OOP

What is OOP - In OOP we design program using classes and object. Object is related to real world entities such as book, house, pencil etc. OOP focuses on writing reusable code.

Object - Object is instance of a class, it is entity that has state(attribute) and behavior. When class is created, we have to create its object to allocate memory.

E.g.: - if dog is an object so it has states like color, age, name etc. and behavior like eating, barking etc.

Class - Class is user defined data type, it is collection of objects. Class is a blueprint of object.

E.g.: - suppose we have a class "Expensive cars" then its objects might be Mercedes, bmw, jaguar etc.
its properties can be price, speed of car and methods are driving, reverse etc.

Method - Method is a collection of statements that performs specific task and returns the result.

Access Specifiers

It defines access type of a method.

Public – accessible throughout the program(in any class).

Private – accessible within that class only.

Protected – accessible within that class and its sub classes.

Polymorphism - In polymorphism one task can be performed in different ways.

E.g :- suppose we have class animal, all animals speak but everyone speak in different way. So here "speak" is polymorphic.

Runtime polymorphism – Method Overriding.

Late Binding/Dynamic Binding – Method Overriding.

```
class Bike{
void run(){
System.out.println("Bike is Running");
}
}

class Polymorphism extends Bike{
void run(){
System.out.println("Splendor is Running....");
}

public static void main(String args[]){
Bike b = new Polymorphism();
b.run();
}
}
```

Compile Time Polymorphism – Method Overloading

Early binding/Static Binding – Method Overloading.

```
class Main{
public int addition(int a, int b){
return a+b;
}

public int addition(int a, int b,int c){
return a+b+c;
}
}

class Meth_Over{
public static void main(String args[]){
Main m = new Main();
System.out.println(m.addition(2,3));
Main m1 = new Main();
System.out.println(m1.addition(2,3,4));
}
}
```

Encapsulation - Binding or wrapping code and data together in a single unit is called encapsulation. In this variables of a class are always hidden from another class. We restrict access to methods and variables. They can only access by methods of that class.

```
class Student{
private String name;
private int rollno;
private int age;

public String getname(){
return name;
}
public int getrollno(){
return rollno;
}
public int getage(){
return age;
}
public void setname(String newname){
name = newname;
}
public void setrollno(int newroll){
rollno = newroll;
}
public void setage(int newage){
age = newage;
}
}
```

Abstraction - Hiding internal details and showing only functionalities is called abstraction. E.g.:- Phone Call, we don't know the inter processing we just need to dial the number and call it. Another e.g. is driving a car, while driving we don't have to be concerned with its internal working we just need to concern about parts like steering, break etc.

```
abstract class Shape{
abstract void draw();
}
//In real scenario, implementation is provided by others i
class Rectangle extends Shape{
void draw(){System.out.println("drawing rectangle");}
}
class Circle1 extends Shape{
void draw(){System.out.println("drawing circle");}
}
//In real scenario, method is called by programmer or us
class TestAbstraction1{
public static void main(String args[]){
Shape s=new Circle1();//In a real scenario, object is provi
s.draw();
}
}
```

Association - It is relationship between two objects.

These are: one to one, one to many, many to one, many to many.

E.g.: - many students can associate with one teacher while one student can associate with many teachers.

Aggregation - Child object can't belong to another parent object. Object has its separate life cycle.

E.g.: - one teacher cannot belong to multiple departments.

Composition - Represents relationship where one object other object as a part of its state. If we delete parent object then all the child objects get delete automatically.

E.g.: - Any house can have several rooms, one room cant be a part of two houses if we delete the house room will also get deleted automatically.

Super – use to call parent class constructer. If both parent and base class has same method name then if we call that method using sub class object it will execute method of a subclass, so to call parent class method we use super keyword.

Friend Function – if a function is defined as friend, then private and protected data of that class can be accessed by that function, by using keyword friend compiler knows that the function is friend function. For accessing data friend function should be declared inside the class body.

```
class class_name
{
   friend data_type function_name(argument/s);
};
```

Inheritance – it is a mechanism in which one class can acquire properties of another class.

Parent Class or Base class

Derived or Child class – who can acquire properties of parent class.

Solution to diamond problem – use default methods and interfaces.

```
interface DemoInterface1
{
public default void display()
{
System.out.println("the display() method of DemoInterface1 invoked");
}
}
interface DemoInterface2
{
public default void display()
{
System.out.println("the display() method of DemoInterface2 invoked");
}
}
public class DemoClass implements DemoInterface1, DemoInterface2
{
public void display()
{
DemoInterface1.super.display();
DemoInterface2.super.display();
}
public static void main(String args[])
{
DemoClass obj = new DemoClass();
obj.display();
}
}
```

Anonymous Function – Function which does not have any name are called as Anonymous function, This functions are declared dynamically at run time.

Include JS file - <script type=”text/javascript” src=”filename”>

Include CSS file = <link rel = “stylesheet” href=”filename”>

Exception – It is a event that disturbs the flow of a program.

Exception Handling – It is mechanism to handle runtime errors such as ClassNotFound, IOException etc.

1. Checked Exceptions – Classes that inherit throwable class, these exceptions are Checked at compile time, e.g- IOException, SQLExceptio.

2. Unchecked Exceptions - Classes that inherit RunTimeException, these exceptions are Checked at run time, e.g- ArithmaticException, ArrayIndexoutofBoundExceptio.

Try block – in try block we write the code in which we think error will occuered, we cant write try block alone it is must fllowed by catch or finally.

Catch block - The "catch" block is used to handle the exception.

Finally - It is executed whether an exception is handled or not.

```
public class JavaExceptionExample{
 public static void main(String args[]){
  try{
    //code that may raise exception
    int data=100/0;
  }catch(ArithmeticException e){System.out.println(e);}
  //rest code of the program
  System.out.println("rest of the code...");   }  }
```

Constructor – constructor has the same name as classname, in this we initialize object, it get called when we create instance of a class. When we call the constructor memory for object get allocate.

- **Default Constructor –** used to provide default values to object like 0, null.
- **Parameterized Constructor**

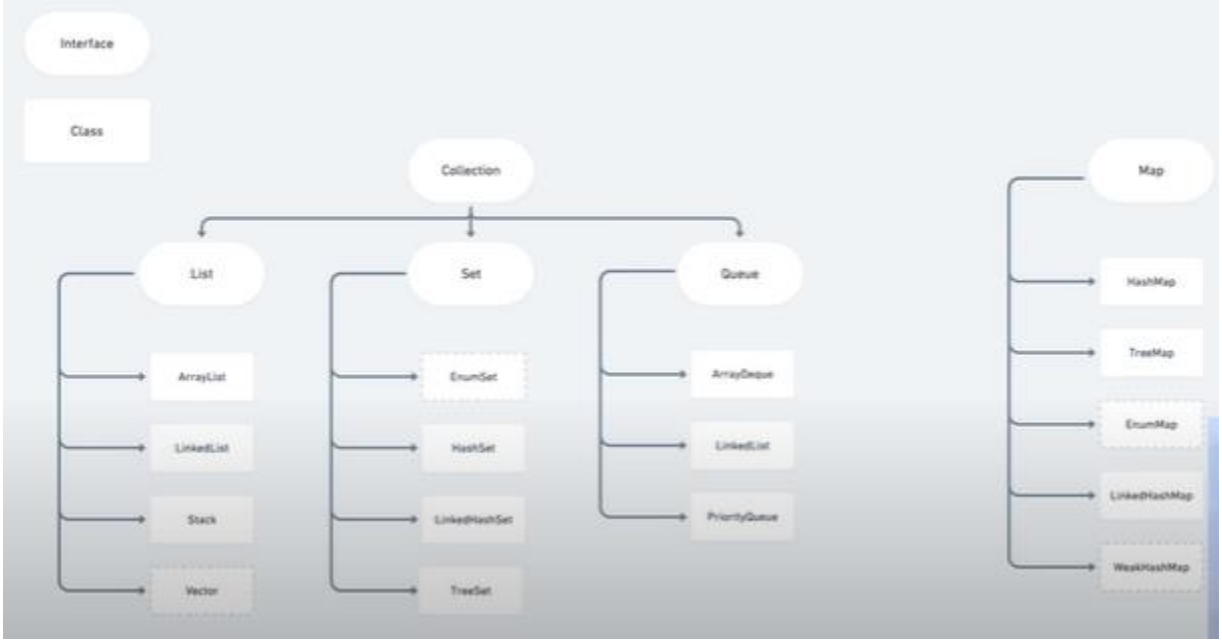
Destructor – when we create object of a class it occupies space in the memory and if we do not delete these it remains in memory, so to free up memory we use destructor.

Garbage Collector - It automatically deletes the unused objects (objects that are no longer used) and free-up the memory. Eg:- **Finalize()** - protected void finalize throws Throwable{}

Structure	**Union**
You can use a struct keyword to define a structure.	You can use a union keyword to define a union.
Every member within structure is assigned a unique memory location.	In union, a memory location is shared by all the data members.
Changing the value of one data member will not affect other data members in structure.	Changing the value of one data member will change the value of other data members in union.
The total size of the structure is the sum of the size of every data member	The total size of the union is the size of the largest data member

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritance**.	Interface **supports multiple inheritance**.
3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.

6) An **abstract class** can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.
7) An **abstract class** can be extended using keyword "extends".	An **interface** can be implemented using keyword "implements".
8) A Java **abstract class** can have class members like private, protected, etc.	Members of a Java interface are public by default.



JS

Arrow Function –

- Arrow functions are introduced in ES6 version. It provides shorter syntax for declaring functions.
- Arrow functions are declared without the function keyword

```
// Traditional Function Expression
var add = function(a,b){
  return a + b;
}

// Arrow Function Expression
var arrowAdd = (a,b) => a + b;
```

Advantages -

- It reduces code size.
- The return statement is optional for a single line function

ES6 FEATURES -

Let keyword, const keyword, arrow functions, Classes, Generators, and iterators

LET Keyword - The variables declared using **let** keyword will be mutable, i.e., the values of the variable can be changed.

CONST Keyword: The variables declared using **the const** keyword are immutable. The value of the variables cannot be changed.

OS

Deadlock – It is a condition where all processes are blocked because each process holding a resource and waiting for another resource to acquire.

Conditions for deadlock

Mutual Exclusion – one or more than one resources are non sharable

Hold and Wait – process holding a resource and waiting for another resource.

No Preemption – resource cant be taken from process unless process releases the resource.

Circular wait – set of processes waiting for each other in circular form.

Fragmentation - Fragmentation is a phenomenon of memory wastage. It reduces the capacity and performance because space is used inefficiently

- **Internal fragmentation**: It is occurred when we deal with the systems that have fixed size allocation units.
- **External fragmentation**: It is occurred when we deal with systems that have variable-size allocation units

Demand paging – It is a method that loads pages into memory on demand. This method is mostly used in virtual memory. In this, a page is only brought into memory when a location on that particular page is referenced during execution

Multithreading - It is a process of executing multiple threads simultaneously. A thread is a lightweight sub-process, the smallest unit of processing

Thread life cycle
New – thread is in this state when we create instance of thread class.
Runnable – Thread is in this state when we invoke start() method.
Running – when thread schedular select it
Non-Runable – thread is alive but not eligible to run
Terminate – thread is terminated or dead.
Thread Example –

```
class Multi extends Thread{
public void run(){
System.out.println("thread is running...");
}
public static void main(String args[]){
Multi t1=new Multi();
t1.start();
 }
}
```

Scheduling – It is a process to improve efficiency by utilizing maximum cpu space in minimum waiting time

Paging – It is memory management technique that allows os to retrieve peocesses from secondary memory to main memory.
Segmentation - It is memory management technique that divides the process into modules and parts odifferent sizes.
Process – process is a program in execution.
Process life cycle
Start – it is initial stage when process is created.
Ready – process is waiting to be assigned to resource
Running – Process is assigned to processor.
Waiting – process moves to this state when it has to wait for a resource
Terminate – once the process finish its execution enters in this state

SQL(https://www.javatpoint.com/sql-syntax)

MYSQL, SQL Server, Oracle MS Access

Constraint – constraints are on column level.

Not Null – ensures column does not have null value.

Default – specifies default value when none is specified.

Unique – ensures that all values in column are unique

DDL Commands – Create, Alter, drop, Truncate.

DML Commands – Insert, Update, Delete, Select.

DCL Commands – Grant, Revoke.

Diff between delete, truncate and drop

- Delete only deletes one specific rows
- Truncate deletes all data inside table but table structure remain
- Drop deletes(drops) the entire table

CREATE

- Create table tablename(column name datatype)
- e.g:- create table student(name varchar(30), rollno int primary key)

SELECT

- Select columnname from tablename
- Select name from student or select * from student.

ALTER – add, delete, modify column in table

- Alter table tablename add columnname datatype.
- Alter table student add marks int. or alter table student drop marks

DROP – deletes the entire table

- Drop table tablename
- Drop table student

TRUNCATE – deletes all data from the table but table structure remains in db.

- Truncate table tablename
- Truncate table student

INSERT

- Insert into tablename (column names) values (data)
- Insert into student (name, rollno, marks) values (“saurabh”,04,95)

UPDATE

- Update tablename set columnname = value where condition
- Update student set name = “viki” where rollno=04

DELETE

- Delete from tablename where condition
- Delete from student where name = “Saurabh”

Describe – Describes the table structure.

DISTINCT – shows distinct values from specified column names

COMMIT – saves the changes permanently in the db which are done In transaction

ROLLBACK – undo transaction and operations which are not saved yet.

INDEXING –

- It is used to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed.
- It is used to locate and access the data in a database table quickly.

- Index structure consist of two parts, 1)search key – contains copy of primary key or candidate key in sorted 2)order, Data reference - contains a set of pointers which holds the address of the disk block where the value of the particular key can be found

CONCURRENCY – It is ability of database to allow multiple users to affect multiple transactions on database at one time.

Concurrency Control in Database Management System is a procedure of managing simultaneous operations without conflicting with each other.

Problems – Dirty Read Problem, Lost Update Problem

VIEW – view is a virtual table based on result set of sql query. It contains rows and columns just like real table, fields in view are from one or more tables in the database.

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

```
CREATE VIEW Brazil AS
SELECT CustomerName, ContactName
FROM Customers
WHERE Country = 'Brazil';
```

INDEX – indexes are likely lookup tables they are used to retrieve data more quickly than normal, users cannot see index they just used to speedup query.

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);
```

```
CREATE INDEX idx_pname
ON Persons (LastName, FirstName);
```

Triggers – Triggers invoked automatically when any special event occurs on database.

```
create trigger [trigger_name]
[before | after]
{insert | update | delete}
on [table_name]
[for each row]
[trigger_body]
```

ACID – Atomicity, Consistency, Isolation, Durability.

Atomicity – Atomicity defines data remains atomic that means if any operation performed on data it should execute completely or not at all.

Consistency – It makes ure that database remains constant before and after transaction.

Isolation - This property ensures that multiple transactions can occur concurrently without leading to the inconsistency, In short, the operation on one database should begin when the operation on the first database gets complete.

Durability - the term durability ensures that the data after the successful execution of the operation becomes permanent in the database

STUDENT TABLE

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	HARSH	DELHI	xxxxxxxxxx	18
2	PRATIK	BIHAR	xxxxxxxxxx	19
3	RIYANKA	SILIGURI	xxxxxxxxxx	20
4	DEEP	RAMNAGAR	xxxxxxxxxx	18
5	SAPTARHI	KOLKATA	xxxxxxxxxx	19
6	DHANRAJ	BARABAJAR	xxxxxxxxxx	20
7	ROHIT	BALURGHAT	xxxxxxxxxx	18
8	NIRAJ	ALIPUR	xxxxxxxxxx	19

STUDENTCOURSE TABLE

COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

INNER JOIN – returns records that are matching in both tables.

SELECT StudentCourse.COURSE_ID, Student.NAME, Student.AGE FROM Student

INNER JOIN StudentCourse

ON Student.ROLL_NO = StudentCourse.ROLL_NO;

COURSE_ID	NAME	Age
1	HARSH	18
2	PRATIK	19
2	RIYANKA	20
3	DEEP	18
1	SAPTARHI	19

LEFT JOIN – all the rows from left table and only matched rows from right table.

```
SELECT Student.NAME, StudentCourse.COURSE_ID
FROM Student
LEFT JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	*NULL*
ROHIT	*NULL*
NIRAJ	*NULL*

RIGHT JOIN – all rows from right tale and only matched rows from left table.

```
SELECT Student.NAME, StudentCourse.COURSE_ID
FROM Student
RIGHT JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
NULL	4
NULL	5
NULL	4

FULL JOIN - FULL JOIN creates the result-set by combining result of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both the tables. The rows for which there is no matching, the result-set will contain NULL values.

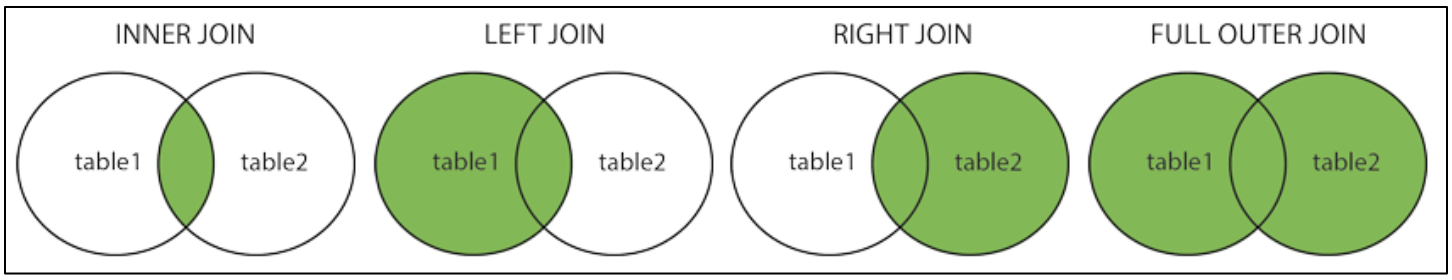
SELECT Student.NAME, StudentCourse.COURSE_ID

FROM Student

FULL JOIN StudentCourse

ON StudentCourse.ROLL_NO = Student.ROLL_NO;

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	*NULL*
ROHIT	*NULL*
NIRAJ	*NULL*
NULL	9
NULL	10
NULL	11



Normalization – it is the process of organizing data in the database. It is used to minimize redundancy from table. Normalization divides larger table into smaller ones and links them using relationship.

1NF – relation is in 1nf if it contains atomic values. It says attribute cant hold multiple values it must contains single value.

2NF - A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.

3NF - A relation will be in 3NF if it is in 2NF and no transition dependency exists.

Indexing – It is a way of optimizing performance of a database by minimizing no of disk accesses required when query is processed.

Primary key – it is a key that uniquely identifies each row in a table.

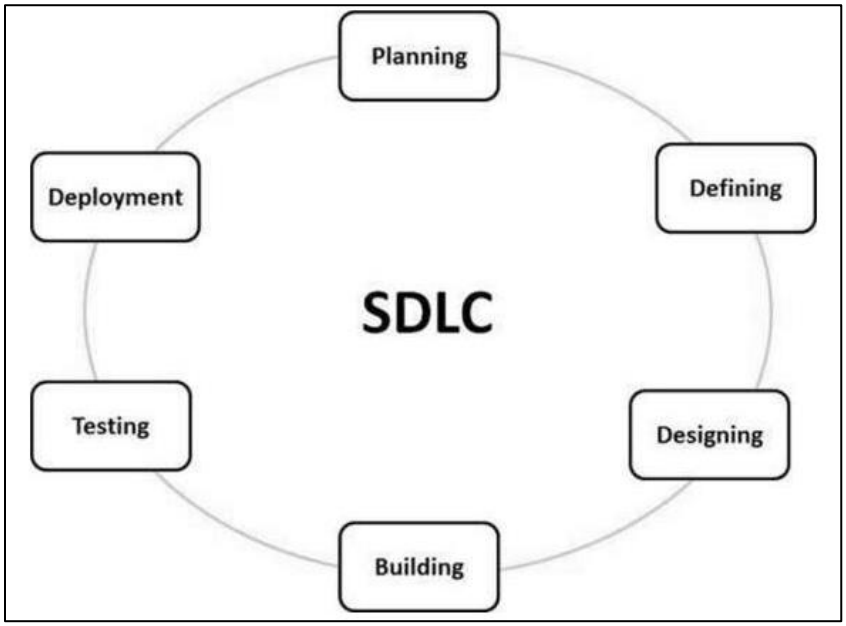
Foreign key - A foreign key is a column or group of columns in a relational database table that provides a link between data in two tables.

Unique Key – Unique key is a constraint that is used to uniquely identify tuple in table.

SDLC & Models

SDLC – Software Development Life Cycle

It is a process used by software industry to design, develop and test high quality softwares.



Planning – it is most imp phase of sdlc, it is performed by the senior members of a team with inputs by customer.

Defining – after planning next step is to define and document product requirements and get it approved by customer.

This is done through **SRS(software requirement specification)** which contains all the requirements of product to be design and developed during SDLC.

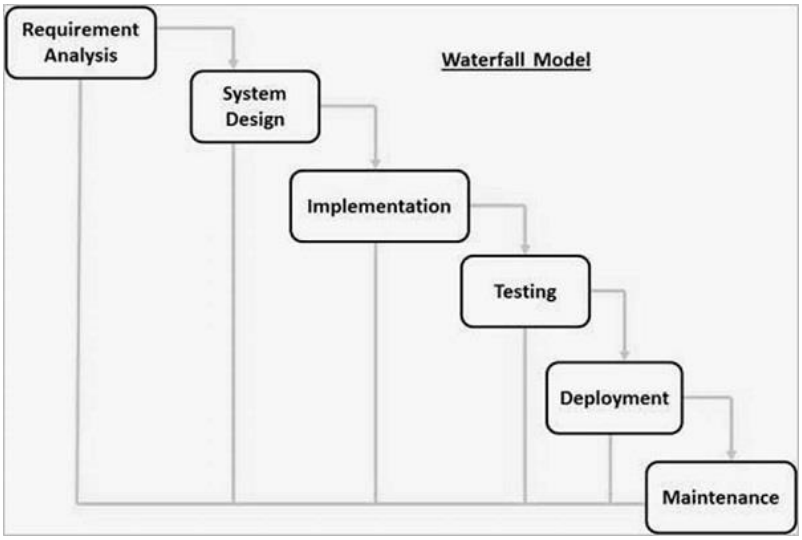
Designing – here we design the layout of a product that means how the product will look after completion etc, inshort we design UI/UX of a product.

Development – Here we actucally code the product, build the product.

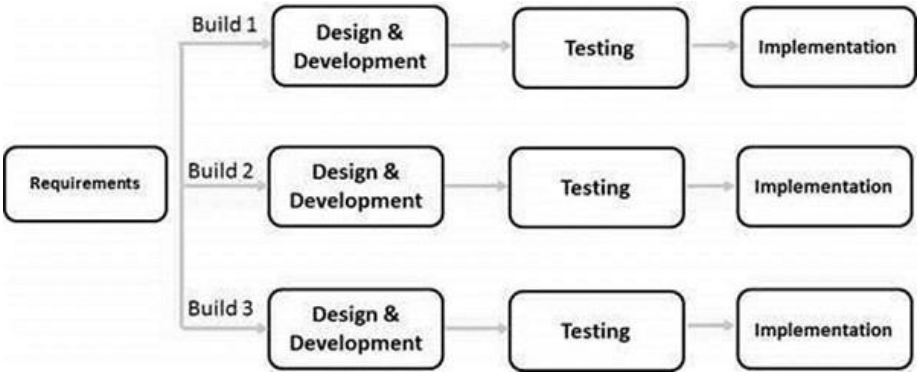
Testing – Here product is tested by testers where bug is reported, tracked, fixed and retested until product meets the quality standard defined in SRS.

Deployment – Once the product is tested and ready to deploy its handed over to customer and released in the market.

Waterfall Model - The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.



Iterative Model - Iterative process starts with a simple implementation of the software requirements and iteratively enhances the versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added.



RAD - The **RAD (Rapid Application Development)** model is based on prototyping and iterative development with no specific planning involved, In the RAD model, the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery

QUERIES

Q)Write an SQL query to fetch "FIRST_NAME" from Worker table using the alias name as <WORKER_NAME>

select FIRST_NAME as WORKER_NAME from worker

Q-2. Write an SQL query to fetch "FIRST_NAME" from Worker table in upper case

select upper(FIRST_NAME) from worker

Q-3. Write an SQL query to fetch unique values of DEPARTMENT from Worker table

select distinct DEPARTMENT from worker

Q-4. Write an SQL query to print the first three characters of FIRST_NAME from Worker table

select distinct DEPARTMENT from worker

Q-6. Write an SQL query to print the FIRST_NAME from Worker table after removing white spaces from the right side

select RTRIM(FIRST_NAME) from worker

Q-7. Write an SQL query to print the DEPARTMENT from Worker table after removing white spaces from the left side

select ltrim(DEPARTMENT) from worker

Q-8. Write an SQL query that fetches the unique values of DEPARTMENT from Worker table and prints its length

select distinct length(DEPARTMENT) from worker

Q-9. Write an SQL query to print the FIRST_NAME from Worker table after replacing 'a' with 'A'

select replace(FIRST_NAME,'a','A') from worker

Q-10. Write an SQL query to print the FIRST_NAME and LAST_NAME from Worker table into a single column COMPLETE_NAME. A space char should separate them

select concat(FIRST_NAME,' ',LAST_NAME) as COMPLETE_NAME from worker

Q-11. Write an SQL query to print all Worker details from the Worker table order by FIRST_NAME Ascending

select * from worker order by FIRST_NAME asc

Q-12. Write an SQL query to print all Worker details from the Worker table order by FIRST_NAME Ascending and DEPARTMENT Descending

SELECT * FROM worker ORDER BY FIRST_NAME ASC, DEPARTMENT DESC

Q-13. Write an SQL query to print details for Workers with the first name as "Vipul" and "Satish" from Worker table

SELECT * FROM WORKER WHERE FIRST_NAME IN('Vipul','Satish')

Q-14. Write an SQL query to print details of workers excluding first names, “Vipul” and “Satish” from Worker table
SELECT * FROM WORKER WHERE FIRST_NAME NOT IN('Vipul','Satish')

Q-15. Write an SQL query to print details of Workers with DEPARTMENT name as “Admin”
SELECT * FROM WORKER WHERE DEPARTMENT = 'Admin'

Q-16. Write an SQL query to print details of the Workers whose FIRST_NAME contains ‘a’
SELECT * FROM WORKER WHERE FIRST_NAME like '%a%'

Q-17. Write an SQL query to print details of the Workers whose FIRST_NAME ends with ‘a’
SELECT * FROM WORKER WHERE FIRST_NAME LIKE '%a'

Q-18. Write an SQL query to print details of the Workers whose FIRST_NAME ends with ‘h’ and contains six alphabets
SELECT * FROM WORKER WHERE FIRST_NAME LIKE '_____h'

Q-19. Write an SQL query to print details of the Workers whose SALARY lies between 100000 and 500000
SELECT * FROM WORKER WHERE SALARY BETWEEN 100000 AND 500000

Q-20. Write an SQL query to print details of the Workers who have joined in Feb’2014
SELECT * FROM WORKER WHERE JOINING_DATE LIKE '2014-02%'

Q-21. Write an SQL query to fetch the count of employees working in the department ‘Admin’
SELECT COUNT(*) FROM WORKER WHERE DEPARTMENT = 'Admin'

Q-22. Write an SQL query to fetch worker names with salaries >= 50000 and <= 100000
SELECT * FROM WORKER WHERE SALARY BETWEEN 50000 AND 100000

Q-23. Write an SQL query to fetch the no. of workers for each department in the descending order
SELECT DEPARTMENT, COUNT(WORKER_ID)NOS FROM WORKER GROUP BY DEPARTMENT ORDER BY NOS DESC

Q-24. Write an SQL query to print details of the Workers who are also Managers
SELECT W.FIRST_NAME, T.WORKER_TITLE FROM WORKER W JOIN TITLE T ON W.WORKER_ID = T.WORKER_REF_ID AND T.WORKER_TITLE = 'Manager'

Q-25. Write an SQL query to fetch duplicate records having matching data in some fields of a table
SELECT WORKER_TITLE, AFFECTED_FROM, COUNT(*) FROM TITLE GROUP BY WORKER_TITLE, AFFECTED_FROM HAVING COUNT(*)>1

Q-26. Write an SQL query to show only odd rows from a table
SELECT * FROM WORKER WHERE MOD (WORKER_ID, 2) <> 0

Q-27. Write an SQL query to show only even rows from a table
SELECT * FROM WORKER WHERE MOD (WORKER_ID, 2) = 0

Q-28. Write an SQL query to clone a new table from another table
SELECT * INTO WORKERCLONE FROM WORKER

Q-29. Write an SQL query to fetch intersecting records of two tables
(SELECT * FROM WORKER) INTERSECT (SELECT * FROM WORKERCLONE)

Q-31. Write an SQL query to show the current date and time
SELECT NOW()

Q-32. Write an SQL query to show the top n (say 5) records of a table
SELECT * FROM WORKER LIMIT 5

Q-33. Write an SQL query to determine the highest salary from a table
SELECT SALARY FROM WORKER ORDER BY SALARY DESC OFFSET 0 LIMIT 1

Q-35. Write an SQL query to fetch the list of employees with the same salary
SELECT DISTINCT W.WORKER_ID, W.FIRST_NAME, W.Salary FROM WORKER W, WORKER W1 WHERE W.SALARY = W1.SALARY AND W.WORKER_ID!=W1.WORKER_ID

Q-36. Write an SQL query to show the second highest salary from a table
SELECT SALARY FROM WORKER ORDER BY SALARY DESC OFFSET 2 LIMIT 1

Q-40. Write an SQL query to fetch the departments that have less than five people in it
SELECT DEPARTMENT , COUNT(WORKER_ID) FROM WORKER GROUP BY DEPARTMENT HAVING COUNT(WORKER_ID) < 5

Q-41. Write an SQL query to show all departments along with the number of people in there
SELECT DEPARTMENT, COUNT(WORKER_ID) FROM WORKER GROUP BY DEPARTMENT

Q-42. Write an SQL query to show the last record from a table
SELECT * FROM WORKER WHERE WORKER_ID = (SELECT MAX(WORKER_ID) FROM WORKER)

Q-43. Write an SQL query to fetch the first row of a table
SELECT * FROM WORKER WHERE WORKER_ID = (SELECT MIN(WORKER_ID) FROM WORKER)

Q-44. Write an SQL query to fetch the last five records from a table
SELECT * FROM WORKER OFFSET 3 LIMIT 5

Q-46. Write an SQL query to fetch three max salaries from a table
SELECT DISTINCT SALARY FROM WORKER ORDER BY SALARY DESC LIMIT 3

Q-47. Write an SQL query to fetch three min salaries from a table
SELECT DISTINCT SALARY FROM WORKER ORDER BY SALARY ASC LIMIT 3

Q-48. Write an SQL query to fetch 4th max salaries from a table
SELECT DISTINCT SALARY FROM WORKER ORDER BY SALARY DESC OFFSET 3 LIMIT 1

Q-49. Write an SQL query to fetch departments along with the total salaries paid for each of them
SELECT DEPARTMENT, SUM(SALARY) FROM WORKER GROUP BY DEPARTMENT

Q-50. Write an SQL query to fetch the names of workers who earn the highest salary
SELECT SALARY, FIRST_NAME FROM WORKER WHERE SALARY=(SELECT MAX(SALARY) FROM WORKER)