key(tsid) references trainer(tid));**

* **Table level constraints**

**insert into student values(100,'Seeta',21,1);**

**We can’t drop primary table if table link with foreign key of another secondary table.**

**Many – to – Many**

**Students**

**Sid PK**

**SName**

**create table students(sid int, sname varchar(10) not null, constraints s1\_pk primary key(sid));**

1. **Raj**
2. **­­Ravi**

**SkillSet**

**SSId PK**

**SSName**

**create table skillset(ssid int, ssname varchar(10) not null, constraints ss1\_pk primary key(ssid));**

**100 C**

**101 Java**

**102 .net**

**Student\_SkillSet ­­**

**SSS\_Id pk**

**Sid FK for Student**

**SSId FK for SkillSet**

**111 1 100**

**112 1 101**

**113 1 102**

**114 2 100**

**115 2 101**

**create table students\_skillset(sss\_id int,**

**sid int,**

**ssid int,   
constraints sss\_pk primary key(sss\_id),**

**constraints sid\_fk foreign key(sid) references students(sid),**

**constraints ssid\_fk foreign key(ssid) references skillset(ssid));**

**select \* from students\_skillset;**

**insert into students\_skillset values(115,2,101);**

**StudentName Technology**

**Use Join**

**Using where as well as on clause**

**One – To – One (Using Shared Primary key)**

**Person (Primary )**

**PK**

**PId PName**

**create table person(pid int, pname varchar(10), constraints pp\_pk primary key(pid));**

**PK**

**PanCard (Secondary)**

**PCId ValidDate**

**create table pancard(pcid int primary key references person(pid), validdateinfo date);**

**or**

**create table pancard(pcid int, validdateinfo date,constraints pc\_pk primary key(pcid), constraints pc\_fk foreign key(pcid) references person(pid));**

**DDL Assignments with Constraints**

**Table ---> Customer**

**custId, firstName,lastName,age,city, mobileNumber, dob**

**Add the Constraints**

**custId is Primary Key**

**firstName not null**

**age must be greater than 21**

**mobile must be unique**

**Table ----> Branch**

**branchId, branchName, city**

**Add the Constraints**

**branchId is Primary Key**

**branchName not null**

**city not null**

**Table -----> Account**

**accountNumber, openingBalance, typeOfAccount, status,BankId,CustId**

**Add the Constraints**

**accountNumber is primary key**

**openingBalance must be greater than 5000**

**typeOfAccount must be saving/current**

**BankId is foreign key refer to BranchId(Primary key) Branch table**

**CustId is foreign key refer to Customer(Primary key) Customer table**

**Table ----> Transaction**

**transactionId, transactionDate, MediumOfTransaction, TransactionAmount**

**Add the Constraints**

**transactionId is primary key**

**Table ----> Loan**

**LoanId, loanAmount, customerId and bankdId**

**Add the Constraints**

**loadId is primary key**

**loanAmount must be +ve**

**BankId is foreign key refer to BranchId(Primary key) Branch table**

**Using alter command add / drop constraints**

**Update and delete cascade**

**TCL : Transactional Control Language**

**Commit**

**Rollback**

**Savepoint**

**DML Operation**

**Insert / Delete and Update**

**Update account set amount = amoun-500 where account =123;**

**Update account set amount = amount+500 where account =567;**

**ACID Properties**

**Savepoint savepointName;**

**Example**

**Savepoiont s1;**

**Sequence : sequence is pre-defined object which help to create auto increment number in oracle database.**

**Create sequence sequenceName;**

**Create sequence pro\_seq**

**pro\_seq.nextval : It help to increment the number start from 1 and increment by 1**

**create sequence pro\_seq1 start with 100 increment by 2;**

**View : View is use to create the virtual table it always point to base table.**

**Create view viewName**

**as**

**Select clause**

**If we are created view on single with all columns through view we can do DML Operation.**

**create view prod\_view1 as select \* from product;**

**create view prod\_view2 as select \* from product where price > 100000;**

**create view prod\_view3 as select \* from product where price > 50000 with check option;**

**(We can insert only those records where price must be > 50000)**

**insert into prod\_view3 values(7,'Mobile',42000); (Error price must be > 50000)**

**insert into prod\_view3 values(7,'Mobile',52000);**

**create view prod\_view4 as select \* from product with read only;**

**WE can’t do any DML Operation through this view**

**insert into prod\_view4 values(8,'Pant',2500); (Error )**

**create the view**

**as**

**select query with join with group by with sub query**

**select \* from viewName;**

**create view myview**

**as**

**select t.tid,t.tname,s.sname,s.age from trainer t, student s where t.tid = s.tsid;**

**select \* from myview**

**DCL :**

**PL SQL**

**Day 10 : 11/2/20**

**PL SQL :** Programming or Procedure Language on SQL (Structure Query Language).

SQL :

DRL, DDL, DML (Insert/Delete/Update):

Limit of SQL.

In SQL we can execute only one query at time(The query can effect more than one records).

If We wan to execute more than one query with term and conditions( programming features).

PL SQL syntax

declare

variable declaration (optional)

begin

coding or coding

SQL and PL SQL

exception (optional)

exception handling block

end; (it must be end with semicolon)

Welcome program

begin

dbms\_output.put\_line('Welcome to Pl SQL');

end;

by default plsql console is disable

to enable pl sql console

set serveroutput on;

Variables : variable is a name which hold the value.

Scalar variable (The value can hold only one value).

Syntax to declare the variable

variableName datatype;

variable initialization

variableName datatype := value;

Different type of scalar variable

variablename datatype;

varaibleName datatype:=value

variableName dateTye not null:=value; // it must hold value

variableName constant datatype:=value; //here must be assign the value as well as we can’t change the value

PL SQL Select clause with into clause

Select columName1, columnNme2,….. columnNamen into variableName1, variableName2…. variableNamen from tableName where clause with primary key column

## **Variables anchors :** When pl sql engine or tools execute the select statement it check the size and datatypes of variable. If any mismatch between data type as well as size of data types. We will get the error.

Variables anchors makes Pl SQL flexible even if the table structure change in future. It automatically load the data type as well data type size.

Syntax

variableName tableName.columnName%type;

if statements

if else

if else if

Looping :

1. Simple loop
2. While loop
3. For loop

Scalar data types : This type of data types can hold only value at time.

Composite Data types : This type of data type can hold more than one value.

1. Collection : It is a type of composite data type which help to store more than one value of same type like array in C or Java.
2. Records : It is a type of composite data type which help to store more than one value of different types like Structure in C or Class in Java

Collection :

VArray : The VArray is one type of collection data types.

Syntax

type arrayVariableName is varray(size) of datatype;

variableName arrayVariableName;

variableName number(10);

Nested table : size increase dynamically.

Syntax

Type tableVariableName is table of datatype;

variableName tableVariableName;

**Records :** It is a type of composite data type which help to store more than one value of same or different types like Structure in C language.

Syntax

type recordTypeName is record(variableName datatype, variableName datetype…..varaibleName datype);

recordVariable recordTypeName;

**RowType : Row type is a type of attribute which help to create the records of that specific table with all column as well as type of that columns (with number character with precision).**

**Syntax**

Record\_VariableName tableName%rowtype;

The table columnName itself is behave as record member.

Record\_variable.record\_member (tableColumnName)

PL SQL Assignment

1. Write a program to interchange the salaries of employee 120 and 122.
2. Increase the salary of employee 115 based on the following conditions: If experience is more than 10 years, increase salary by 20% If experience is greater than 5 years, increase salary by 10% Otherwise 5%
3. Change commission percentage as follows for employee with ID = 150. If salary is more than 10000 then commission is 0.4%, if Salary is less than 10000 but experience is more than 10 years then 0.35%, if salary is less than 3000 then commission is 0.25%. In the remaining cases commission is 0.15%.
4. Find out the name of the employee and name of the department for the employee who is managing for employee 103.
5. Change salary of employee 130 to the salary of the employee with first name ‘Joe’. If Joe is not found then take average salary of all employees.

If more than one employee with first name ‘Joe’ is found then take the least salary of the employees with first name Joe.

**Day 11 : 11/3/20**

**Exception Handling :**

**Exception is a type of error which occurs in PLSQL block when PLSQL engine execute the PL SQL code (No SQL and SQL Statement).**

**Because of exception your application will get terminate abnormally.**

**Block**

**1**

**4 line**

**6 line**

**10 code**

**In PLSQL the exception are divided into three types.**

1. **Pre-defined exception (pre-defined name and pre-defined error code as well as message).**
2. **Pre-defined exception (pre-defined error but not pre-defined name).**
3. **User-defined exception (user-defined name and user-defined error code with some range provided by oracle).**

**Syntax of exception**

**declare**

**begin**

**1**

**To**

**20**

**exception**

**when exception\_name then**

**do task**

**when exception\_name then**

**do task**

**when exception\_name then**

**do task**

**when others then**

**other task**

**end;**

**-- First Exception example**

**set serveroutput on;**

**declare**

**a number(2):=10;**

**result number(2);**

**begin**

**result := a/0;**

**dbms\_output.put\_line('Result is '||result);**

**exception**

**when others then**

**dbms\_output.put\_line('Exception generated');**

**end;**

**set serveroutput on;**

**declare**

**a number(2):=10;**

**result number(2);**

**v\_name varchar2(10);**

**begin**

**result := a/1;**

**dbms\_output.put\_line('Result is '||result);**

**select first\_name into v\_name from employees;**

**--insert into product values(100,'Abc',10000,21);**

**--dbms\_output.put\_line('Record inserted');**

**exception**

**when zero\_divide then**

**dbms\_output.put\_line('Divided by zero');**

**dbms\_output.put\_line(SQLCODE);**

**dbms\_output.put\_line(SQLERRM);**

**when too\_many\_rows then**

**dbms\_output.put\_line('to many recods found');**

**--dbms\_output.put\_line(SQLCODE);**

**--dbms\_output.put\_line(SQLERRM);**

**when others then**

**dbms\_output.put\_line('to many recods found');**

**dbms\_output.put\_line(SQLCODE);**

**dbms\_output.put\_line(SQLERRM);**

**end;**

**user-defined exception name with pre-defined error code**

**foreign\_key\_exception**

**set serveroutput on;**

**declare**

**dup\_rec exception; -- created the variable of type exception**

**foreign\_key\_exception exception;**

**pragma exception\_init(dup\_rec,-1); -- register that exception using pre-defined procedure**

**pragma exception\_init(foreign\_key\_exception,-02291);**

**begin**

**--insert into trainer values(1,'Ravi','Java');**

**--dbms\_output.put\_line('REcord inserted');**

**insert into student values(1,'Seeta',21,2);**

**dbms\_output.put\_line('Record inserted successfully');**

**exception**

**when dup\_rec then**

**dbms\_output.put\_line('Primay key exception - record already present');**

**when foreign\_key\_exception then**

**dbms\_output.put\_line('foreign key exception - record must present in primary - trainer table');**

**when others then**

**dbms\_output.put\_line('Generic Exception');**

**dbms\_output.put\_line(SQLCODE);**

**end;**

**user-defined name as well as user-defined error code**

**Error code must be between the range of -20000 to -20999**

**Cursor : A Cursor is a temporary memory created by PL SQL engine when we execute SQL (DML,DDL,DRL) query in PL - SQL Block.**

**That temporary memory hold the details about that query which executed in PL SQL block**

**Types of cursor**

1. **Implicit cursor : DML Operation**

**All implicit cursor start with sql% followed by cursor name**

1. **Sql%found : Boolean**
2. **Sql%notfound : Boolean**
3. **Sql%rowcount : number**
4. **Sql%isopen : Boolean**
5. **Explicit cursor : DRL or DQL Operation**

**Explicit cursor can use with DRL or DQL which help to retrieve or fetch one record at a time.**

**steps to create explicit cursor**

1. **create cursor**
2. **open cursor**
3. **fetch record**
4. **close cursor**

**Cursor creation syntax**

**cursor cursorName is select clause**

**syntax to open the cursor**

**open cursorName;**

**fetch the records from cursor**

**fetch cursorName into recordsName**

**Pl SQL Block**

1. **Anonymous block (No name for this block so we can’t do reusability)**
2. **Names block : we can do reusability** 
   1. **Stored Procedure or Procedure**
   2. **Function**

**Syntax**

**Create procedure procedurename**

**as**

**begin**

**end procedureName;**

**Procedure Parameters**

**types of parameters**

1. **In parameter (by default) (read the value)**
2. **Out parameter (write)**
3. **In Out parameter (read and write)**

**Function is a type of named block which return mandatory one value using return keyword.**

**Syntax**

**Create or replace function functionName**

**Return datatype**

**As**

**Begin**

**return value;**

**End functionName;**

**Trigger : Trigger is a type of special stored procedure which get fire or automatically get call when we do DML Operation on specific table.**

**Trigger is a constructor in OOPs Language.**

**Syntax**

**Create or replace trigger triggerNam**

**Before/After**

**Insert/Delete/Update**

**On tableName**

**For each row**

**Begin**

**end triggerName;**

**On single table we can create 6 trigger**

**Trigger task always execute on background of application.**

**Emp Table**

**EmpId,EmpName,EmpSalary : Insert the records**

**Emp\_Track\_Info**

**EmpId,EmpName,dateandtime**

**Package : package is a collection of procedure, functions, cursor etc.**

**Package divided into two parts**

**Package header part**

**Package body part**

**P1 and p2 inside a package**

**mypack**

**p1, p2**

**Product Table**

**Pid PName price**

**Day 12 : 11/4/20**

**C Programming Language :**

**C is basic or structure programming language which help to develop generic as well as specific application.**

**C develop in 1972.**

**B**

**BCPL**

**Pascal**

**Cobol : IBM mainframe**

**Fortan 77**

**C topic**

**C compiler**

**C welcome program**

**C function (pre-defined main)**

**Data types**

**Array concept**

**If statement (type of if statements)**

**Switch statement**

**Looping while loop, do while loop for loop**

**User – defined function**

**Type of variables local and global**

**Pointer**

**Macro**

**Structure**

**Union**

**Etc**

**C Programming Language structure**

**Pre-processing instructions :**

**Global variable declaration (optional)**

**Pre-defined or user-defined function**

**Inside a main function**

**Variable declaration**

**Statements one or more**

**First welcome program in C Language**

**#include<stdio.h> //standard input and output h means header**

**Syntax**

**returnType functionName(parameterList) {**

**}**

**Return type : int, float, char, char[], void etc**

**#include<stdio.h>**

**void main() {**

**printf(“Welcome to C Program”);**

**}**

**vi demo.c**

**Data Types : Data type is a type of which tells what type of value it can hold.**

**JS :**

**Shell scripting :**

**var**

**Data types divided into 2 types**

1. **Basic data types** 
   1. **int (without decimal )**
   2. **float (with decimal point)**
   3. **char (any single character)**
   4. **double (with decimal point)**
2. **Derived data types** 
   1. **array**
   2. **pointer**
   3. **structure**
   4. **union**

**syntax for data types**

**datatype variableName;**

**#include<stdio.h>**

**#include<stdlib.h>**

**void main() {**

**int a;**

**system("clear");**

**printf("%d",a); // garbage value +ve, -ve or 0**

**}**

**gcc datatypeexample.c**

**./datatypeexample.exe**

**Write C program to declare int, float, char and double values and display it.**

**#include<stdio.h>**

**#include<stdlib.h>**

**void main() {**

**int a=10;**

**float b = 10.10;**

**system("clear");**

**printf("----------------------------------\n");**

**printf("The value of a is %d\n",a);**

**printf("The value of b is %f\n",b);**

**printf("---------------------------------\n");**

**}**

**Data Type size.**

**int –size 2 byte (generic way)**

**float –size 4 byte**

**char –size 1 type**

**double – size 8 byte**

**sizeof() : it is pre-defined function provided in C which help to check the size of variable base upon that type.**

**Taking the value through keyboards**

**scanf(): it is a pre-defined function which help to take the value through keyboard.**

**Syntax**

**scanf(“format specifiers ”,&variableName);**

**Sum of two number**

**#include<stdio.h>**

**#include<stdlib.h>**

**void main() {**

**int a,b,sum;**

**printf("----------------------------------\n");**

**printf("Enter the value of a\n");**

**scanf("%d",&a);**

**printf("Enter the value of b\n");**

**scanf("%d",&b);**

**sum = a+b;**

**printf("Sum of two number is = %d\n",sum);**

**printf("---------------------------------\n");**

**}**

**C language support integer-based and float-based data types. C Language support signed and unsigned literal.**

**Signed data can hold +ve and – ve values.**

**Unsigned data types can hold large value as well +ve values.**

**int a;**

**float b;**

**char c;**

**double d;**

**short e;**

**long int f;**

**long double g;**

**#include<stdio.h>**

**#include<stdlib.h>**

**void main() {**

**char c1='a';**

**char c2 = 97;**

**printf("Size of character is =%d\n",sizeof(c1));**

**printf("value of c1 is =%c\n",c1);**

**printf("Size of character is =%d\n",sizeof(c2));**

**printf("value of c2 =%d\n",c2);**

**printf("value of c2 =%c\n",c2);**

**}**

**#include<stdio.h>**

**#include<stdlib.h>**

**void main() {**

**int a=100;**

**short b=200;**

**long int c=300;**

**long long int d=400;**

**int e = -100;**

**printf("The value of a is =%d\n",a);**

**printf("The size of a is =%d\n",sizeof(a));**

**printf("The value of b is =%d\n",b);**

**printf("The size of b is =%d\n",sizeof(b));**

**printf("The value of c is =%d\n",c);**

**printf("The size of c is =%d\n",sizeof(c));**

**printf("The value of d is =%d\n",d);**

**printf("The size of d is =%d\n",sizeof(d));**

**printf("The value of e is =%d\n",e);**

**}**

**Operators :**

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

**Relational Operator : >, >=, <, <=, ==, !=**

**Assignment operator =**

**Conditional operator , &&, ||, !**

**Increment and decrement : ++, --**

**Bitwise operator : \*, |**

**Ternary operator : condition:?true:false;**

**Bitwise operator**

**#include<stdio.h>**

**#include<stdlib.h>**

**void main() {**

**int a=2; //0010**

**int b=3; //0011**

**int result1,result2;**

**result1 = a&b; // 0010 + 0011 = 0010**

**result2 = a|b; // 0010 + 0011 = 0011**

**printf("Bitwise & =%d\n",result1);**

**printf("Bitwise | =%d\n",result2);**

**}**

**Ternary operator : shortcut of if else statement**

**Pre and post (increment or decrement ) if your using variable alone. That time pre and post are increment by 1 only.**

**But if you store or use in some expression or displaying value in printf then pre and post work differently.**

**Pre means first increment and use it or display it**

**Post means first store or use or display and then increment.**

**Operator example**

**#include<stdio.h>**

**#include<stdlib.h>**

**void main() {**

**int a=10;**

**int b;**

**b = a;**

**printf("The value of b is = %d\n",b);**

**b=++a; //increment and store**

**printf("The value of b is = %d\n",b);**

**b=a++; //store and increment**

**printf("The value of b is = %d\n",b);**

**/\***

**int a=10;**

**printf("The value of a is = %d\n",a);**

**++a; //pre-increment or pre-decrement**

**printf("The value of a is = %d\n",a);**

**a++; //post increment or post-decrement**

**printf("The value of a is = %d\n",a);**

**\*/**

**/\*int a=4;**

**int b=3;**

**int result1;**

**result1 = a>b?a:b; //exp1?exp2:exp3;**

**printf("Result is %d\n",result1);**

**\*/**

**/\*int a=2; //0010**

**int b=3; //0011**

**int result1,result2;**

**result1 = a&b; // 0010 + 0011 = 0010**

**result2 = a|b; // 0010 + 0011 = 0011**

**printf("Bitwise & =%d\n",result1);**

**printf("Bitwise | =%d\n",result2);\*/**

**}**

**If statement :**

1. **simple if**

**if(codidition) {**

**}**

1. **if else**

**if(coditions) {**

**block1**

**}else {**

**block2**

**}**

1. **nested if**

**if(condition1) {**

**if(condition2) {**

**block 1**

**}else {**

**block 2**

**}**

**}else {**

**if(condition3) {**

**block3**

**}else {**

**block4**

**}**

**}**

1. **if else if or if ladder**

**if(condition1) {**

**}else if(conditions2) {**

**}else if(conditions3) {**

**}else {**

**}**

**switch statement**

**user can decide which block do you want to execute.**

**Syntax**

**switch(variableName) {**

**case lable1: block1;**

**break;**

**case lable2: block2;**

**break;**

**case lable3: block3;**

**break;**

**default : defaultblock;**

**break;**

**}**

**switch, case, break and default are keywords.**

**VariableName must be type of int or char not float/double.**

**#include<stdio.h>**

**#include<stdlib.h>**

**void main() {**

**int ch;**

**printf("Plese enter the nubmer between 1 to 4\n");**

**scanf("%d",&ch);**

**switch(ch) {**

**case 1:printf("1st block");**

**break;**

**default:printf("Wrong choice");**

**case 2:printf("2nd block");**

**break;**

**case 3:printf("3rd block");**

**break;**

**}**

**printf("\nFinish");**

**}**

**Looping**

**Initialization : Start and Stop**

**Condition true**

**Increment / decrement**

**While loop : Entry loop**

**Do while loop : Exit loop**

**For loop : fixed iteration loop**

**i=0;**

**n=10;**

**while(i<=n) {**

**do the task**

**i++ or n--**

**}**

**i=0;**

**n=10;**

**do {**

**do the task**

**i++ or n--**

**}while(i<=n);**

**Syntax for loop**

**1 2 4**

**for(initialization;condition;increment/decrement){**

**body of the loop 3**

**}**

**Initialization only once and loop will execute till the condition becomes false.**

**English, GK, Math**

**3 3 3**

**do {**

**1: English, 2: GK, 3 :Math : choice**

**switch() {**

**case 1: English**

**break**

**case 2: GK**

**break**

**case 3: Math**

**break**

**default : no option to write the exam**

**break**

**}**

**Do you want to continue. (y/n| 1 /0)-🡪**

**}while();**

**Every block you have to visit mandatory and only one time.**

**30 min :**

**array : array is a user-defined data type which help to store more than one value of same types.**

**int a=10;**

**a=20;**

**syntax**

**datatype arrayName[size];**

**functions : function is use to do reusability of the logic or code.**

**Types of functions**

1. **function no passing parameter and no return type**
2. **function passing parameter and no return type**
3. **function no passing parameter but return type**
4. **function passing parameter and return type.**

**string : combination of one or more than one character enclosed in double quote or character array is known as string.**

**In C every string end with null character \0.**

**#include<stdio.h>**

**void main() {**

**char a='R';**

**char name[10]={'R','a','j','D','e','e','p'};**

**char desg[]={"Trainer"};**

**printf("A character value is = %c\n",a);**

**printf("name 1st character value is = %c\n",name[0]);**

**printf("name 2nd character value is = %c\n",name[1]);**

**printf("name 3rd character value is = %c\n",name[2]);**

**printf("name is = %s\n",name);**

**printf("name is = %s\n",desg);**

**}**

**#include<stdio.h>**

**void main() {**

**char name[20];**

**char name1[20];**

**printf("Enter the name\n");**

**gets(name);**

**printf("Name is = %s\n",name);**

**/\***

**printf("Enter name\n");**

**scanf("%s",&name);**

**printf("The name is = %s\n",name);**

**printf("Enter name\n");**

**scanf("%c",&name1);**

**printf("The name is = %c\n",name1);**

**\*/**

**}**

**Case 1: English**

**User – defined functions 3**

**Q1:**

1. **1**
2. **2**
3. **3**
4. **4**

**TotalEnglishMarks = 30/20/10/0**

**Case 2:GK**

**User – defined function 3**

**Case 3: Math**

**User – defined function 3**

**Answer type may be integer, float, char, string (char array)**

**Day 13 : 11/05/20**

**String function : string.h 🡪 library contains lot of pre-defined functions**

**strcmp: compare to string**

**strleng : length of string**

**strcpy : copy the string**

**strcat : concate the string**

#include<stdio.h>

#include<string.h>

void main() {

char name1[10];

char name2[10];

printf("Enter first name1\n");

scanf("%s",&name1);

printf("Enter second name2\n");

gets(name2); //enter key

gets(name2);

if(strcmp(name1,name2)==0){

printf("Equal");

}else {

printf("Not Equal");

}

}

#include<stdio.h>

#include<string.h>

void main() {

char name1[10];

char name2[10];

char name3[]={'R','a'};

char name4[]={"Raj"};

printf("Enter first name1\n");

scanf("%s",&name1);

printf("Enter second name2\n");

gets(name2); //enter key

gets(name2);

if(strncmp(name1,name2,2)==0){

printf("Equal");

}else {

printf("Not Equal");

}

}

#include<stdio.h>

#include<string.h>

void main() {

char name[]={"Raj Deep"};

int len;

len = strlen(name);

printf("Number of character is = %d\n",len);

//find the length of string without strlen or any other function.

}

#include<stdio.h>

#include<string.h>

void main() {

char fname[10]={"Raj"};

char lname[10]={"Deep"};

printf("First name is %s\n",fname);

printf("Last name is %s\n",lname);

//strcpy(fname,lname); //copy 2nd parameter value to 1st parameter.

strncpy(fname,lname,1);

printf("First name is %s\n",fname);

printf("Last name is %s\n",lname);

}

#include<stdio.h>

#include<string.h>

void main() {

char fname[10]={"Raj"};

char lname[10]={"Deep"};

printf("First name is %s\n",fname);

printf("Last name is %s\n",lname);

strcat(fname,lname);

printf("First name is %s\n",fname);

printf("Last name is %s\n",lname);

}

Structure : Structure is user-defined data types which help to store different types of values.

Syntax

struct structureName {

int a;

float b;

char c;

double d;

};

Here a, b, c and d are known as structure members.

Inside a main we can’t access structure member directly we have to create structure variable.

void main() {

struct structureName structureVariableName;

printf(“%d”,structureVariableName.strctureMember);

}

#include<stdio.h>

#include<string.h>

struct Abc {

int a;

float b;

};

void main() {

struct Abc s1;

struct Abc s2={10,10.10};

struct Abc s3;

printf("Welcome to Structure Program\n");

printf("s1 variable a value = %d\n",s1.a);

printf("s1 variable b value = %f\n",s1.b);

printf("s2 variable a value = %d\n",s2.a);

printf("s2 variable b value = %f\n",s2.b);

printf("Enter the value of a \n");

scanf("%d",&s3.a);

printf("Enter the value of b \n");

scanf("%f",&s3.b);

printf("s3 variable a value = %d\n",s3.a);

printf("s3 variable b value = %f\n",s3.b);

}

#include<stdio.h>

#include<string.h>

struct Employee {

int id;

char name[10];

float salary;

}emp1[10];

void main() {

printf("Enter the id\n");

scanf("%d",&emp1.id);

printf("Enter the name\n");

scanf("%s",&emp1.name);

printf("Enter the salary\n");

scanf("%f",&emp1.salary);

printf("Id is = %d\n",emp1.id);

printf("Name is = %s\n",emp1.name);

printf("Salary is = %f\n",emp1.salary);

/\*

Structure with array id,name,salary, designation 10

n number of records;

designation --->manager, programmer, clerk

5000 ---> manager

2000---> programmer

1000---> remaining

\*/

}

Union : It is also type of user-defined data type which help to store different types of values.

#include<stdio.h>

#include<string.h>

struct Abc {

int a, b;

/\*int a;

float b;

char c;

double d;\*/

};

union Xyz {

int a, b;

/\*int a;

float b;

char c;

double d;\*/

};

void main() {

struct Abc obj1;

union Xyz obj2;

printf("Size of Structure= %d\n",sizeof(obj1));

printf("Size of Onion= %d\n",sizeof(obj2));

obj1.a=10;

obj1.b=20;

obj2.b=20;

obj2.a=10;

printf("Value of a from struct= %d\n",obj1.a);

printf("Value of b from struct= %d\n",obj1.b);

printf("Value of a from union= %d\n",obj2.a);

printf("Value of b from union= %d\n",obj2.b);

}

/\*union Employee {

int id;

char name[10];

float salary;

}emp1;

void main() {

printf("Enter the id\n");

scanf("%d",&emp1.id);

printf("Enter the name\n");

scanf("%s",&emp1.name);

printf("Enter the salary\n");

scanf("%f",&emp1.salary);

printf("Id is = %d\n",emp1.id);

printf("Name is = %s\n",emp1.name);

printf("Salary is = %f\n",emp1.salary);

}\*/

**Pointer :**

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

Syntax of pointer variable

dataType \*varibleName;

#include<stdio.h>

void main() {

int a=10;

int \*ptr=&a;

int \*\*ptr1 = &ptr;

printf("The value of a is = %d\n",a);

printf("The address of a variable is = %d\n",&a);

printf("The value of a through ptr variable is = %d\n",\*ptr);

printf("The address a through ptr is = %d\n",ptr);

printf("The address of ptr variable is = %d\n",&ptr);

printf("The value of a through ptr1 pointer variable = %d\n",\*\*ptr1);

printf("The address of ptr through ptr1 = %d\n",ptr1);

printf("The address of ptr1 through ptr1 = %d\n",&ptr1);

}

**Mathematical Operation with variable and pointer variable.**

#include<stdio.h>

void main() {

int a=10;

int \*ptr=&a;

printf("The value of a = %d\n",a); // 10

printf("The value of a = %d\n",\*ptr); //10

a++;

printf("The value of a = %d\n",a); // a=

printf("The value of a = %d\n",\*ptr); // \*ptr =

(\*ptr)++;

printf("The value of a = %d\n",a); // a=

printf("The value of a = %d\n",\*ptr); // \*ptr =

}

Mathematical operation on address.

#include<stdio.h>

void main() {

int a=10;

int \*ptr=&a;

int b=20;

int c=30;

//&a = &a+1;

printf("Address of a through %d\n",&a);//6422296

printf("Address of a through %d\n",ptr);//6422296

printf("Value of a = %d\n",a);

printf("Value of a = %d\n",\*ptr);

printf("Value of a = %d\n",b);

printf("Address of b = %d\n",&b);

printf("Value of c = %d\n",c);

printf("Address of c = %d\n",&c);

ptr = ptr -1;

printf("Address of a through %d\n",&a); //

printf("Address of a through %d\n",ptr); //

printf("Value of = %d\n",\*ptr);

printf("Value of = %d\n",b);

}

**Day 14th : 11/6/2020**

**Pointer with Array**

dataType arrayName[]={v1,v2,v3,v4};

int num[]={10,20,30,40,50};

intr \*ptr = num;

Value using array variable abc[0];

Value using pointer variable (\*ptr+0)

Value using array variable abc[1];

Value using pointer variable (\*ptr+1)

Address using array variable &abc[0]

Address using pointer variable (ptr+0)

Address using array variable &abc[1]

Address using pointer variable (ptr+1)

#include<stdio.h>

void main() {

int num[]={10,20,30,40,50};

int \*ptr = num;

int i;

printf("Value using array variable = %d\n",num[0]);

printf("Value using pointer variable = %d\n",\*(ptr+0));

printf("Address using array variable = %d\n",&num[0]);

printf("Address using pointer variable = %d\n",(ptr+0));

for(i=0;i<5;i++) {

printf("Value using array = %d and pointer %d \n",num[i],\*(ptr+i));

printf("Address using array = %d and pointer %d \n",&num[i],(ptr+i));

}

}

**Call by value and call by reference**

#include<stdio.h>

void cal(int x, int y); // function declaration or function prototype

void main() {

int a=10;

int b=20;

printf("Before calling \n");

printf("\nThe value of a is %d and b is %d",a,b);

cal(a,b); //here a and b actual parameter

printf("\nAfter calling \n");

printf("\nThe value of a is %d and b is %d",a,b);

}

void cal(int x, int y) { // x and y formal parameter here call by value;

//printf("call function \n");

x=x+1;

y=y+1;

printf("\nThe value of x is %d and y is %d",x,y);

}

Call by Reference

#include<stdio.h>

void cal(int \*x, int \*y); // function declaration or function prototype

void main() {

int a=10;

int b=20;

printf("Before calling \n");

printf("\nThe value of a is %d and b is %d",a,b);

cal(&a,&b); //here a and b actual parameter

printf("\nAfter calling \n");

printf("\nThe value of a is %d and b is %d",a,b);

}

void cal(int \*x, int \*y) { // x and y formal parameter here call by reference;

//printf("call function \n");

\*x=\*x+1;

\*y=\*y+1;

printf("\nThe value of x is %d and y is %d",\*x,\*y);

}

Add, Sub, Mul and Div

But only method and don’t give return type by (means void)

All all operation you have to in cal() function.

And display the output in main method.

**Call by reference more than one value**

#include<stdio.h>

void cal(int x, int y, int \*sum,int \*sub);

void main() {

int a=10;

int b=20;

int add, sub, mul, div;

cal(a,b,&add,&sub);

printf("Sum of two number is = %d\n",add);

printf("Sub of two number is = %d\n",sub);

}

void cal(int x, int y, int \*sum,int \*sub) {

\*sum = x+y;

\*sub = x-y;

}

**Pointer with string**

#include<stdio.h>

void main() {

char name1[]={'R','a','j'};

char name2[]={"Raj Deep"};

char name3[10]={"Raj"};

char \*name4 = "Welcome to Pointer with String";

printf("name1 is = %s\n",name1);

printf("name2 is = %s\n",name2);

printf("name3 is = %s\n",name3);

printf("name4 is = %s\n",name4);

}

**Dynamic memory allocation :**

Creating memory at the run time.

int abc[10]; fixed size.

Malloc : Memory allocation

dataType \*ptr;

ptr = (dataType\*)malloc(n\*sizeof(datatype));

base upon the n size.

#include<stdio.h>

#include<stdlib.h>

void main() {

int \*ptr;

int n,i;

printf("How many number do you want to store\n");

scanf("%d",&n);

ptr =(int\*)malloc(n\*sizeof(int)); //dynamic memory get created

//depending ou on the n size.

printf("Enter the number one by one\n");

for(i=0;i<n;i++) {

scanf("%d",(ptr+i)); //&abc[i] which if fixed.

}

printf("The numbers are \n");

for(i=0;i<n;i++) {

printf("Value is = %d\n",\*(ptr+i)); // abc[i]

}

free(ptr); //clear dynamic memory.

}

pointerVariable = (dataType\*) calloc(size,sizeOfDataType)

ptr =(int\*)calloc(n,sizeof(int));

English()

Gk()

Maths();

ExamNumber

Name :

Pointer get all marks in main function

Display the result :

80>

Java :

JavaSE (Java Standard Edition)

Basic Programming

OOPs Concept Using Java

Java packages user-defined and pre-defined packages.

Exception Handling

Overview File handling

Collection Framework (Data Structure)

JDBC (Pro\* C) --🡪

Java8 Features

Web Application : JavaEE

Overview of XML, Servlet, JSP

EJB (Theory)

JAX\_Rs (RESTFull Web Service)

Spring Framework

Spring core

Spring MVC

Spring boot

Spring Data (Database Connectivity)

C

OOPs

C + OOPs = C++

C with classes

C++ is partial object oriented programming language.

And Platform dependent(OS).

Java : Java is pure object and platform independent programming language.

1990

Oak initial name of is is Oak.

Rename to Java in Nov 1995.

James gosling and Team

Java was belong to Sun Microsystem but now it is a part of Oracle.

Version of Java 1.0 to 14.x

Stable Java in Organization Java8, Java9 and Java11.

class : it is use-defined data type like Structure.

class ClassName {

variables;

methods;

main method (pre-defined method)

}

ClassName follow Pascal naming rule

Means If class contains one world first letter in upper class if class contains more than one world each world first letter upper class.

IDE :

class Demo {

public static void main(String abc[]) {

System.out.print(“”)

System.out.printf(“”)

System.out.println(“”);

}

}

Save the program

ClassName.java

**javac : Java compiler**

**java : Java interpreter**

**Both are translator converting one format to another format. Compiler convert whole code at time. Interpreter convert line by line.**

**javac filename.java (Source file)**

**java filename (Byte code).**

**JVM : Java Virtual Machine.**

**JDK : Java Development Kit**

**Compile the program**

**JRE : Java Runtime environment**

**Run the program**

**JVM**

**Basic programming using Java**

**First Welcome Program**

**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....");**

**}**

**}**

**DataTypes :**

In C language +ve or –ve number is consider as true

And 0 consider as false.

2 Types

1. Primitive data types : It is use to store only value.
   1. byte : 1 btye
   2. short : 2 byte
   3. int : 4 byte
   4. long : 8 byte

without decimal

* 1. float : 4 byte
  2. double : 8 byte

with decimal

* 1. char : 2 byte (Unicode) any single character
  2. boolean : 1 bit true or false.

1. Non primitive data types or reference data types : It is use to store value as well as reference of another data byte.
   1. array
   2. class (user-defined or pre-defined)
   3. interface (user-defined or pre-defined)
   4. enum (user-defined or pre-defined)

class Demo {

public static void main(String args[]) {

int a=10;

System.out.println(a);

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

System.out.printf("The value of a is %d\n",a);

}

}

Declare all type of variable with values and display it.

class Demo {

public static void main(String args[]) {

byte a=10;

short b=20;

int c =30;

long d = 40;

System.out.println(" "+a+" "+b+" "+" "+c+" "+d);

System.out.printf("%d, %d, %d, %d, ",a,b,c,d);

}

}

**Type casting :**

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

1. Implicit type casting
2. Explicit type casting

class Demo {

public static void main(String args[]) {

float a =10.10f;

System.out.println(a);

}

}

int family

implicit -----------🡪

byte ---🡪 short ---🡪 int ----🡪 login

🡨--------Explicit

dataType variableName = (dataType)varialbeName;

int family

class Demo {

public static void main(String args[]) {

byte a=10; // range -128 to 127

short b =a;

System.out.println(a);

System.out.println(b);

short c =130; // 128, 129, 130

byte d =(byte)c; //Explicit type castig.

System.out.println(c);

System.out.println(d);

}

}

**int to float**

implicit --🡪

int float

🡨-- explicit

class Demo {

public static void main(String args[]) {

long a=2147483648l;

byte b = (byte)a;

System.out.println(a);

System.out.println(b);

}

}

Operator :

If statement

Simple if

Nested if

if else if

switch statement

loop

while loop

do while loop

for loop

array : collection of more than one value of same type.

Syntax

dataType arrayName[];

int abc[]; //no size.

int []abc;

int [] abc;

int[] abc;

arrayName.length : which help to find the size of the array.

**For each loop or enhanced loop**

Syntax

for(dataType variableName : arrayName) {

}

class Demo {

public static void main(String args[]) {

int abc[]={10,20,30,40,50,56,23,78,90,12,56,98};

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

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

for(int i=2;i<8;i=i+2) {

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

}

System.out.println("For each loop");

for(int a:abc) {

System.out.println(a);

}

}

}

Syntax

dataType array[]=new dataType[size];

int abc[]=new int[10]; // like int abc[10] in C

Four ways

1. Using Scanner class
2. Using BufferedReader
3. Using DataInputStream
4. Using command line arguments

Scanner is pre-defined class which help to take the value through keyboard.

Syntax

Scanner obj = new Scanner(System.in);

Scanner class part of util package. Package is a collection of classes and interfaces.

import java.util.\*; //like a header file

class Demo {

public static void main(String args[]) {

Scanner obj = new Scanner(System.in);

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

int id = obj.nextInt(); //method follow camelNaming rules

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

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

String name = obj.next();

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

/\*

.nextByte()

.nextShort();

.nextInt();

.nextLong()

.nextFloat();

.nextDouble();

.nextBoolean();

but no .nextChar()

.next() it help to take string value throug keyboards

Through keyboard ask how number employee details do you want to store.

id,name,salary of array type.

int id[]=new int[n];

then receive all employee details ie id,name,salary for all n employee

hra, da, pf

where hra is 10% on salary

da is 5% on salary

pf is 7% on salary

id,name,salary(GrossSalary)

\*/

}

}

**Day 15 : 9/11/2020**

import java.util.\*;

class Demo {

public static void main(String args[]) {

Scanner obj = new Scanner(System.in);

System.out.println("How many number do you want store\n");

int n = obj.nextInt();

int num[]=new int[n]; //like malloc

System.out.println("Enter the number one by one");

for(int i=0;i<n;i++) {

num[i]=obj.nextInt();

}

for(int i=0;i<n;i++) {

if(num[i]>20) {

num[i]=num[i]-5;

}else {

num[i]=num[i]+5;

}

}

for(int i=0;i<n;i++) {

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

}

}

}

**Java OOPs Concept**

object : object is a any real world entity.

Example

Name,age, h, c –int, float, char, double

String,

Person

Talking(), walking(), sleeping(), etc

Bank

Animal

Car

Bike

Properties (state)—have – fields / variables

Wheel, price, color etc

Car

Behavior –do/does – functions / methods

Start, appliedGear, moving, stop

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

Class is a user-defined data type which help to describe the object

Object creation in Java

ClassName objectReferenceName = new ClassName();

class Car {

int wheel;

String color;

float price;

void start() {

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

}

void appliedGear() {

}

void moving() {

}

void stop() {

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

}

}

class CarTest {

public static void main(String args[]) {

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

//start();

Car bmw = new Car();

bmw.start();

bmw.stop();

}

}

**Types of Variable or Fields**

1. types
2. Instance variable :
   1. The variable which declare outside a method including main method but inside a class is known as instance variable.
   2. The instance variable hold default value according to their data types.
   3. Instance variable we can access within all methods but the methods must be part of same class as well as it is must non static method.
3. Local variable :
   1. The variable which declare inside a method including main method is known as local variable.
   2. Local variable doesn’t hold any default value we have to initialize.
   3. The scope of the variable within that block where it declare.
4. Static variable :

Employee

id,name,salary 🡪 instance variable

Scanner obj;

read() receive id,name,salary

calSalary()

3 local variable

Hra, da, pf

display()

id,name,salary(GrossSalary)

EmployeeTest

Main method

Create the object

Call read method

Syntax to create the array object

className emp[]=new ClassName[size];

Employee emp[]=new Employee[10];

emp[0]=new Employee();

emp[1]=new Employee();

emp[0].read()

emp[0].calSalary();

0

1

10

None of these

Employee emp = new Employee();

import java.util.\*;

class Employee {

int id;

String name;

float salary;

Scanner obj = new Scanner(System.in);

void read() {

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

id = obj.nextInt();

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

name = obj.next();

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

salary = obj.nextFloat();

}

void calSalary() {

float hra = salary\*0.10f;

float da = salary\*0.05f;

float pf = salary\*0.07f;

salary = salary +hra +da -pf;

}

void display() {

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

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

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

}

}

class EmployeeTest {

public static void main(String args[]) {

/\*Employee emp1 = new Employee();

emp1.read();

emp1.calSalary();

emp1.display();

Employee emp2 = new Employee();

emp2.read();

emp2.calSalary();

emp2.display(); \*/

Scanner obj = new Scanner(System.in);

System.out.println("how many record do you wan to store");

int n = obj.nextInt();

Employee emp[]=new Employee[n];

System.out.println("Enter the records one by one");

for(int i=0;i<n;i++){

emp[i]=new Employee();

emp[i].read();

emp[i].calSalary();

}

System.out.println("Number of records are ");

for(int i=0;i<n;i++) {

emp[i].display();

}

}

}

**Constructor : It is a type of special method which help to create the object**

**Pts**

**Constructor have same name as class itself.**

**Constructor doesn’t contains return type not even void also**

**Constructor no need to call explicitly it will call automatically when we create the objects.**

**Constructor Simple Example**

**import java.util.\*;**

**class Employee {**

**Employee() {**

**System.out.println("object created....");**

**}**

**void display(){**

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

**}**

**}**

**class EmployeeTest {**

**public static void main(String args[]) {**

**Employee emp = new Employee();**

**//emp.display();**

**}**

**}**

**import java.util.\*;**

**class Employee {**

**void Employee() {**

**System.out.println("object created....");**

**}**

**void display(){**

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

**}**

**}**

**class EmployeeTest {**

**public static void main(String args[]) {**

**Employee emp = new Employee();**

**emp.display();**

**}**

**}**

**If we doesn’t write any constructor then default constructor is available in all OOPs language.**

**But when write explicitly empty or parameter then Java compiler doesn’t provide any default constructor.**

**In the life of the object if you want to perform task only one time that type task we have to write inside a constructor (it may be default or parameterized).**

**In the life of the object if we want to perform any task more than one time that type of task we have to write inside a methods.**

**Method and constructor code**

**import java.util.\*;**

**class Operation {**

**int a,b,sum;**

**Operation() {**

**int a=10;**

**int b=20; //local variable**

**this.a = 100;**

**this.b =200;**

**}**

**Operation(int a, int b){ // parameter variable**

**this.a=a;**

**this.b=b;**

**}**

**void setValue(int a, int b) {**

**this.a =a;**

**this.b =b;**

**}**

**void addOperation() {**

**sum = a+b;**

**}**

**void display() {**

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

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**Operation op1 = new Operation(1,2);**

**op1.addOperation();**

**op1.display();**

**Operation op2 = new Operation();**

**op2.addOperation();**

**op2.display();**

**Operation op3 = new Operation(1,2);**

**op3.addOperation();**

**op3.display();**

**Operation op4 = new Operation(4,8);**

**op4.addOperation();**

**op4.display();**

**Operation op5 = new Operation();**

**op5.setValue(1,2);**

**op5.setValue(5,6);**

**op5.setValue(10,20);**

**op5.addOperation();**

**op5.display();**

**}**

**}**

**GC : Auto Garbage Collector**

**obj1 = null;**

**import java.util.\*;**

**class Operation {**

**int a;**

**Operation() {**

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

**}**

**void display() {**

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

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**Operation op1, op2, op3; //**

**int b;**

**op1 = new Operation(); // one memory get created...**

**op1.a=10;**

**op2 = new Operation();**

**op2.a=20;**

**op3 = op1;**

**op1.display(); // a=10**

**op2.display(); // a=20**

**op3.display(); // a=10**

**op3.a=30;**

**op1.a=40;**

**op2.a=50;**

**op1.display(); // a=40**

**op2.display(); // a=50**

**op3.display(); // a=40**

**op1 = null;**

**op2 = null;**

**//op1.display();**

**//op2.display();**

**op3.display();**

**}**

**}**

**Encapsulation : Binding or wrapping data(fields/variables) and code(functions/methods) in a single unit is known as Encapsulation.**

**class Employee {**

**fields;**

**methods;**

**}**

**class Employee {**

**private int id;**

**private String name;**

**private float salary;**

**void setValue(int id, String name, float salary) {**

**this.id =id;**

**this.name=name;**

**//this.salary = salary;**

**if(salary<0) {**

**this.salary = 8000;**

**}else {**

**this.salary = salary;**

**}**

**}**

**void display() {**

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

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

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

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**Employee emp = new Employee();**

**//emp.id =100;**

**//emp.name="Ravi";**

**//emp.salary=-14000;**

**emp.setValue(100,"Ravi",-12000);**

**emp.display();**

**//id=100;**

**//display();**

**}**

**}**

**Inheritance : It is use to acquire or inherits the properties and behavior of old class to new class.**

**class OldClass { //super class, base class or parent class**

**fields;**

**methods;**

**}**

**class NewClass extends OldClass{**

**fields;**

**methods;**

**} //sub class, derived class or child class**

**class A {**

**void dis1() {**

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

**}**

**}**

**class B extends A{**

**void dis2() {**

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

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**A obj1 = new A();**

**B obj2 = new B();**

**obj1.dis1();**

**obj2.dis2();**

**obj2.dis1();**

**}**

**}**

**Type of Inheritance**

* **Single Inheritance : One super class and one sub class**

**class A {}**

**class B extends A{}**

* **Multilevel Inheritance : One super and n number of sub classes connected one by one**

**class A {}**

**class B extends A{}**

**class C extends B{}**

**class D extends C {}**

* **Hierarchical Inheritance : One super class n number of sub classes connected directly to super class.**

**class A {}**

**class B extends A{}**

**class C extends A {}**

**class D extends A {}**

* **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.**

**OOPs Relationship**

* **is a relationship : Inheritance** 
  + **Manager/Programmer is a Employee**
* **has a relationship**
  + **Employee has a Address**

**class Employee {**

**id,name,salary**

**Scanner obj = new Scanner();**

**skillset[];**

**Address add = new Address();**

**read() id,name,salary,skillset {**

**add.readAddress();**

**}**

**calSalary()**

**display();**

**}**

**class Manager extends Employee{**

**numberOfEmp; 5 / 10**

**readMgr(); receive numberOfEmp**

**{**

**}**

**disMgr(); display numberOfEmp**

**}**

**class Programmer extends Employee{**

**projectName; string**

**readPrg();**

**disPrg();**

**}**

**class Address {**

**city,state**

**Scanner obj=new Scanner(System.in);**

**readAdd() receive city and state**

**disAdd()**

**}**

**Has a**

* + - 1. **Association**
      2. **Aggregation :**
      3. **Composition**

**class A {**

**//B obj = new B();**

**}**

**class B {**

**A obj = new A();**

**}**

**0-1**

**0-infinity**

**1-0**

**Infinity -0**

**class Employee {**

**Address add = new Address(); one or more**

**}**

**class Address {**

**}**

**class Student {**

**StudentHistory sh = new StudentHistory();**

**}**

**class StudentHistory {**

**}**

**Day 16 : 10/11/20**

**Polymorphism : One name many form or many implementation.**

**2 types**

1. **Compile time polymorphism**

**javac**

**static binding or early binding**

**Ex: Method Overloading**

**Method Overloading : The method have same name but different parameter list (number of parameter list as well as type of parameter list must differ) but return type must be same.**

1. **Run time polymorphism**

**java**

**run time polymorphism or late binding**

**Method Overriding**

**Method Overloading**

**class Operation {**

**void add(int x, int y) {**

**System.out.println(x+y);**

**}**

**void add(int x, int y, int z) {**

**System.out.println(x+y+z);**

**}**

**void add(float x, float y) {**

**System.out.println(x+y);**

**}**

**void add(String s1, String s2) {**

**System.out.println(s1+s2);**

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**Operation op = new Operation();**

**op.add(1,2);**

**op.add("1","2");**

**op.add("Raj"," Deep");**

**op.add(10.10f,20.20f);**

**op.add(1,2,3);**

**}**

**}**

**Method Overriding : The method have same name and same method signature (number of parameter list, type of parameter list as well as return type must be same).**

**To achieve the method overriding class must be in inheritance.**

**class Bike {**

**void speed() {**

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

**}**

**}**

**class Pulsar extends Bike {**

**void color() {**

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

**}**

**void speed() { //method override**

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

**}**

**}**

**class Honda extends Bike {**

**void color() {**

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

**}**

**}**

**class Tvs extends Bike {**

**void color() {**

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

**}**

**void speed() { //merge**

**super.speed(); //calling super class speed() method code**

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

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

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

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

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

**}**

**}**

**abstract : abstract is a keyword we can use with method and class but not with variable.**

1. **abstract method : The method without body or without curly braces or incomplete method is known as abstract method.**

**Syntax**

**abstract returnType methodName(parameteList);**

**abstract void speed();**

1. **abstract class : To declare the class as a abstract we have to use abstract keyword before class keyword.**

**Syntax**

**abstract class className {**

**}**

1. **if class contains abstract method then we have to declare the class as an abstract class.**
2. **Abstract class can contains 1 or many or zero abstract method.**
3. **If class is a abstract we can’t create the object of that class.**
4. **Abstract class can contains default as well as we can write parameterized constructor. (abstract class can contains instance variable).**
5. **Which ever class extends abstract class that class must be provide the body for all abstract method mandatory. That class can ignore only if that class itself is a abstract.**

**abstract class Bike {**

**abstract void speed();**

**}**

**abstract class Pulsar extends Bike {**

**void color() {**

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

**}**

**abstract void mailage();**

**}**

**class SuperPulsar extends Pulsar {**

**void mailage() {**

**System.out.println("20km/lt");**

**}**

**void speed() {**

**System.out.println("150km/hr");**

**}**

**}**

**class Honda extends Bike {**

**void color() {**

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

**}**

**void speed() { //method override**

**System.out.println("50km/hr");**

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

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

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

**SuperPulsar sp = new SuperPulsar(); sp.speed(); sp.mailage(); sp.color();**

**}**

**}**

**Final keyword : final is a keyword we can use with variable, method and class.**

1. **final variable : to declare a constant in java we use final keyword.**

**Syntax**

**final int A=10;**

1. **final method : If method is a final we can’t override that method.**

**But method can access or call by sub class object.**

**final void speed() {**

**}**

1. **final class : if class is a final we can’t extends that class can’t inherits that class.**

**final class Demo {**

**}**

**final class Bike {**

**final void speed(){**

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

**}**

**}**

**class Honda extends Bike {**

**void color() {**

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

**}**

**/\*void speed() { //method override**

**System.out.println("50km/hr");**

**}\*/**

**}**

**class Test {**

**public static void main(String args[]) {**

**final int A=10;**

**System.out.println(A);**

**//A = 20;**

**System.out.println(A);**

**Honda hh = new Honda();**

**hh.speed();**

**}**

**}**

**We can’t use final and abstract together.**

**static : static is keyword we can use with variable and method but not with class. (If class is inner class then we can use static class but not for outer class.)**

1. **if method is static we can access that method with help of className as well as objectRereference.**
2. **If variable is static we can assign the value for that variable using className as well as through object reference.**
3. **Inside a non-static method we can access non static we well as static variable directly. But inside a static method we can access only static variable directly.**

**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 obj = new Abc();**

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

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

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**Abc.dis2();**

**Abc obj1 = new Abc();**

**obj1.dis1();**

**obj1.dis2();**

**obj1.a=10;**

**obj1.b=20;**

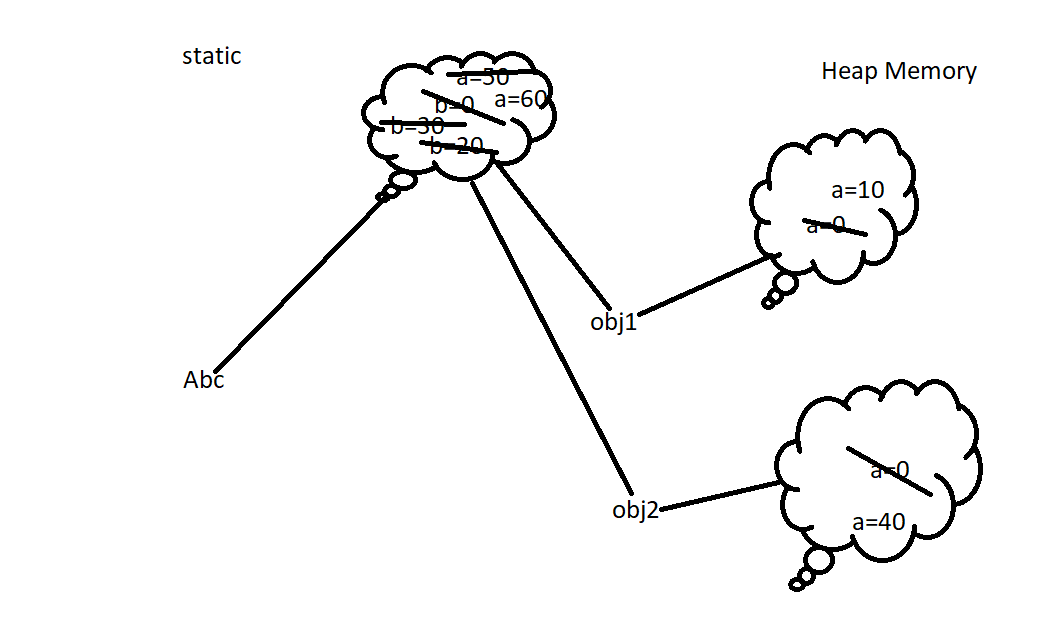
**Abc.b=30;**

**}**

**}**

**Heap Memory or Instance memory**

**Static memory**

****

**class Abc {**

**int a;**

**static int b;**

**void dis1() {**

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

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

**}**

**static void dis2() {**

**//System.out.println(a);**

**System.out.println(b);**

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**Abc.dis2();**

**/\*Abc obj1 = new Abc();**

**Abc obj2 = new Abc();**

**obj1.a=10;**

**obj1.b=20;**

**Abc.b=30;**

**obj2.a=40;**

**obj2.b=50;**

**Abc.b=60;**

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

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

**}**

**}**

**Static memory only one copy for every class but instance memory number of copies for objects.**

**Emp🡪id,name,salary,desg 🡪instance**

**MgrId, ClientDetails, CompanyName -🡪 Static**

**Static is a global to all objects.**

**Interface**

**Interface is a type of reference data type which is also known as 100% pure abstract class till Java7 version.**

**Syntax**

**interface interfaceName {**

**fields;**

**methods;**

**}**

**By default all fields or variable are public , static and final.**

**By default all method are public and abstract.**

**interface Abc {**

**public static final int A=10;**

**static final int B=20;**

**final int C=30;**

**int D =40;**

**public abstract void dis1();**

**void dis2();**

**}**

**interface Abc {**

**int A=10;**

**void dis1();**

**}**

**interface Mno {**

**int B=20;**

**void dis2();**

**}**

**interface Xyz extends Abc,Mno{**

**int C=30;**

**void dis3();**

**}**

**Overriding rules using access specifiers**

**Super class/Super interface Sub class**

**public public**

**protected public/protected**

**default(nothing) public/protected/default**

**private can’t override.**

**Difference between abstract class and interface**

1. **Abstract class can contains normal as well as final variable but interface must be contains only final variable.**
2. **Abstract class can contains 1 or more or zero abstract method but interface must be contains only abstract methods.**
3. **Normal class or abstract class can extends only one abstract class but can implements more than one interface.**
4. **Abstract class can contains default as well as we can write parameterized constructor but interface doesn’t contains default constructor.**

**Common**

1. **We can’t create the object of interface as well as abstract class.**

**Run time Polymorphism using object creation**

**class A {**

**void dis1() {**

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

**}**

**}**

**class B extends A {**

**void dis1() {**

**System.out.println("A class override method");**

**}**

**void dis2() {**

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

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

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

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

**A obj3 = new B(); //sub class object and super class reference : Valid**

**//With the help of super class reference we can call**

**//only those method which belong to super class or override method.**

**//B obj4 = new A(); //super class object and sub class reference : InValid**

**obj3.dis1(); //obj3.dis2();**

**}**

**}**

**Run time polymorphism**

**class Bike {**

**void speed() {**

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

**}**

**}**

**class Pulsar extends Bike {**

**void speed() {**

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

**}**

**void color() {**

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

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**Pulsar pu = new Pulsar();**

**pu.speed();**

**pu.color();**

**Bike bb = new Pulsar(); // Run time polymorphsim using object.**

**bb.speed();**

**}**

**}**

**2nd way**

**abstract class Bike {**

**abstract void speed();**

**}**

**class Pulsar extends Bike {**

**void speed() {**

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

**}**

**void color() {**

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

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**Bike bb = new Pulsar(); // Run time polymorphsim using object.**

**bb.speed();**

**}**

**}**

**3rd way**

**interface Bike {**

**void speed();**

**}**

**class Pulsar implements Bike {**

**public void speed() {**

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

**}**

**void color() {**

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

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**Bike bb = new Pulsar(); // Run time polymorphsim using object.**

**bb.speed();**

**}**

**}**

**Sub class object super class/abstract class / interface reference is possible.**

**Abstraction : Hiding the internal implementation without known background details.**

**/\*abstract class A {**

**abstract int add(int x, int y);**

**}**

**abstract class B{**

**abstract int sub(int x, int y);**

**}\*/**

**interface A {**

**abstract int add(int x, int y);**

**}**

**interface B{**

**abstract int sub(int x, int y);**

**}**

**class Server implements A,B{**

**public int add(int x, int y){**

**return x+y;**

**}**

**public int sub(int x, int y){**

**return x-y;**

**}**

**public void ownMethod() {**

**System.out.println("Server side method");**

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**Server ss = new Server();**

**System.out.println(ss.add(1,2));**

**System.out.println(ss.sub(1,2));**

**ss.ownMethod();**

**A obj1 = new Server(); // obj1 is a A interface refernce so with the help of that reference we can call only those method which belong to A interface**

**B obj2 = new Server();**

**System.out.println(obj1.add(1,2));**

**//System.out.println(obj1.sub(1,2));**

**//System.out.println(obj2.add(1,2));**

**System.out.println(obj2.sub(1,2));**

**}**

**}**

**this, this(), super, super()**

**class A {**

**int n=10;**

**}**

**class B extends A {**

**int n=20;**

**void dis1() {**

**int n =30;**

**System.out.println("n "+n);**

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

**System.out.println("super n "+super.n);**

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**B obj1 = new B();**

**obj1.dis1();**

**}**

**}**

**this() : it is use to invoke same class constructor like constructor chaining. It must inside a constructor as well as it must be first statement inside a constructor.**

**class Employee {**

**int id;**

**String name;**

**float salary;**

**Employee() {**

**this.id =123;**

**this.name ="Unknown";**

**this.salary = 8000;**

**}**

**Employee(int id) {**

**this(); // it will call empty constructor without creating fresh memory.**

**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+" Name is "+name+" Salary is "+salary);**

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

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

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

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

**Employee emp4 = new Employee(102,"Mahesh",24000); emp4.dis();**

**}**

**}**

**super() : By default every sub class constructor first statement super() is available . It is use to invoke super class constructor to sub class constructor.**

**It must be first statement inside a constructor.**

**class A {**

**A() {**

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

**}**

**A(int x) {**

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

**}**

**}**

**class B extends A{**

**B() {**

**super(100);**

**System.out.println("B class constructor");**

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**B obj = new B();**

**}**

**}**

**init block and static block**

**class Emp {**

**static {**

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

**}**

**{**

**System.out.println("Init block");**

**}**

**Emp() {**

**System.out.println("Constructor block");**

**}**

**void dis1() {**

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

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**Emp emp1 = new Emp();**

**Emp emp2 = new Emp();**

**emp1.dis1();**

**emp1.dis1();**

**emp2.dis1();**

**}**

**}**

**class Emp {**

**static {**

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

**}**

**{**

**System.out.println("Init block");**

**}**

**Emp() {**

**System.out.println("Constructor block");**

**}**

**void dis1() {**

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

**}**

**}**

**class Test {**

**public static void main(String args[]) {**

**Emp emp1 = new Emp();**

**Emp emp2 = new Emp();**

**emp1.dis1();**

**emp1.dis1();**

**emp2.dis1();**

**}**

**}**

**Exception Handling :**

**Exception is a object which occurs during the execution of program.**

**error**

**compile time error run time error**

**syntax error**

**or typo error Error Exception**

**Error and Exception both are pre-defined classes part of lang package.**

**Package : package is a collection of classes and interface.**

**By default every java program import java package So without import lang package we can use all classes and interfaces belong to lang package.**

**Error : The error which generate at run time which we can’t handle it.**

**Ex : JVM Crash, Software or Hardware issue or out of memory etc.**

**Exception The error which generate at run time which we can handle it.**

**Ex :ArithmeticException etc**

**Object**

**Throwable**

**Error Exception**

**Checked Exception Unchecked Exception**

**(groups) (groups)**

**RuntimeException 🡪pre-defined class**

**SQLException**

**IOException ArithmeticException**

**NumberFormatException**

**etc ArrayIndexOutOfBoundsException**

**NullPointerException**

**Etc**

**To handle both type of exception it provided 5 keywords**

1. **try**
2. **catch**
3. **finally**
4. **throw**
5. **throws**

**try and catch block**

**try {**

**try block**

**}catch(Exception e) {**

**catch block**

**}**

**class Test {**

**public static void main(String args[]) {**

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

**int a=10;**

**int b=0;**

**try{**

**int res = a/b; //new ArithmeticException();**

**System.out.println("Result is "+res);**

**}catch(Exception e) {**

**//System.out.println("I Take Care!");**

**//System.out.println(e.getMessage());**

**//System.out.println(e.toString());**

**e.printStackTrace();**

**}**

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

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

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

**}**

**}**

**Array Index Out of bounds**

**class Test {**

**public static void main(String args[]) {**

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

**int a=10;**

**int b=1;**

**int abc[]={1,2,3,4,5};**

**try{**

**int res = a/b; //new ArithmeticException();**

**System.out.println("Result is "+res);**

**int res1 = 10/abc[6];**

**System.out.println("Result is "+res1);**

**}catch(Exception e) {**

**System.out.println(e.toString());**

**}**

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

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

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

**}**

**}**

**Try with multiple catch block**

**class Test {**

**public static void main(String args[]) {**

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

**int a=10;**

**int b=1;**

**int abc[]={1,2,3,4,5};**

**try{**

**int res = a/b; //new ArithmeticException();**

**System.out.println("Result is "+res);**

**int res1 = 10/abc[4];**

**System.out.println("Result is "+res1);**

**}catch(ArithmeticException e) {**

**System.out.println(e.toString());**

**}**

**catch(ArrayIndexOutOfBoundsException e) {**

**System.out.println(e.toString());**

**}**

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

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

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

**}**

**}**

**Try with multiple catch block with generic exception class**

**class Test {**

**public static void main(String args[]) {**

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

**int a=10;**

**int b=1;**

**int abc[]={1,2,3,4,5};**

**try{**

**int res = a/b; //new ArithmeticException();**

**System.out.println("Result is "+res);**

**int res1 = 10/abc[4];**

**System.out.println("Result is "+res1);**

**}catch(ArithmeticException e) {**

**System.out.println(e.toString());**

**}catch(ArrayIndexOutOfBoundsException e) {**

**System.out.println(e.toString());**

**}catch(Exception e){**

**System.out.println(e.toString());**

**}**

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

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

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

**}**

**}**

**Day 16 : 11/11/2020**

**finally : finally is a block which will execute 100% sure if exception generate or not generate.**

**Finally block which help to close the resources.**

**File Handling**

**Database Connection**

**Exception handling**

**try{**

**open file or connect to data**

**operation on file do task on database (create/delete/update)**

**}catch(Exception e) {**

**}finally {**

**Close the resources**

**}**

**class ExpDemo {**

**public static void main(String args[]) {**

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

**try {**

**int res = 10/1;**

**System.out.println("No Exception");**

**}catch(Exception e) {**

**System.out.println("Catch block");**

**}finally {**

**System.out.println("Finally block");**

**}**

**System.out.println("Normal Statement");**

**}**

**}**

**try – catch**

**try – catch –catch –catch**

**try – catch –finally**

**try –catch – catch – catch –finally**

**try – finally**

**Another try – catch and finally block code**

**class ExpDemo {**

**public static void main(String args[]) {**

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

**try {**

**int res = 10/0;**

**System.out.println("No Exception");**

**}finally {**

**System.out.println("Finally block");**

**}**

**System.out.println("Normal Statement");**

**}**

**}**

**throw : throw keyword is use to raise or generate user-defined(custom) or pre-defined exception depending upon the conditions.**

**Syntax**

**throw new Exception();**

**Or**

**throw new ExceptionSubClass();**

**class MyException extends Exception {**

**MyException() {}**

**MyException(String msg){**

**super(msg);**

**}**

**}**

**class ExpDemo {**

**public static void main(String args[]) {**

**int a=10;**

**int b=5;**

**try{**

**if(a>b) {**

**//throw new Exception();**

**//int res =10/0;**

**//throw new ArithmeticException();**

**//throw new ArithmeticException("a>b");**

**//throw new MyException();**

**throw new MyException("a>b");**

**}**

**}catch(Exception e){**

**System.out.println(e.toString());**

**}**

**}**

**}**

**throws : throws is use to throw checked or unchecked exception to caller methods.**

**returnType methdoName(parameterList) throws Exception,ExceptionSubClass {**

**}**

**class ExpDemo {**

**/\*static String sayHello() {**

**return "Welcome";**

**}\*/**

**static void dis1() throws Exception{**

**//try{**

**int res = 10/0;**

**//}catch(Exception e){}**

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

**}**

**static void dis2() throws Exception{**

**//try{**

**dis1();**

**//}catch(Exception e){}**

**System.out.println("dis2()");**

**}**

**public static void main(String args[]) throws Exception{**

**try{**

**dis2();**

**}catch(Exception e){}**

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

**}**

**}**

**Package : package is a collection of classes and interfaces.**

**2 types**

**User-defined package**

**education**

**school college pg**

**Attendance Attendance Attendance**

**Syntax to create the user define package**

**package com;**

**class ExpDemo {**

**public static void main(String args[]){**

**System.out.println("Welcome to User defined package");**

**}**

**}**

**javac ExpDemo.java**

**Then create folder with packagename and copy/paste .class file inside com directory**

**Then run**

**java com.ExpDemo**

**Pre-defined or built in package**

**Date : util**

**Date : sql**

**java javax --🡪Root package**

**lang swing**

**io sql**

**util servlet**

**sql ejb**

**jms**

**etc**

**API : Application Programming interface. (in the form class, package, functions, modules etc);**

**By default every java program import lang package.**

**By default every java class extends Object class.**

**Lang package**

**String**

**StringBuffer**

**StringBuilder**

**Object class**

**Math**

**Exception and all Exception classes**

**Thread**

**Runnable**

**String : String is pre-define class part of lang package.**

**Syntax**

**String str1 = “Welcome to Java Training”;**

**String str2 = new String(“Welcome to Java Training”);**

**API Details**

**javap java.packageName.className/interface**

**class ExpDemo {**

**public static void main(String args[]){**

**String str1 ="Welcome to Java Training";**

**String str2 = new String("Welcome to Java Training");**

**System.out.println(str1);**

**System.out.println(str2);**

**System.out.println(str1.length());**

**System.out.println(str1.toUpperCase());**

**System.out.println(str1.toLowerCase());**

**System.out.println(str1.substring(4));**

**System.out.println(str1.substring(4,10));**

**}**

**}**

**== Always check value as well as reference or memory or hash code**

**.equals() : it only check value.**

**equals() :This method always check the value for two object reference.**

**String is known as immutable class (can’t change).**

**JavaBean : JavaBean it is a normal Java with set of rules and regulations.**

**In Database**

**Product**

**PId PName Price**

**Product Model Layer :**

**int pid**

**String pname;**

**float price;**

**setter and getter methods.**

**Annotation : It is a meta – data : Data about data.**

**All annotation start with pre-fix @ followed by annotation name.**

**@Override**

**Service Layer : Business Logic**

**Product :**

**Add product**

**GetAllProductDetails**

**GetProductById**

**DeleteProduct**

**UpdateProductPriceUsingId**

**Etc**

**CRUD Operation**

**Product Java Bean**

**Pid,pname,price , setter and getter, constructor(empty and parameterized)**

**Product Service**

**Product pp[]=new Product[10];**

**5 business methods**

**Main class**

**do {**

**switch (){**

**case 1: Add Product : Pass as a JavaBean of product Id must be unique**

**100,101,102,103**

**case 2: delete product using productId**

**case 3: update price using product Id**

**case 4: getAllProductDetails : display in main**

**case 5: getPrice of product Using Id : display in main**

**case 6 : Productorder**

**take oriderId 111**

**orderDate 12-Nov-20**

**getAllProdcutAllDetails();(100,101102,103)**

**qty; 2**

**pass order object to OrderService class method.**

**case 7: display all order**

**case 8: Using orderId generate bill**

**OrderId, ProductName, TotalPrice = amount\*qty+gst**

**}**

**Do you want to continue.**

**}while()**

**class Order {**

**oid**

**orderDate;**

**pid;**

**qty**

**setter and getter methods**

**}**

**class OrderService {**

**List<Orders> listOfOrders = new ArrayList<Orders>();**

**giveOrder(Order ord) {**

**listOfOrders.add(ord);**

**}**

**}**

**Day 17 : 10/16/20**

**Every class / layer for specific purpose.**

**Bean : JavaBean class (Like a Container for property hold property set and get).**

**One class / layer set value another class / layer get value.**

**Service class : Pure Business Logic. Any other give value to Service layer may primitive or objects or array of objects. On those values apply business rules and give the confirmation.**

**Service layer is not responsible to interact with input device like keyboards, browser or any device.**

**Don’t create the Scanner class object in Service layer.**

**Don’t use System.out.println(): in Service layer.**

**Main class (controller ) : Main class is responsible to interact with input device to receive the value from keyboards pass value to service layer and base upon service layer acknowledgement it give confirmation to input device ie keyboards.**

**Access Specifiers :**

**private : instance variable, static variable, non-static method, static method, constructor but not with class and local variable.**

**Scope : Within a same class we can access. If constructor is private we can’t create the object.**

**default : We can use with all .**

**Scope : Within a same package.**

**protected: instance variable, static variable, non-static method, static method, constructor but not with class and local variable.**

**Scope :Within a same package other package if it is a sub class.**

**public : instance variable, static variable, non-static method, static method, constructor and class but not with local variable.**

**Scope : Same package as well as other package.**

**In one editor or notepad or one file only one class must be public not more than that. Which class is public you have save the program with that name only.**

**pack1 pack2**

**import pack1.A;**

**class A { class C extends A{**

**private int a; void dsi2(){**

**int b; c and d**

**protected int c; a (private)**

**public int d; b (default)**

**void dis1() { }**

**a,b,c,d;**

**}**

**} }**

**class Demo { class D {**

**main()**

**A obj1 = new A(); A obj1 = new A();**

**obj1.dis1(); we can’t access a(private) obj1.a (private) can’t**

**obj1.b, obj1.c, obj1.d obj1.b(default can’t**

**obj1.c(protected) can’t**

**obj1.d (public)**

**} }**

**Collection Framework( Data Structure)**

**int a=10; only one value**

**array :**

**int abc[]; we can store more than one value but same type**

**class Employee {**

**int id, String name, float salary;**

**}**

**Employee emp = new Employee(); emp.id=100,emp.name=”Ravi”,emp.salary=12000;**

**Array object**

**Employee emps[]=new Employee[10];**

**emps more than one employee objects.**

**int abc[]={100,200,300,400,500};**

**30**

**abc[2]=30;**

**abc[3]=0;**

**Collection framework contains set of collection classes and interfaces which help to store collection of elements or objects of any type.**

**As well as it provide set of methods which help to store, remove, search, iterate very easily.**

**Collection Framework Hierarchy**

**Collection framework is part of util package.**

**Collection -🡪 interface**

**extends extends**

**List Set Queue Map-🡪 interfaces**

**List, Set and Query internally extends Collection but Map doesn’t extends.**

**List : It store set of elements or objects. It allow duplicate and use index position to store the elements or objects.**

**Set : It store set of elements or objects. It doesn’t allow duplicate. Some API maintain order, un-order as well as sorting (ascending)**

**Queue : First In First Out : Depending upon the priority.**

**Map : In Map we can store data in the form of key-value pairs Where key is unique and value may be duplicate.**

**List : Stack, ArrayList, LinkedList and Vector are type of List classes.**

**These classes directly or indirectly implements List interface.**

**Vector class is known legacy(old class). By default all method in Vector class are synchronized(only one thread can execute at time). So the performance wise Vector is slower other List classes.**

**Set : HashSet, LinkedHashSet, TreeSet are type of Set classes.**

**These classes directly or indirectly implements Set interface.**

**HashSet : Display the elements randomly**

**TreeSet Sort the elements in ascending order (By default).**

**In TreeSet all elements must be same type other wise at run the ClassCastException generate (It is a type of unchecked exception).**

**LinkedHastSet is a sub class of HashSet. Display the elements which ever order we inserted.**

**Arrays and Collections**

**ArrayList al = new ArrayList();**

**List ll = new ArrayList();**

**JavaBean class : Java Technologies**

**POJO : Spring Framework**

**Plain Old Java Object (JavaBean)**

**POJI :**

**Plain Old Java Interface**

**Pojo and Poji classes not to extends or implements pre-defined class.**

**Server Layer**

**Service Interface**

**Service Implements**

**PK NoKey NoKey NoKey Nonkey**

**EmpId EmpName Age DepartmentName ProjectName**

**PK**

**Empid Name Age Projectid ProjectName DeptId Dept**

**Pk FK FK**

**Empid Name Age ProjectId DeptId**

**Pk**

**ProjectId ProjectName…………….**

**PK**

**DeptId DeptName0**

**Queue : PriorityQueue it s type of Queue class.**

**Map : HashMap, LinkedHashMap, TreeMap, Hashtable are type of Map classes directly or indirectly implements Map interface.**

**Map store key-value pairs**

**Where key is unique and value may be duplicate.**

**Accno-name**

**Srno-name**

**Stack : First In Last Out : Method invocation.**

**By default collection framework classes allow to store collection of elements or object of any time.**

**Before 1.5 Java**

**If we want to store any primitive value in Collection in classes we have to covert primitive to object with the help of wrapper classes.**

**8 primitive type 8 wrapper classes**

**byte Byte**

**short Short**

**int Integer**

**long Long**

**float Float**

**double Double**

**char Character**

**boolean Boolean**

**Wrappers is wrap primitive values which help to convert primitive to object and vice-versa.**

**int a=10;**

**Integer b = new Integer(a);**

**Collection Framework with Generics**

//Collection Framework with generics

//ClassName<Type> obj = new ClassName<Type>();

Type may be Integer, Float, Character, String or UserDefinedClassName

**Day 19 : 17/10/20**

**Collection Framework**

**Arrays and Collections : Both are pre-defined classes part of util packages. Both classes provided lot of pre-static methods which help to do operation on primitive array as well as Collection of classes.**

**Arrays mainly provided methods for primitive arrays**

**Collections provided methods for List (ArrayList, LinkedList, Vector)**

**Comparable and Comparator**

**Comparable : it is a interface part of lang package which provide one method ie compareTo(T) (abstract method).**

**By default all wrapper classes (8) + string internally implements Comparable interface and provided implementation to compareTo() method to do sorting for values(primitive as well as string values).**

**Comparator : It is a interface part of util package. Which contains more than one method and two are abstract method compare() and equals.**

**compare() method take two parameter as a object.**

**String name1=”Raj”**

**String name2 = “Ajay”;**

**System.out.println(name2.compareTo(name1));**

**HashCode() and equals() :**

**JDBC : Java Database Connectivity : JDBC is a API which provide set of classes and interfaces which help to Connect RDBMS (oracle or mysql) through Java Technologies.(Like Pro \*C)**

**API : Means application programming interface.**

**Steps to connect the database.**

1. **We have to import java.sql or javax.sql package.**
2. **Main method or user-defined method must be with exception handling concept (try-catch or throws exception.)**
3. **Load the Driver**

**Driver software.**

**Driver is a pre-defined class provided by vendor or database people which help to connect the database through java technologies.**

**Four types of driver**

1. **Type 1 driver or jdbc odbc bridge driver ( From Java 8 onward type 1 driver is deprecated).**
2. **Type 2 jdbc native api driver**
3. **Type 3 jdbc net protocol driver**
4. **Type 4 jdbc pure or thin driver**

**oracle.jdbc.driver.OracleDriver : This is for Oracle**

**1 way**

**Class.forName(“driver Name”)**

**In Java pre-defined class name itself is Class contains forName static method which help to load the Driver.**

1. **Establish the connection**

**DriverManager is a pre-defined class which contains getConnection() method(static methods) which takes three parameter url, username and password and returntype of this method is Connection interface reference.**

**Connection con = DriverManager.getConnection(url,username,password);**

1. **Create a type of statement**

**It is provided set of methods which help to do operation on tables.**

1. **Types**
2. **Statement**
3. **PreparedStatement**
4. **CallableStatement**

**All are interfaces.**

1. **Statement :**

**Statement stmt = con.createStatement();**

1. **DML Operation (Insert / Delete and Update)**

**int result = stmt.executeUpdate(“DML Operation”);**

1. **DRL or DQL (Select clause )**

**stmt.executeQuery(“select \* from employees”)**

**This method return type of ResultSet interface reference.**

**ResultSet rs = stmt.executeQuery(“select \* from employees”);**

**Types of Statement**

1. **Statement : Each time query compile on java side, send query to database, execute in database and get the acknowledgement success or failure.**
2. **PreparedStatement : Query compile only once and execute n number of times. So performance wise PreparedStatement is faster than Statement**

**PreparedStatement support paratermized query concept. Place holder as ?**

1. **CallableStatement**

**DAO Layer : Data Access Object : (Pure JDBC Code).**

**View 🡪 Console (React JS /Angular)**

**App 🡪 Controller Main (Spring Boot with RestFull Web Service)**

**Service 🡪 Service Layer**

**Dao -🡪 JDBC Code Spring data.**

**Resource -🡪 Database Connection**

**Day 20 : 18/11/2020**

One Object in Java Side --- One Record in Database side

Update price using PId

Delete Product Using Pid

getPrice using Pid

display all details in main methods

**Java 8 Features**

**Interface :**

From Java 8 onwards interface can contains method with body.

But the method must be with default or static keyword.

Default methods

interface A {

void dis1();

default void dis2() {

System.out.println("This is default implementation");

}

}

class B implements A {

public void dis1() {

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

}

}

class C implements A {

public void dis1() {

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

}

public void dis2() {

System.out.println("dis2() default method override by C class");

}

}

class Demo {

public static void main(String args[]) {

B obj1 = new B(); obj1.dis1(); obj1.dis2();

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

}

}

Default and static method code

interface A {

void dis1();

default void dis2() {

System.out.println("This is default implementation");

}

static void dis3() {

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

}

}

class B implements A {

@Override

public void dis1() {

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

}

public static void dis3() { // its own method

System.out.println("Static method override");

}

}

class C implements A {

public void dis1() {

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

}

public void dis2() {

System.out.println("dis2() default method override by C class");

}

}

class Demo {

public static void main(String args[]) {

A obj1 = new B(); obj1.dis1(); obj1.dis2();

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

A obj3 = new B(); // Run time polymorphism

//obj3.dis3();

A.dis3();

}

}

Marker interface ­­

Cloneable interface

Serializable interface

Functional interface :

The interface which contains only one abstract method. It can contains more than one default or static methods. That type of interface is known as Functional interface.

**Lambda Expression**

@FunctionalInterface

interface A {

void dis1();

default void dis2() {

System.out.println("This is default implementation");

}

static void dis3() {

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

}

}

class Demo {

public static void main(String args[]) {

}

}

Inner classes

**Class within another class is known as Inner class.**

1. Non static inner class
2. Static inner class
3. Anonymous inner class
4. Local inner class

Non static inner class example

class Outer {

int a;

void dis1() {

System.out.println("non static method of outer class ");

Inner in = new Inner();

in.dis2();

}

class Inner {

int b;

void dis2() {

System.out.println("non static inner class method");

}

}

}

class Demo {

public static void main(String args[]) {

Outer out = new Outer();

out.dis1();

Outer.Inner in = out.new Inner(); //syntax to create the Inner non static class object.

in.dis2();

Outer.Inner in1 = new Outer().new Inner();

in1.dis2();

}

}

Static inner class

class Outer {

int a;

void dis1() {

System.out.println("non static method of outer class ");

}

static class Inner {

int b;

void dis2() {

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

}

}

}

class Demo {

public static void main(String args[]) {

//Outer.Inner in = new Outer().new Inner();

Outer.Inner in = new Outer.Inner();

in.dis2();

}

}

Lambda Expression : From Java 8 onwards Java is also known as Functional programming language because of Lambda Expression.

Function within another function or method within another methods we can write.

Lambda Expression is also known as Anonymous Inner function.

Lambda expression can apply for interface but interface must be functional interface.

Syntax

InterfaceName refereceName = (parameterList)-> body for lambda expression

Anonymous inner class and Lambda expression example

interface A {

void dis1();

}

class B implements A {

public void dis1() {

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

}

}

class Demo {

public static void main(String args[]) {

A obj1 = new B();

obj1.dis1();

A obj2 = new A(){

public void dis1() {

System.out.println("A interface method override by Anonymous class");

}

};

obj2.dis1();

A obj3 = new A(){

public void dis1() {

System.out.println("A interface method override by Anonymous class Once again");

}

};

obj3.dis1();

obj2.dis1();

A obj4 = ()->System.out.println("Override by Lambda Expression....");

obj4.dis1();

}

}

interface Operation {

int add(int x, int y);

}

class Demo {

public static void main(String args[]) {

Operation op1 = (x,y)->x+y;

System.out.println(op1.add(10,20));

Operation op2 = (a,b)->a+b;

System.out.println(op2.add(100,200));

Operation op3 = (int x,int y)->x+y;

System.out.println(op3.add(101,202));

Operation op4 = (a,b)->{

int sum = a+b;

return sum;

};

System.out.println(op4.add(111,222));

}

}

**Stream API :**

**Stream : Flow of data.**

Stream API provided lot of pre-defined method which help to load the data on demand as well as data get loaded for temporary purpose.

Collection

Framework

---🡪 Stream -🡪 Intermediate operation1🡪2🡪3… Terminal operator

Primitive

Array

Intermediate operator or operation return type is stream itself and Terminal operator return type may void, int , boolean etc.

Top most four functional interface part of function package.

1. Function

apply() : Take T parameter and return R value.

1. Supplier

No parameter but return T value

1. Consumer

Only take T but no re­­turn type

1. Predicate

Take T and return boolean value

1. Display Sorting
   1. ProductId Asc
   2. ProductId Desc
   3. PName Asc
   4. PName Desc
   5. Price Asc
   6. Price Dec Using Comparator
2. Display all product details based upon the conditions
   1. Name start with character
   2. Name contains some character
   3. Price >
   4. Price < Using Stream API

**Day 21 : 19/11/2020**