GitHub Link : https://github.com/VIJJU-7/TMF_COURSE_WORK

05/03/24

JDBC :

It is an application programming interface (API) which defines how a client may access a database. It is a part of JavaSE (Java Standard Edition) and is used to connect and execute queries with the database. JDBC API uses JDBC drivers to connect with the database.

There are five ways to connect JDBC :

Import the Packages.

→ **Load the Drivers Using the forName () method :**

\\ Class.forName ("oracle.jdbc.driver.OracleDriver"); \\

→ **Register Driver.**

\\ DriverManager.registerDriver (new oracle.jdbc.driver.OracleDriver ()) \\

→ **Establish a connection Using the Connection class object :**

\\ Connection con = DriverManager.getConnection(url,user,password) \\

user : Username from which your SQL command prompt can be accessed.

password : password from which the SQL command prompt can be accessed.

con : It is a reference to the Connection interface.

Url : Uniform Resource Locator which is created as shown below.

Url method : \\ String url = " jdbc:oracle:thin:@localhost:1521:xe" \\

→ **Create a Statement :**

Once a connection is established you can interact with the database. The JDBCStatement, CallableStatement, and PreparedStatement interfaces define the methods that enable you to send SQL commands and receive data from your database.

\\ Statement st = con.createStatement(); \\

→ **Execute the Query :**

Query for updating / inserting a table in a database :

The executeUpdate(sql query) method of the Statement interface is used to execute queries of updating/inserting.

Query for retrieving data :

The executeQuery() method of the Statement interface is used to execute queries of retrieving values from the database.

Method :

\\ int m = st.executeUpdate(sql);

if (m==1)

System.out.println("inserted successfully : "+sql);

else

System.out.println("insertion failed"); \\

→ Close connection.

\\ con.close (); \\

04/03/24

Super () :

The super keyword in Java is a reference variable that is used to refer to the immediate parent class or a parent object of a subclass. It is used to access the members like methods, variables, and constructors of the parent class within the subclass.

Invoking parent Class Method :

public class Cat extends Animal {

public Cat () {

super (Fname, Lname);

this.getLname ();

this.getFname ();

}

}

Inner classes or Anonymous Inner Class :

It is a class that is declared inside another class or interface.

Ex : class Main {

```
        class Inner {
        public void main () {
        System.out.println("In a nested class method");
        }
    }
}
class Main {
public static void main(String[] args) {
Outer.Inner in = new Outer().new Inner();
}
}
```

Assignment :

Can you create multiple static block statements?

Can we write multiple empty blocks?

If we create an Inner class in the main class how many class files will be created?

Ans : Yes, It compiles and creates two classes in one file.

03/03/24

Practiced Programs

02/03/24

Java Networking :

Java Networking is used for connecting two or more computing devices together so that we can share resources.

Terminologies in Java Networking :

IP Address : A unique number assigned to a node of a network e.g. 192.168.0.1. It is composed of octets that range from 0 to 2551.

Protocol : A set of rules basically that is followed for communication. For example: TCP, FTP, Telnet, SMTP, POP, etc1. POP.

Port Number : Used to uniquely identify different applications. It acts as a communication endpoint between applications.

MAC Address : A unique identifier of NIC (Network Interface Controller). A network node can have multiple NIC but each with a unique MAC address.

Socket : An endpoint between two-way communications.

01/03/24

Learned about synchronization program examples and practiced some programs.Will be provided in Java programs log sheet.

28/02/24

Synchronization :

It is used to control the access of multiple threads to any shared resource. It's used to prevent thread interference and consistency problems1. There are two types of synchronization: Process Synchronization and Thread Synchronization.

Thread Synchronization can be achieved in three ways :

Synchronized Method : If you declare any method as synchronized, it is known as a synchronized method.

Synchronized Block : A synchronized block in Java is synchronized on some object. All synchronized blocks synchronize on the same object and can only have one thread executed inside them at a time.

Static Synchronization : Static synchronization synchronizes on the class's Class object, not on individual objects of that class

Deadlock :

It occurs when a thread is waiting for an object that is acquired by another thread, and the second thread is waiting for an object that is acquired by the first thread.

Daemon thread :

When all the user threads die, JVM terminates this thread automatically, Daemon threads are used for background supporting tasks and They are low priority threads.

29/02/24

Practiced Interview Questions

27/02/24

Practiced Java programs.

26/02/24

Multithreading concepts :

It is a process of simultaneous execution of multiple threads **.**

Assignment :

Using two threads print numbers in range from 1 to 100.

Program : **Main Thread**

```
1
2   public class ThreadExample
3     {
4   public static void main(String[]args) throws InterruptedException
5   {
6
7
8           Thread2 obj1=new Thread2();
9           Thread t2=new Thread(obj1);
10          t2.start();
11          Thread1 obj=new Thread1();
12          Thread t1=new Thread(obj);
13          t1.start();
14
15  }
16  }
```

```java
public class Thread1 extends Thread {
 public void run()
 {
        for(int i=2;i<=100;i++)
        {
                if(i%2==0)
                {
                        System.out.println(i);
                        try {
                                Thread.sleep(100);
                        } catch (InterruptedException e) {

                                e.printStackTrace();
                        }
                }
        }
 }
}
```

Thread1

Thread2

```java
public class Thread2 extends Thread {
 public void run()
 {
        for(int i=0;i<100;i++)
        {
                if(i%2!=0)
                {
                        System.out.println(i);
                        try {
                                Thread.sleep(100);
                        } catch (InterruptedException e) {

                                e.printStackTrace();
                        }
                }
        }
 }
}
```

There are two ways to create a threads in java

1. Extending the thread class.

This class overrides the run() method available in the Thread class. A thread begins its life inside run() method. We create an object of our new class and call start()method to start the execution of a threadStart() invokes the run() method on the Thread object.

Ex : \\ class Multithreading extends Thread

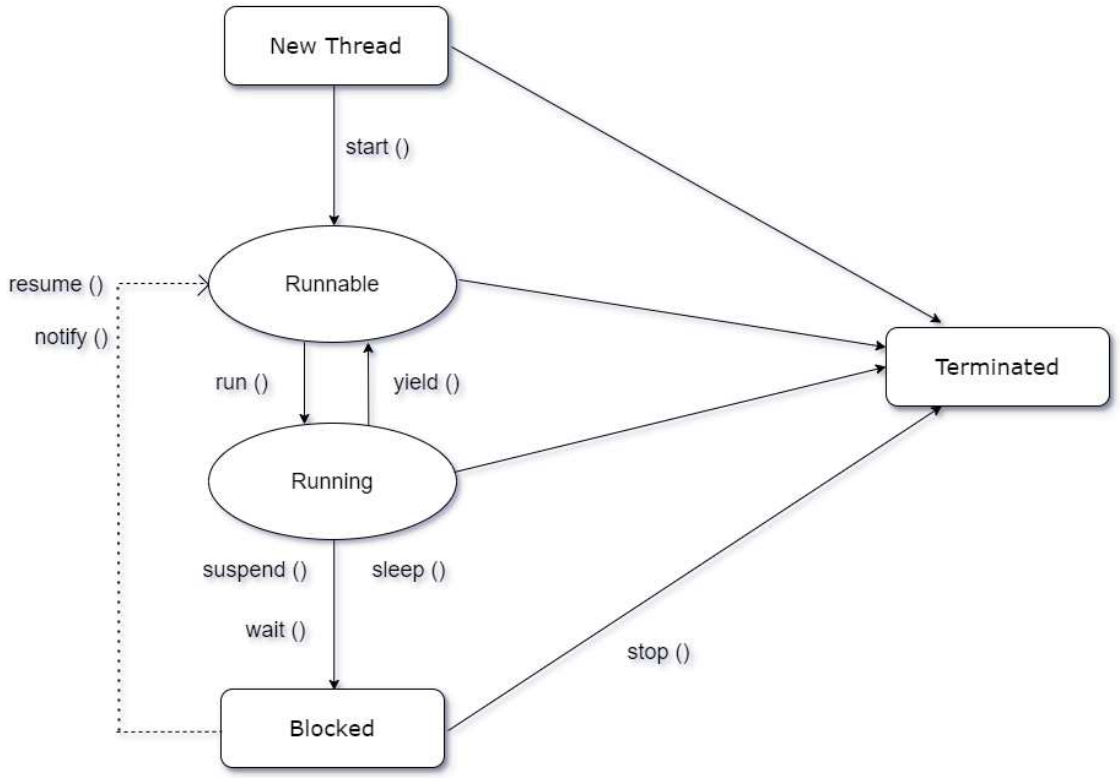
2. Implementing the Runnable Interface.

We create a new class which implements java.lang.Runnable interface and overrides run() method. Then we instantiate a Thread object and call start() method on this object

Ex :\\ class MultithreadingDemo implements Runnable

24/02/24

Life Cycle Of A Thread :



New State:

The thread is in the “new” state when it is created but has not yet started executing.Its code is yet to be run, and it hasn’t started execution.

Runnable State :

A thread that is ready to run is in the “runnable” state. It might actually be running or waiting for CPU time. The thread scheduler gives each thread a chance to run.

Running State :

Running State of a thread where the currently executing processor is said to be in a Running state. It is the responsibility of the thread scheduler to give the thread time to run.

Blocked State :

A thread is in the “blocked” state when it is trying to acquire a lock held by another thread. It moves to the runnable state once it acquires the lock.

Terminated State:

A thread terminates either normally or due to an exceptional event. In the terminated state, the thread no longer runs.

23/02/24

this Keyword :

It refers to the current object within a class.

final Keyword :

It is used to declare a constant value which cannot be changed, It prevents method overriding.

static Keyword :

It allows non-static values but cannot provide any specific instance, It indicates variables or a method belonging to the class.

abstract Keyword :

It cannot be instantiated directly, they are blueprints for subclasses. Used to declare abstract class or method.

22/02/24

Comparable :

It is an Interface which allows objects to compare themselves with other objects of the same type.

Method : \\ compareTo () is used.

Ex : \\ class Student implements Comparable <Student>

\\ public int compareTo(Student st)

Comparator :

It can be used without modifying the actual class and it provides multiple sorting sequences.

Method : \\ compare () is used.

Ex : \\ class AgeComparator implements Comparator<Student>

{

\\ public int compare(Student s1, Student s2)

{

\\ return Integer.compare(s1.age, s2.age);

}

}

21/02/24

Serialization :

It is an Interface and all implemented methods will be done, it is used to convert object state into an order of bytes. And that byte contains all the information about the object. It is used to transfer objects from one JVM (Java Virtual Machine) to another JVM and deserialize in this JVM.

Ex :

\\ Public class Student implements Serializable \\

20/02/24/

Assignment :

Enumeration : It is Java ENUM or Enumeration. It has only a fixed set of constants in words a variable cannot be changed, It also has a rule that all constants should be in capital letters.

Ex :

Class EnumExample

{

Public enum Cars {FORD, BENZ, AUDI}

{

FileReader :

It is used to read data from a text file character by character.It inherits from the InputStreamReader class.

Methods :

read () - reads a single character.

read char (char [] buffer) - reads an array of characters.

close () - closes the reader

FileWriter :

A File Reader is a class which is used to create and write data to a text file. It constructs a FileWriter object.

Methods :

write (int c) - writes a single character.

write (char[] str) - writes an array of characters.

write (String str) - writes a string.

flush ();

close () - flushes the stream first and then closes the writer.

BufferedWriter :

It is a subclass of (java.io.writer) it is used to minimize the write operations or reduce the frequent write operations.

BufferedReader :

it is a class and it is part of the (java.io) package it is used to read text from a character based input stream such as files and buffers the character for efficient reading.

FileInputStream :

It is used for reading streams of characters considered using the FileReader class.

FileOutputStream :

It is an output stream used for writing data to a file, Also used with a FileWriter class.

19/02/24

Collection Framework	Insertion Order	Duplicate Order	Sorted at Insertion	Null Elements	Synchronized
ArrayList	**YES**	**YES**	**NO**	**YES**	**NO**
HashSet	**NO**	**NO**	**NO**	**YES**	**NO**
TreeSet	**NO**	**NO**	**YES**	**NO**	**YES**
LinkedHashSet	**YES**	**YES**	**NO**	**NO**	**NO**
LinkedList	**YES**	**YES**	**NO**	**YES**	**NO**
HashMap	**NO**	**NO**	**NO**	**YES**	**NO**
Hashtable	**NO**	**NO**	**NO**	**NO**	**YES**

13/02/24

Program of Anagram:

```
public class Anagram
     {
public static void main(String[] Args) {
String s1 ="RACE";
String s2= "CARE";
if(s1.length()==s2.length())
       {
char[] ch1 = s1.toCharArray();
char[] ch2 = s2.toCharArray();
Arrays.sort(ch1);
Arrays.sort(ch2);
boolean b = Arrays.equals(ch1,ch2);
if(b)
       {
System.out.println(b + " The Given Strings are Anagram ");
         }

         }
else
```

```
        {
System.out.println(" the given strings length not equal");
        }
    }
}
```

09/02/24

Collection Frameworks :

A collection represents a single unit of objects which can be a group of elements.

Framework includes interfaces and classes

Interfaces – List, Set Map.

Classes – ArrayList, LinkedList, HashSet, LinkedHashSet, TreeSet.

"Java.util "contains all classes and interfaces.

07/02/24

Throw and Throws :

S.no	Throw	Throws
1	It is used to throw an exception explicitly inside a function or the block of code.	Used in the method signature to declare an exception which can be thrown by the function while the execution of a code.
2	We can only see or rectify unchecked exceptions; we cannot rectify checked exceptions using throw only.	In this we can declare both checked and unchecked exceptions.checked exceptions only raised by using throws.
3	Followed by an instance of an exception.	Followed by class names of exceptions to be thrown.
4	Used within a method.	Used within a method signature.

02/02/24

Exception Handling : It is most probably used to handle the runtime errors when an exception occurred or a type of error in the runtime so that flow of execution will be run properly, For this we use Try and Catch block statements.

An exception is an object which is thrown at runtime and it can be resolved by try and catch block statements.Exception handling handles errors such as IOException, NumberformatException, ArithmeticException, NullpointerException…etc.

Scope of variables : It is used in the { ‘’ ‘’ } parentheses which can be accessed with method or class when we declare a variable. There are 4 types of scope variables.

→ **Local** – which are declared inside a method, constructor or code block.

```
public class
{
 public void main(String [ ] args)
 {
  int a = 5; // local variable
  int b =10; // local variable
  int sum = a + b;
  Sysout ( “The sum is : “ + sum )
  }
  }
```

→ **Instance** – It will be within a class but outside the methods, constructors or block.

```
 public class
 {
 double radius; // Instance variable
  public double calculateArea ( )
  {
  return Math.PI * radius * radius;
   }
   }
```

→ **Class or static** – In a class outside a method, constructor or block.

```
public class
{
static double intersetRate; // class or static variable
 }
```

→ **Method parameters** – variables which are passed or sent to a method when it is invoked known as method parameters.

```
 public void printName( String name) // name is a method
{
 Sysout("Hello," + name +"!");
 }
```

01/02/24

Final Keyword : It's a variable which we cannot change the values or we can call it as a constant or fixed value, In other words it cannot be changed by any user.

Final can be –

-> variable.

-> method – we cannot override it.

-> class – We cannot extend it.

Static Members : It is a non-access modifier used for methods and attributes. these can be accessed without creating an object of a class.

We can access static methods in non-static but we cannot access non-static methods in static.

30/01/24

Discussed UML diagrams of Paymentsapp banking.

29/01/24

Abstraction : A class which is declared with the abstract keyword is known as an abstract class. Abstraction is a process of hiding the implementation details and showing only functionality to the user.

There are two ways to achieve abstraction in java

1. Abstract class.
2. Interface.

Final : The Final keyword in java is a constant which cannot be changed after given a value or in other words used to restrict the user. It can be initialized in the constructor only. Final keyword can be:

1. variable
2. method
3. class

Encapsulation : Wrapping of data together into a single unit. It is only used or executed in the designated place.

Java Bean : A JavaBean is a Java class in which all data members are private and a reusable software can hold or encapsulate multiple objects into one object access from multiple positions.

1. It should have a no-arg constructor.
2. It should be Serializable.
3. It should provide methods to set and get the values of the properties, known as getter and setter methods.

25/01/24

Packages : A java package is a group of similar types of classes, interfaces and sub-packages. It can be categorized in two forms, built-in package and user-defined package.

There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

example :

package com.vijay.projectname.

To compile : javac -d . projectname.java

To Run : java com.vijay.projectname

Accessing a package from another package :

1. import package. * ;
2. import package.classname;
3. fully qualified name.

The import keyword is used to make the classes and interface of another package accessible to the current package.

```
package projectname;

import com.vijay. * ;
```

Abstract class : It is a type of class with implemented and unimplemented methods.

Assignment :

```
Public
public class A{
                Public void
                }
                Class B{
                Public static void main(String [ ] args){
                A ob = new A();
                }
                }
Default
```

Default is like public, used in the same class and packages.

Private - private class A

```
{
Private int a=10;
Private void msg(){ }
}
Public class B{
Public Static void main (String [ ] args){
A obj = new a();
obj.msg();  //compile time error
}
}
```

Protect

public class A

```
{
Protected void msg(){ }
}
class B extends A{
Public Static void main (String [ ] args){
B obj = new B();
obj.msg();
}
}
```

Access Modifiers	Within the Class	Outside the class	With in the packages	Outside the package

public	Yes	Yes	Yes	Yes
private	Yes	No	No	No
protect	Yes	Yes	Yes	No
default	Yes	No	Yes	No

24/01/24

IS - A Relationship :

One class is a subtype of another class. It uses “extends” keyword.

```
Class Animal
{
Void sound()
{
System.out.println(“Animal”);
}
}
Class Dog extends Animal{
        Void sound()
{
                System.out.println(“Dog”);
}
}
Class Cat extends Animal{
        Void sound()
{
                System.out.println(“Cat”);
}
}

Public class main
```

```
{
        Public static void main(string[] args)
{
        Animal pet = new dog();
        pet.sound();
        Pet = new cat();
        pet.sound();
}
}
```

HAS - A Relationship :

One class is an instance of another class as one of its numbers.

```
Class Engine
{
        Void start()
{
        System.out.println("Engine is starting…");
}
}
Class Car{
        Private Engine carEngine = new Engine();
        Void startCar(){
        System.out.println("Car is starting…");
        carEngine.start();
}
}
public class Main
{
   public static void main(String[] args)
{
      myCar.startCar();
}
}
```

Program of instanceof:

```
package demo;
```

```
public class Shape
{
	public static void main(String[] args)
{
		System.out.print("Main");
}
	public void shape()
{
		System.out.println("Shape");
}
	public void Triangle()
{
		System.out.println("Triangle");
}
	public void Rectangle()
{
		System.out.println("Rectangle");
}
}
```

```
package demo;

public class Square extends Shape
{
	public static void main(String[] args)
 {
		Square t = (Square) new Shape();
		System.out.println(t instanceof Square);
		System.out.println(t instanceof Shape);
}
}
```

Output : True
True

23/01/24

Polymorphism :

The word "poly" means many and "morphs" means forms. We can perform a single task in multiple ways. There are two types of polymorphism in Java:

1. **Compile-time polymorphism** - It is a static method polymorphism that is resolved during compile time. Overloading of methods is called through the reference variable of a class.
2. **Runtime polymorphism** - Runtime polymorphism or Dynamic Method Dispatch is a process used to call an overridden method that is resolved during runtime.

We can perform polymorphism in java by method overloading and method overriding.

Method overloading : If a class has multiple methods having the same name but different in parameters and different signatures is called method overloading, (Number of parameters and types of parameters).

Ways to overload the method

1. By changing the no.of arguments.
2. By changing the data type.

Method overriding : If a child class has the same method as declared in the parent class is known as Method overriding. It is used for runtime polymorphism.

Rules :

1. The method must have the same name as in the parent class
2. The method must have the same parameter as in the parent class.
3. There must be an IS-A relationship (inheritance).

Type casting :

Type casting is a method or process that converts a data type into another data type in both ways manually and automatically. The automatic conversion is done by the compiler and manual conversion performed by the programmer.

Two types of type casting :

1. Implicit &
2. Explicit Type Casting.

Implicit Type Casting : It converts from low range to high range of data types without any data loss.

example :

```
int a = 5;

long b= a;
```

Explicit Type Casting : It converts high range to low range of data types with data loss.

example :

```
double x = 1.7;

int y =(int) x;
```

Upcasting : Upcasting is a type of object type casting in which a child object is type casted to a parent class object. We can access the variables and methods of the parent class to the child class.We cannot access all the variables and the method in the child class. We can access only some specified variables and methods. Here the parent class references the object to the child class.

Downcasting : Here, the subclass object is referred to by the parent class.

19/01/24

Shortcut for single line comments in eclipse : Ctrl + /

Shortcut for multiline comments in eclipse : Ctrl + Shift + /

Interface:

It is a type of class in which we can implement all methods but cannot create objects, It is used to achieve abstraction. By interface, we can support the functionality of multiple inheritance.

Public class Class name Extends parent Class name implements Interface_name

{

\\ block statement

}

```
        Public Interface Account
{
         Double FindAvgIntrest ();
}
         Double FindAvgIntrest
{
         Return 0.0;
}
```

Interface Rules :

1. Class implements interface 1,2,3
2. Class Extends Class Implements interface 1,2,3
3. Interface extends Interface_name.
4. Class can not be parent of Interface.

Inheritance :

Child object acquires all the properties and behaviors of a parent object or a class. When we inherit from an existing class, we can reuse methods and fields of the parent class.

```
class Vehicle
{
         Color;
}
         Class car Extends vehical
{
```

```
    Car_name;
}
```

Rules in inheritance :

1. Class Extends class.
2. Class implements interface 1,2,3,..
3. Class extends class implements interface 1,2,3…
4. Class can not be a parent for interface.
5. Interface can extend Interface.

18/01/24

Rules for creating multiple classes in one file : public must be mentioned for the main class.

Creating a function

```
Public class test{
Public static void main(String[ ] args){
Int x =0;
Int y=0;
Int k= add(x,y)
}
Int add(int a,int b){
Int c = a+b;
Return c;
}
 Strings[ ] str={ }; // return into the loop.
}
 main(str);
```

Defining an Object in Java :

```
Bankaccount ba = new BankAccount( )
```

From above the class name is bankaccount followed by the object name ba and “new” keyword, followed by the constructor BankAccount( ).

```
ba.accnum=1234;
ba.actype=”Savings”;
Ba.is currentacc= false;
```

```
System.out.println(ba.accnum);
BankAccount ba1 = new BankAccount( ); //new object creation.
```

17/01/24

Write a program to find the given number is prime or not :

```
public class prime{
public static void main(String[] args)
{
        int x = 2;
        int count = 0;
        if(x == 1 || x<0)
{
        System.print.outln("Given number is not a prime number");
}
         for (int i = 2; i<= x-1; i++)
         if(x % i == 0)
{
         count ++;
}
}
         if (count > 0)
{
        System.print.outln("Given number is not a prime number");
}
         else
{
         System.print.outln("Given number is a prime number");
```

```
    }
  }
}
```

Output: Given number is a prime number.

10/01/24

New Keyword : this keyword allocates a memory in dynamic memory location in the derived data types.

Creating a method in java using objects :

```
class  FirstProg
        {
        // Declaration of Variable in int data type.//
        Int i ;
        Student s1.new Student ( );
        Public static void main(string[ ] args) {
        Student s1 = new student( );

        // Giving the value through the index values or size of array.//
        s1.roll no = args[0];
        s1.name = args[1];
        S1.age = args [2];
        }
        // wrapper classes primitive data types.//
        String s1.rollno = Integer.parseint(args[0]);
        Int getstudentAge(){
        }
        Void print studentinfo(Student s1){
        }
}
Class Student {
String roll no;
String name;
String age;
}
```

09/01/24

Initialising a variable : int x = 5;

From the above code we used an type int and gave a variable name and given a value to it, For initialising a variable we need to give a variable name which is already constant and having some value initialised.

Assignment:

Find out the initialised values of primitive data types in java

1. byte : The default value of a byte is 0, (1 byte signed integer min and max value -128 and 127).

2. short : Also 0, (2 byte min and max value : -32,768 and 32,767).

3. int : Also 0, (4 byte min and max value : -2^31 and 2^31 -1), (unsigned integer 32-bit min and max value : 0 and 2^64 -1).

4. long : is ol, (8 byte min and max value : -2^63 and 2^63 -1), (unsigned integer 64-bit min and max value : 0 and 2^64 -1).

5. float : is 0.0f, (it is a 4 byte floating point number)also a single precision 32-bit IEE 754 floating point.

6. double : 0.0, (it is an 8 byte floating-point number) it is also a double precision 64-bit IEEE 754 floating point.

7. char : 2 byte unicode character min value : \u0000 or 0 max value : inclusive 65,535 or \uffff.

8. boolean : Which is said to be true or false and the default value is false,

08/01/24

Structure of Java Program:

In this we have two types of members listed below:

1.Data members

2.Behavioural members

1.Data members : In this we two types of data

→ Primitive data type : In this type we have (Byte, Char, Long , Int, Double, Float, Short, Boolean).

→ Derived data type : In this type we have (String, Array ,Etc).

2.Behavioural members : Behavioural members are a method of a class, Methods are functions that are defined within a class and can be called on objects of that class.

Assignment:

Define a bank account type :

1. Savings
2. Current
3. Deposit
4. Salary Account

04/01/24

Assignment:

Arithmetic Operators using Java:

1. Addition Operator.
2. Subtraction Operator.
3. Multiplication Operator.
4. Division Operator.
5. Modulus Operator.

Program for Addition:

```
import java.io.*;
class Addition {
	public static void main(String[] args)
	{
		int num1 = 10, num2 = 20, sum = 0;
		System.out.println("num1 = " + num1);
		System.out.println("num2 = " + num2);
		sum = num1 + num2;
		System.out.println("The sum = " + sum);
```

```
        }
}
```

Output:

Num1 = 10

Num2 = 20

The sum = 30

Program for Subtraction:

```
import java.io.*;
class Subtraction {
        public static void main(String[] args)
        {
                int num1 = 20, num2 = 10, sub = 0;
                System.out.println("num1 = " + num1);
                System.out.println("num2 = " + num2);
                sub = num1 - num2;
                System.out.println("Subtraction = " + sub);
        }
}
```

Output:

Num1 = 20

Num2 = 10

The sum = 10

Program for Multiplication:

```
import java.io.*;
```

```
class Multiplication {
	public static void main(String[] args)
	{
		int num1 = 20, num2 = 10, mult = 0;
		System.out.println("num1 = " + num1);
		System.out.println("num2 = " + num2);
		mult = num1 * num2;
		System.out.println("Multiplication = " + mult);
	}
}
```

Output:

Num1 = 20

Num2 = 10

The sum = 200

Program for Division:

```
import java.io.*;
class Division {
	public static void main(String[] args)
	{
		int num1 = 20, num2 = 10, div = 0;
		System.out.println("num1 = " + num1);
		System.out.println("num2 = " + num2);
		div = num1 / num2;
		System.out.println("Division = " + div);
	}
}
```

Output:

Num1 = 20

Num2 = 10

The sum = 2

Program for Division:

```
import java.io.*;
class Modulus {
	public static void main(String[] args)
	{
		int num1 = 5, num2 = 2, mod = 0;
		System.out.println("num1 = " + num1);
		System.out.println("num2 = " + num2);
		mod = num1 % num2;
		System.out.println("Remainder = " + mod);
	}
}
```

Output:

Num1 = 5

Num2 = 2

The sum = 1

02/01/24

How to get data from two different tables:

We can get data by using “Joins Operation”.

There are different types of relation which can be useful as shown below,

1. One to One Relation.

2. One to Many Relations.

3. Many to Many Relations.

4. Many to One Relation.

30/12/23

Practicing A Program :

Fibonacci series:

```
class Main {
  public static void main (String [] args) {

    int n = 10, firstTerm = 0, secondTerm = 1;
    System.out.println("Fibonacci Series till " + n + " terms:");

    for (int i = 1; i <= n; ++i) {
      System.out.print(firstTerm + ", ");

      // compute the next term
      int nextTerm = firstTerm + secondTerm;
      firstTerm = secondTerm;
      secondTerm = nextTerm;
    }
  }
}
```

Output:

Fibonacci Series till 10 terms:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34,

29/12/23

Creating Database:

1. Create –

Creates the database.

Create Database Database_Name;

Delete:

Drop Database <Database_Name> ;

Use:

Use <Database_Name>;

Delete the table in the database:

Drop Table<Table_Name>;

Display:

Show Tables;

2.Alter –

Add Columns:

Alter table <Table_Name> ADD data-type;

Delete the Columns:

Alter table <Table_Name> DROP column;

Rename Columns:

Alter table table_name RENAME column old_name to new_name;

Modify the Columns:

Alter table <Table_Name> MODIFY column data-type;

Alter table <Table_Name> ALTER column data-type;

SQL Constraints:

- Not Null, Unique, Primary Key, Foreign Key, Check, Default, Auto Increment.

28/12/23

SQL Query's like Having, Exists, All, Any:

1. Having:

```
Select count (customerid),
Country from Customers
group by country
Having count (customerid);
```

2. Exists:

```
Select *
From Expenses
where exists
(select Expcat_id from exp_categories where Expcat_id = 11);
```

3. ALL, ANY:

ALL - Select ProductName

From products

where productid = All

(Select * From order details where quantity=10);

ANY - Select ProductName

From products

where productid = Any

(Select * From order details where quantity=10);

27/12/23

Assignment:

COUNT ():

Count function returns the number of rows that matches a specific criterion.

Syntax:

select count(column_name) from table_name where condition;

Ex: select count(students) from college where marks>50;

SUM ():

sum () function returns the total sum of the numeric column.

Syntax:

select sum(column_name) from table_name where condition;

Ex: select sum(marks) from Exam;

AVG ():

avg() function returns the average value of the numeric column.

Syntax:

select avg(column_name) from table_name where condition;

Ex: select avg(marks) from Exam;

MAX ():

max () function returns the maximum value in the numeric column.

Syntax:

select max(column_name) from table_name ;

Ex: select max(marks) from Exam;

GROUP BY ():

Group by () statement is used to group rows with the same values into some rows.

Syntax:

SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s) ORDER BY column_name(s);

26/12/23

Date formats:

In this the date formats are shown and how it will store them shown below.

Ex: 1970 - Jan -01 00:10:00 : : 6,00,000 (in millisecond format) 2023 - Dec - 25 00:00:00 : : 1,639,872,000,000(in millisecond format till this Date).

Query:

Select STR_TO_DATE (11 / 25 / 2023, '%m/%d/%Y hh:mm:ss') from dual;

Select * From Expenses Where exp_date >=2023-12-27 and exp_date <= 2023-12-29;

Assignment:

Query:

Select TIMEDIFF ("13:10:00","13:10:10"); O/P - 00:00:01.

Get date () - It prints the current date and time. In the format of YYYY:MM:DD hh:mm:ss.

Solution:

SELECT TIMEDIFF ("13:10:11", "13:10:10");

Result:

Number of Records: 1

TIMEDIFF ("13:10:11", "13:10:10")
00:00:01

The TIMEDIFF () function returns the difference between two time/datetime expressions.

22/12/23

Fetching Records Using Different Queries:

Query's:

1. Concat:

It is used to combine the strings from the columns which is 1 or more by using this query to form one column.

2. Order By:

It is used to arrange the Data by Ascending and descending order in the tables, by using Asc and Desc.

3. Not:

By using this query, we can choose which we would not want any word or character to be shown.

4. Operators:

=, <, >, <=, >=, <>, Between, Like, in.

21/12/23

SQL Query:

1. **Select Query:**
   SELECT * FROM Test_Table;
   SELECT <Column Name>FROM<Table Name>;
2. **Distinct:**

   It is Used to find or fetch only the unique ids or values.
3. **Where clause:**

   To find the records which are in the table.

20/12/23

My SQL:

It is used to handle or manage the RDBMS.

In this it runs some commands like (DDL, DML, Etc.)

In this some stages are used

Stage 1: Design – High- Level Design and Low-Level Design.

Stage 2: Administration – Which will be handled by admins by using few queries which will be used in the database.

Stage 3: Development – This will be used in the backend by using queries like select, use, Delete, Create, etc.

Assignment:

Database which are following tree structure

1. Adjacency List.
2. Path Enumeration.
3. Closure Table.
4. Nested Set.
5. Customize Regular Codes.
6. Solution Comparison.

Relational Database Management System

RDBMS is defined or created in rows and columns which are easy to understand in an efficient manner, some are listed.

1. Oracle Database.
2. Informix.
3. Amazon RDS.
4. Redis.
5. Amazon Aurora.
6. SAP Hana.
7. Teradata.
8. Azure SQL Database.

19/12/23

Difference btw SQL and DBMS:

- SQL: It is used to manipulate and access particular data in an efficient way.
- DBMS: It is used to store, retrieve or organize the data from various sources.

Data Warehouse:

A data warehouse can have a large amount of data from multiple sources which we can expand our data by storing in it.

Assignment:

DBMS Software Systems:

1) Oracle RDBMS.
2) IBM DB2.
3) Microsoft SQL Server.
4) SAP Sybase ASE.
5) Teradata.
6) ADABAS.
7) MySQL.
8) FileMaker.
9) Microsoft Access.
10) Informix.
11) SQLite.
12) Postgres SQL.
13) Amazon RDS.
14) MongoDB.
15) Redis.
16) Couch DB.
17) Neo4j.
18) PHP My Admin.
19) Orient DB.
20) SQL Developer.