Date 20
M T W T F S

Driver Manager:-

Driver Manager Manages the set of java database connectivity drivers that are available for an application to use

Connection:-

A conection is a session b/w the java application and the database. It helps to establish connection with data base.

Java Statements:-

helps to exelutes queries and java.

Primary Key	Foreign Key
A primary key is a unique identifier for each record in a table.	A foreign key establishes a relationship between tables by refering the primary key of an other table.
It ensures the value in the specific column is unique.	A table can have multiple foreign keys each linking to primary key of different tables.
There is only one primary key in table	

Date 20
M T W T F S

What is database?

Database is a structural collection of organized data. in a way that a computer program can quickly select and reterieve data.

Relational database:-

It is a type of database which organize data into tables which contains columns and rows. The data can be related to different other tables through common fields.

Exception Handling:-

It is the methadology that is used to handle abnormalities, In the code that occurs during runtime. These errors cause the rest of code to get terminated and not executed.

The two keywords try and catch are used to handle these abnormalities

Date_____20___
M T W T F S

The code that is likely to generate an error is placed in try block.

While the exception is handled in catch block.

Finally → name of block

Finally block is used to execute code that is important to regardless of result of try block.

Event Handlings:-

It is the mechanism that controls the event and decides what to happen next when event is occured. Event listeney is used for this purpose.

Copy Constructor:- It is the method to call the another constructor or copy the value of one object to the other object.

Syntax:-
```
Student (Student Std1) {
    name = St1.name;
    age = Std1.age;
}
```

Note:- when you create object of class and assign to another object it have the same output.

```
Student one = new Student();
Student two = one;
```

Encapsulation:

Wrapping up the implementation of the data members and methods in class.

- In simple means it can hide the data from outside.

Abstraction:

Abstraction means hiding the unnecessary details and showing valuable information.

→ All the extra information is hiding from us and show the only relevant data.

e.g:- when you have a car and need a key to start the car but don't need what is the mechanism behind the car (start).

e.g:- Array list = new Array();

we use List. method to access all the functions of the array but we don't need to know what are the reasons behind this. (Abstraction).

→ Encapsulation are the hiding the data using access specifiers (Public, private, protected) in the class.

→ we use private keyword to hide the data in the class.

OOP (Java)

Kunal

Detail Notes text file is available in Github repo.

Class:- A class is a name group of properties and functions.

Syntax:- class Car { → name of class.
→ members. (same and diff. data type)
}

→ Name of the class is itself a data type.

Object:- The instance of the class is called object.

→ object occupy space in memory.

→ Using the dot operator (.) to access the member of class.

→ dot operator links reference variable with the instance variable.

Constructors:- Student();

→ Same name as class name.

→ A constructor basically defined what happened when the object is created.

→ set default values for all the instance/object of the class.

→ NO return type, Argument or NO Argument constructor.

→ Constructor is the special type of function in the class that have default values. (already built).

→ NO Argument constructor is by default constructor.

Syntax:- Student() { → Assign by default value.
Argument or NO. }

this keyword:- this keyword is replace the name of the object. Inside the constructor use this keyword like Ali.rollno=13; → so we use this instead of Ali.

```
Student() {
this.rollno=13;
}
```

This keyword is mostly use in the [redacted] Argument constructor to assign the value of argument to member of class and use when we have same variable name.

```
Student(int rollno){
this.rollno= rollno;
}
```

↓ Defined in class

Polymorphism: The word polymorphism split in two words.

poly means many morphism means ways to represent so, polymorphism means many ways to represent single entity.

→ Java support polymorphism.

→ Polymorphism occurs during Inheritence (mostly).

- One function represent with same name in different classes and for each class object it has different output.
- You can also create object using parent and child class.

e.g:- Ali ali = new Student();

→ In this the object is of student class.

Example:

```
class shape {
    Public void area(){
    S.O.P("This is shape class");
    }
class square extends shape {
    Public void area(){
    S.O.P("The area of square is 2x2");
    }
class triangle extends shape {
    Public void display(){
    S.O.P("Triangle l.b.");
    }
Public class Main{
    P.S.V. main (String [] args){
    shapes shapes = new shape();
      shapes.area();
    triangle tri = new triangle();
      tri.area();
}}
```


→ The object select the constructor using the arguments no. is also called type of polymorphism.

Types: Types of polymorphisms are:

① Compile time / static polymorphism.

→ Achieved via method overloading. (Java doesn't have opr. is over loading).

Interfaces:

It is not the class.

→ By default variables are static and methods are final in interfaces.
→ It is the contract in simple language and all classes flow the contract.
→ multiple inheritence support in interfaces.
→ We use keyword implements instead of extends in interfaces.
→ The main shinning point is we use multiple ~~extends~~ implements keyword.
→ One interface implements with another interface but only abstract classes it is not possible.
→ Class support multiple interfaces using Implements keyword but only single inheritence using extends.
→ Full abstraction.

Syntax:

```
interface Brake{
        void brake();
    }
interface Engine {
    void start();
    void stop();
    }
class car implements Brake, Engine {
    // It show error if you don't call the interface
    method in the class.
    void brake (){
      S.O.P(" This is brake");
    }
void start(){
        s.out(" This is start");
    }
```

Scanned with CamScanner


```
    void stop (){
     s.o.p(" This is stop");
    } }
```

→ These are all override methods.
→ Interfaces break the hirearchy of inheritence. When you use multiple interfaces they don't care and point the same inferface.
→ If I access the variable using this.

```
Engine car = new car();
```

(Engine: interface; car: obj name; new car(): constructor.)

```
car.a ; // show error bcz it is not in Engine.
```

→ It create the object of constructor type but show the data of interface type. and a is not present in engine.
→ In performing critical code not prefer the interfaces.

This concept is variable type.

Queries of SQL :-

table name, col', data type, size

Create = Create Table students(rollno int(3), name varchar(20));

Insert = Insert into students (rollno, name) VALUES (1, "Ali");

Update = UPDATE students SET name = "Khaqan" WHERE rollno = 1;

Delete = Delete from students where rollno = 1;

Read All = Select * from students;

Five Steps :-

1. Load the driver
2. Establish the connections
3. Create Statement
4. Execute Query
5. Close the connection.

Date_____20___
M T W T F S

Is a Relationship

Is a relationship is known as inheritance.

Is a relationship refers to inheritance where one class is specialized version of another class

Has a Relationship

Has a relationship is known as a composition.

Has a relationship refers to composition where one class has another class as a member.

Data Member:-

The variables which are declared in any class by using any fundamental data types (like int, char, float etc) or derived data type (like class, structure, pointer etc) are known as data members.