AdvJava - 70

Dt : 30/12/2024(day-1)

Summary of CoreJava:

1.Java Programming Components(Java Alphabets)

2.Java Programming Concepts

3.Object Oriented Programming features

1.Java Programming Components(Java Alphabets)

(a)Variables

(b)Methods

(c)Constructors

(d)Blocks

(e)Classes

(f)Interfaces

(g)AbstractClasses

2.Java Programming Concepts

(a)Object Oriented Programming Concept

(b)Exception Handling Process

(c)Multi-Threading Process

(d)Java Collection Framework(JCF)

(Data structure Components are available)

(e)File Storage in Java

(f)Networking in Java

3.Object Oriented Programming features

(a)Class

(b)Object

(c)Encapsulation

(d)Abstraction

(e)PolyMorphism

(f)Inheritance

Note:

=>Using CoreJava Components,Concepts and Construction rules we can developmnet NonServer Applications or Stand-Alone-Applications.

faq:

define Stand-Alone-Applications?

=>The Applications which are installed in one Computer and performs actions in the same Computer are known as Stand-Alone-Applications or NonServer Applications.

Note:

=>AdvJava provide the following technologies to develop Server-based-Applications or Web Applications:

1.JDBC

2.Servlet

3.JSP

1.JDBC:

=>JDBC stands for 'Java DataBase Conectivity' and which is used to interact with database Product.

2.Servlet:

=>Servlet means Server-program,which accepts request from User and provides the response.

3.JSP:

=>JSP stands for 'Java Server Page' and which is response from Web-Application.

faq:

define Server-based-Application?

=>The application which is executed in server environment is known as Server based Application.

Diagram:

Dt : 31/12/2024(day-2)

JDBC(Java DataBase Connectivity):

faq:

define Storage?

=>The memory location where the data is available for accessing is known as Storage.

Types of Storages:

=>According to Java-Application development,the Storages are categorized into four types:

1.Field Storage

2.Object Storage

3.File Storage

4.DataBase Storage

1.Field Storage:

=>The memory generated to hold single-data-value is known as Field Storage.

=>Pimitive datatypes will generate Fileld Storages.

(byte,short,int,long.float,double,boolean,char)

Ex:

int k = 10;

Note:

=>The Field Storages can be in,

=>Class Level ----- Static variable

=>Object Level ----- Instance Variable

=>Method Level ------ Local Variable

**imp*

2.Object Storage:

=>The memory generated to hold group-members is known as Object-storage.

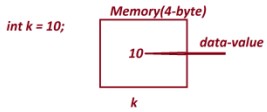
=>NonPrimitive datatype(referential datatypes) will generate Object Storages.

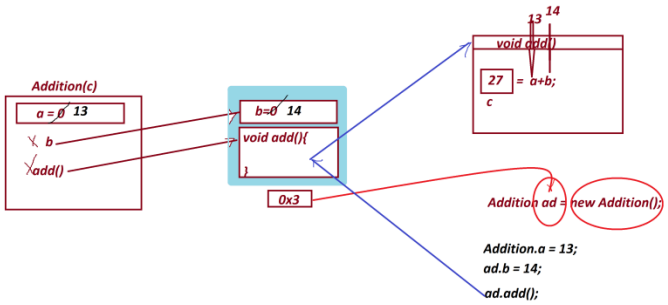
(Class,Interface,Array and Enum)

Ex:

```
class Addition
{
  static int a;
       int b;
  void add()
  {
     int c = a+b;
     Sop(c);
  }
}

Additiob ad = new Addition();

Addition.a=13;
ad.b=14;
ad.add();
```

Diagram:





**imp*

Types of Objects generated from CoreJava:

1.User defined Class Objects

2.String-Objects

3.WrapperClass-Objects

4.Array-Objects

5.Collection<E>-Objects

6.Map<K,V>-Objects

7.Enum<E>-Objects

1.User defined Class Objects

2.String-Objects

(a)String-Class-Objects

(b)StringBuffer-Class-Objects

(c)StringBuilder-Class-Objects

3.WrapperClass-Objects

(a)Byte-Object

(b)Short-Object

(c)Integer-Object

(d)Long-Object

(e)Float-Object

(f)Double-Object

(g)Character-Object

(h)Boolean-Object

4.Array-Objects

(a)Array holding User defined class Objects

(b)Array holding String Objects

(c)Array holding WrapperClass Objects

(d)Array holding Array-Objects(Jagged Array)

(e)Array holding dis-similer objects(Object Array)

5.Collection<E>-Objects:

1.List<E>

(a)ArrayList<E>-Objects

(b)LinkedList<E>-Objects

(c)Vector<E>-Objects

=>Stack<E>-Objects

2.Queue<E>

(a)PriorityQueue<E>-Objects

=>Deque<E>

(b)ArrayDeque<E>-Objects

(c)LinkedList<E>-Objects

3.Set<E>

(a)HashSet<E>-Objects

(b)LinkedHashSet<E>-Objects

(c)TreeSet<E>-Objects

6.Map<K,V>-Objects

(a)HashMap<K,V>-Objects

(b)LinkedHashMap<K,V>-Objects

(c)TreeMap<K,V>-Objects

(d)Hashtable<K,V>-Objects

7.Enum<E>-Objects

---

faq:

wt is the diff b/w

(i)Object

(ii)Object reference

(iii)Object reference Variable

(i)Object:

=>The memory generated to hold instance members of class is known as Object.

(ii)Object reference:

=>The address location where the Object is created is known as Object reference.

(iii)Object reference Variable:

=>The NonPrimitive datatype variable which is holding reference is known as Object

reference variable or Object name.

Dt : 1/1/2024(day-3)

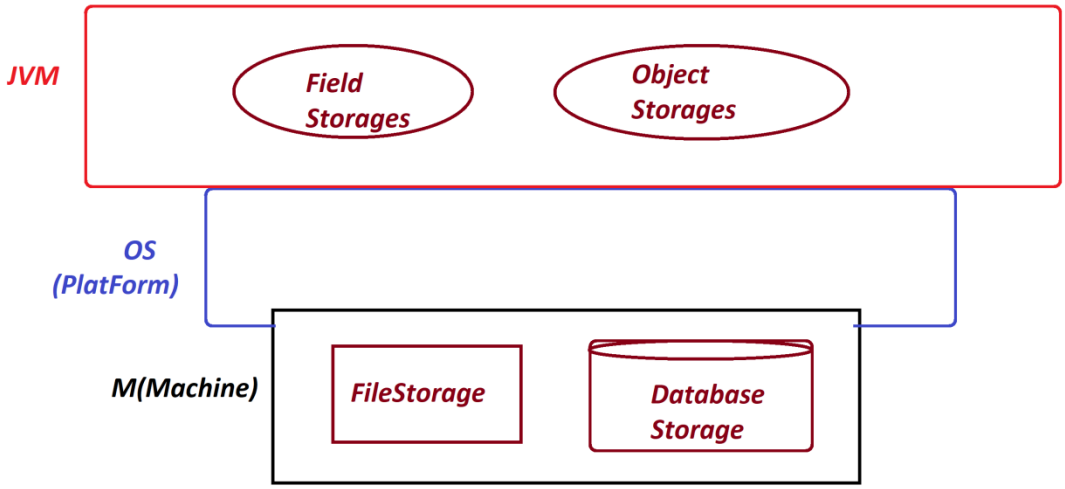
Note:

=>The Field-Storages and the Object-Storages,which are generated part of JVM while application execution will be destroyed automatically when JVM ShutDowns.

=>When we want to have permanent storage for applications,then we take the support of any one of the following:

::File Stotage

::DataBase Storage



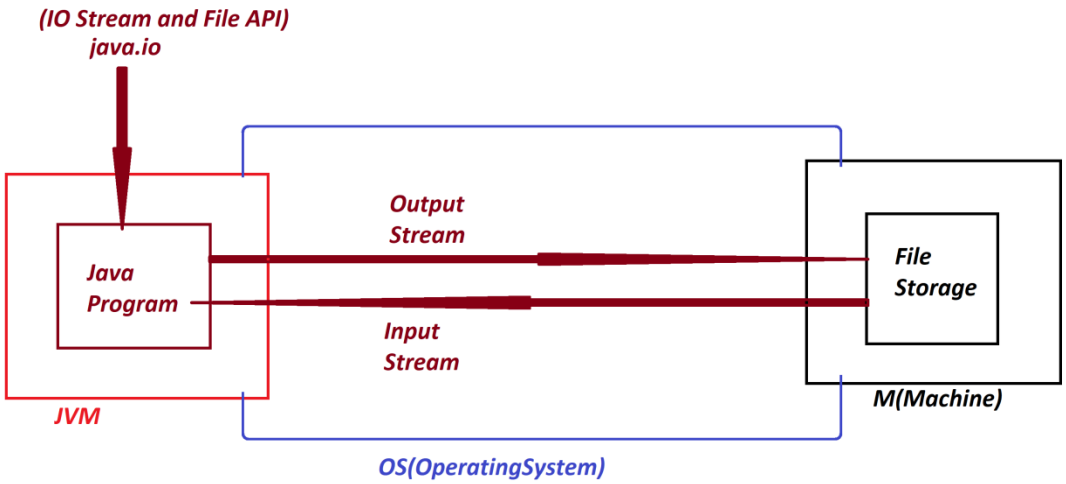
3.File Storage:

=>The smallest permanent storage of ComputerSystem which is 'controlled and managed' by the OperatingSystem is known as File-Storage

=>In the process of establishing communication b/w JavaProgram and FileStorage,the JavaProgram must be constructed using 'Classes and Interfaces' available from 'java.io'

package(IO Stream and File API)

Daigram:



DisAdvantages of File Storage:

(a)Data redundancy

(b)Data Inconsistency

(c)Difficulty in accessing data

(d)Limited data sharing

(e)File System corruption

(f)Security Problems

(a)Data redundancy:

=>Same information will be duplicated in different files.

(data duplication)

(b)Data Inconsistency:

=>data can be inconsistent due to data redundancy

(c)Difficulty in accessing data:

=>Difficulty in accessing data,because the data is available in scattered form and there is no quering process.

(d)Limited data sharing:

=>Limited data sharing because data in scattered form.

(e)File System corruption:

=>File System can be Corrupted due to fragmentation or metadata corruption.

(f)Security Problems:

=>File System will have Security Problems.

Note:

=>Because of DisAdvantages,the file-storage is not preferable as BackEnd Storage for Server-based-Applications in Java.

=>This DisAdvantages of File Storage can be Overcomed using Database Storage.

**imp*

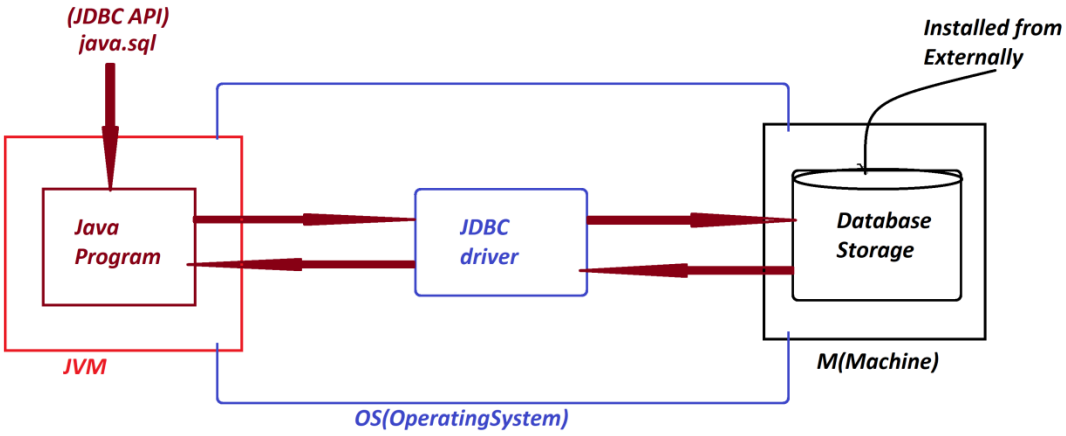
4.DataBase Storage:

=>The largest permanent storage of ComputerSystem which is installed from externally is known an DataBase Storage.

=>In the process of establishing Communication b/w JavaProgram and Database product,the JavaProgram must be Constructed using 'Classes and Interfaces' available from 'java.sql'

package(JDBC API),and the JavaProgram must take the support of JDBC-Driver.

Diagram:



---

faq:

define driver?

=>The Small s/w program part of OperatingSystem,which establishes connection b/w two end-points is known as driver.

Ex:

Audio driver

Video driver

N/W driver

...

---

faq:

define JDBC driver?

=>The driver which establish connection b/w JavaProgram and Database product is known as JDBC driver.

Types of JDBC drivers:

=>JDBC drivers are categorized into four types:

1.JDBC-ODBC bridge driver(Type-1)

2.Native API driver(Type-2)

3.Network protocol driver(Type-3)

4.Thin driver(Type-4)

Note:

=>In realtime for application development we use only Thin driver(Type-4)

Dt : 2/1/2024(day-4)

**imp*

Making ComputerSystem environment ready to execute JDBC Appications:

step-1 : Download and Install Database product(Oracle database)

step-2 : Perform Login Process

step-3 : Create table with name Customer70

(id,name,city,mailid,phno)

primary key : id

create table Customer70(id number(10),name varchar2(15),city varchar2(15),

mailid varchar2(25),phno number(15),primary key(id));

step-4 : Insert min 5 Customer details

insert into Customer70 values(1234,'Alex','Hyd','alex@gmail.com',9898981234);

insert into Customer70 values(2312,'Ram','Hyd','ram@gmail.com',7676761234);

insert into Customer70 values(3212,'Raj','Hyd','raj@gmail.com',4343431234);

step-4 : Copy DB-JAR-File from "lib" folder of Oracle to User defined folder on DeskTop

=>DB-Jar_File is available in the following path:

C:\oraclexe\app\oracle\product\11.2.0\server\jdbc\lib

ojdbc6.jar - Oracle11

faq:

define JAR file?

=>JAR srands for 'Java Archieve' and which is compressed format of more number of class files.

step-5 : Find PortNo and Service Name of Database product(Oracle Product)

=>PortNo and Service Name available from 'tnsnames.ora' file of 'ADMIN" folder of 'network'

C:\oraclexe\app\oracle\product\11.2.0\server\network\ADMIN\tnsnames.ora

Port No : 1521

Service Name : XE

**imp*

JDBC API:

=>'java.sql' package is known as 'JDBC API' and which provide some 'classes and interfaces' to construct JDBC-Applications.

=>'java.sql.Connection' interface is the root of JDBC-API and which is Normal Interface.

=>The following are some important methods of 'Connection' interface:

1.createStatement()

2.prepareStatement()

3.prepareCall()

4.getAutoCommit()

5.setAutoCommit()

6.setSavepoint()

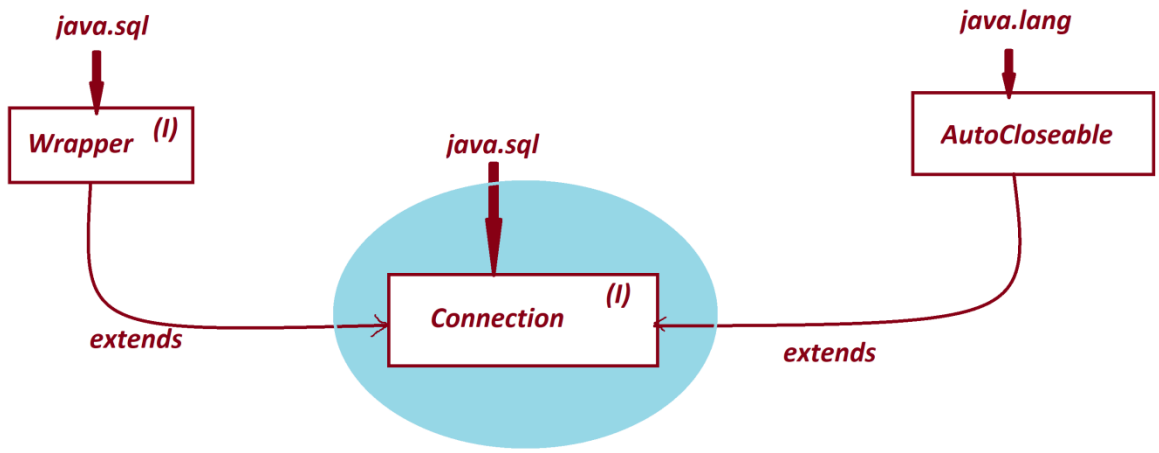
7.releaseSavepoint()

8.commit()

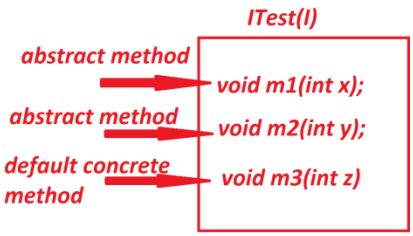
9.rollback()

10.close()

Hierarchy of 'Connection' interface:

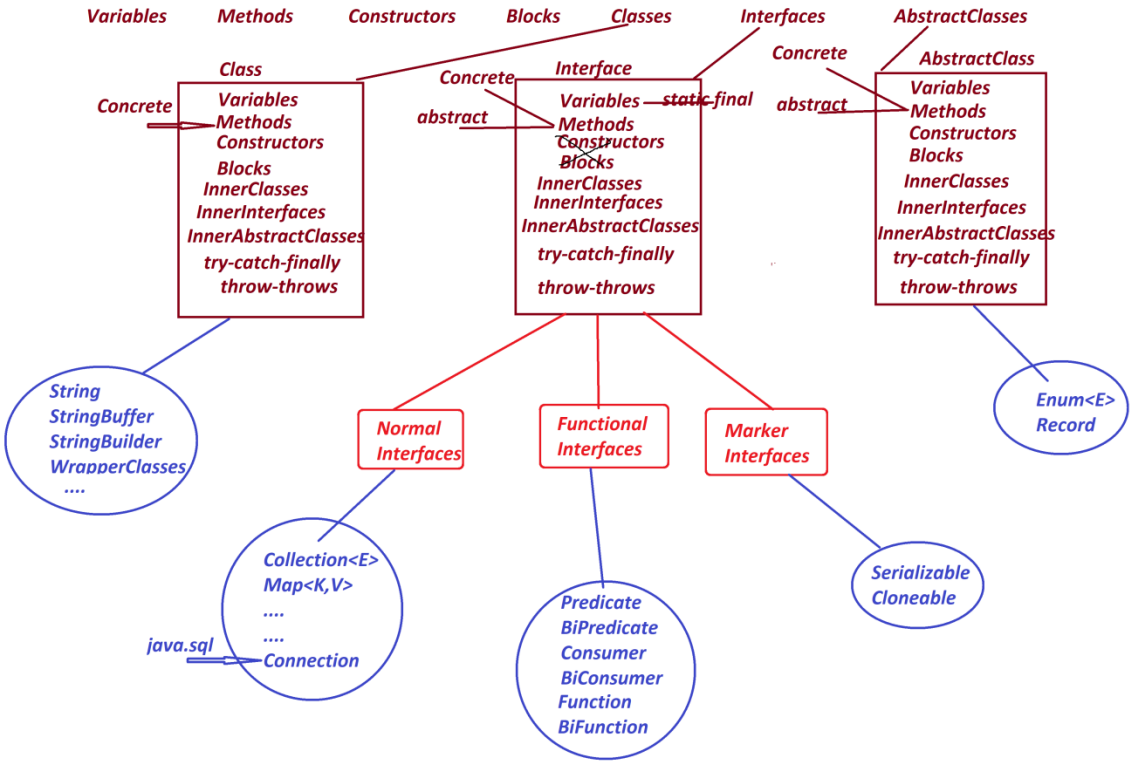


Lab-Assignment



Program-1 : Create implementation object using Implementation class

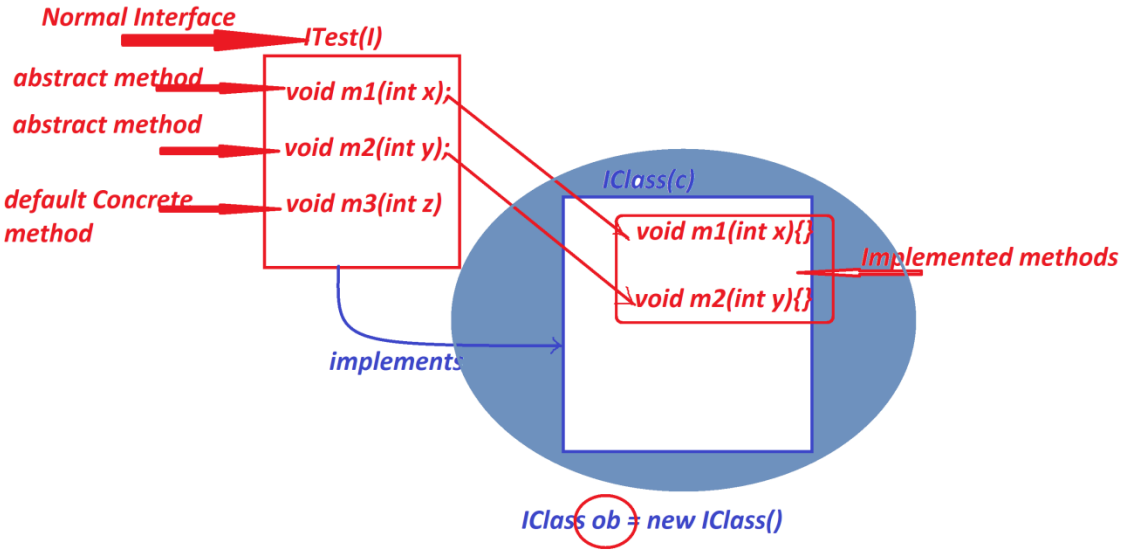
Program-2 : Create implementation Object using 'Anonymous Local Inner as implementation class of Interface'


Variables
Methods
Constructors
Blocks
Classes
Interfaces
AbstractClasses
Class
Concrete
Variables
Methods
Constructors
Blocks
InnerClasses
InnerInterfaces
InnerAbstractClasses
try-catch-finally
throw-throws
Interface
Concrete
abstract
Variables
static final
Methods
Constructors
Blocks
InnerClasses
InnerInterfaces
InnerAbstractClasses
try-catch-finally
throw-throws
Concrete
abstract
AbstractClass
Variables
Methods
Constructors
Blocks
InnerClasses
InnerInterfaces
InnerAbstractClasses
try-catch-finally
throw-throws
String
StringBuffer
StringBuilder
WrapperClasses
....
Normal Interfaces
Functional Interfaces
Marker Interfaces
Enum<E>
Record
Collection<E>
Map<K,V>
....
....
java.sql
Connection
Predicate
BiPredicate
Consumer
BiConsumer
Function
BiFunction
Serializable
Cloneable

Dt : 3/1/2025(Day-5)

Design Pattern from CoreJava:

Diagram:



ProjectName : Design_Model_1_CoreJava

test : ITest.java

```
package test;
public interface ITest
{
   public abstract void m1(int x);
   public abstract void m2(int y);
   public default void m3(int z)
   {
         System.out.println("-----default concrete m3(z)----");
         System.out.println("The value z:"+z);
   }
 }
```

test : IClass.java

```
package test;
public class IClass implements ITest
{
   public void m1(int x)
   {
         System.out.println("-----Implemented method m1(x)-----");
         System.out.println("The value x:"+x);
   }
   public void m2(int y)
   {
         System.out.println("-----Implemented method m2(y)-----");
         System.out.println("The value y:"+y);
   }
}
```

test : DemoModel1.java(MainClass)

```
package test;
import java.util.*;
public class DemoModel1
{
      public static void main(String[] args)
      {
        Scanner s = new Scanner(System.in);//Scanner Object connected to
Console Input
        try(s;){
            System.out.println("Enter the value for x:");
            int x = s.nextInt();
            System.out.println("Enter the value for y:");
            int y = s.nextInt();
            System.out.println("Enter the value for z:");
            int z = s.nextInt();
            IClass ob = new IClass(); //Implementation Object
            ob.m1(x);
            ob.m2(y);
            ob.m3(z);
        }catch(Exception e) {
            e.printStackTrace();
        }
      }
}
```

o/p:

Enter the value for x:

12

Enter the value for y:

21

Enter the value for z:

23

-----Implemented method m1(x)-----

The value x:12

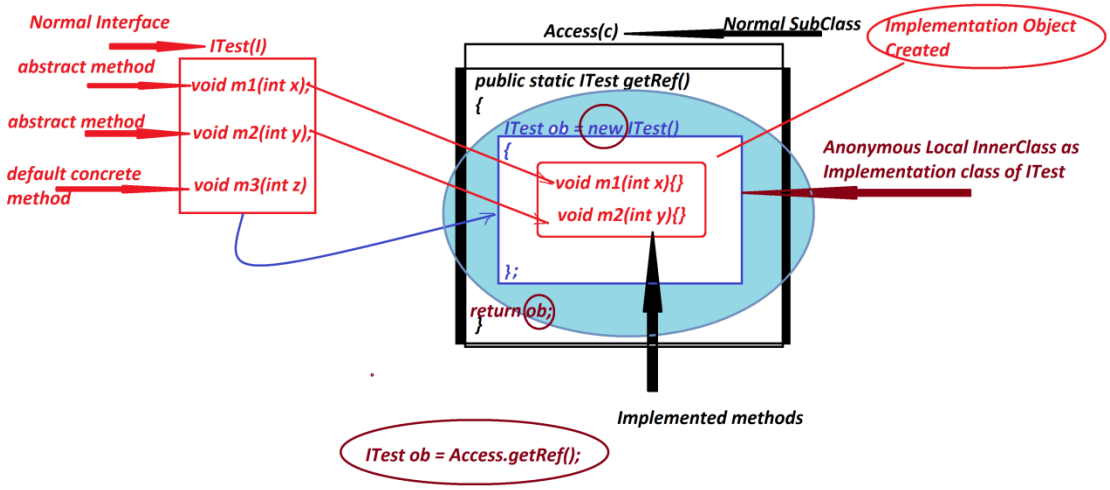
-----Implemented method m2(y)-----

The value y:21

-----default concrete m3(z)----

The value z:23

Diagram:



ProjectName : Design_Model_2_CoreJava

test : ITest.java

```
package test;
```

```
public interface ITest
{
   public abstract void m1(int x);
   public abstract void m2(int y);
   public default void m3(int z)
   {
         System.out.println("-----default concrete m3(z)-----");
         System.out.println("The value z:"+z);
   }
}
```

test : Access.java

```
package test;
public class Access
{
   public static ITest getRef()
   {
         ITest ob = new ITest()
         {
               @Override
               public void m1(int x)
               {
                     System.out.println("-----Implemented method
m1(x)-----");
                     System.out.println("The value x:"+x);
               }
               @Override
               public void m2(int y)
               {
                     System.out.println("-----Implemented method
m2(y)-----");
                     System.out.println("The value y:"+y);
               }
         };
         return ob;
   }//OuterClass static method
}//OuterClass
```

test : DemoModel2.java(MainClass)

```
package test;
import java.util.*;
public class DemoModel2 {
      public static void main(String[] args) {
       Scanner s = new Scanner(System.in);
       try(s;){
         System.out.println("Enter the value for x:");
```

```
            int x = s.nextInt();
            System.out.println("Enter the value for y:");
            int y = s.nextInt();
            System.out.println("Enter the value for z:");
            int z = s.nextInt();

            ITest ob = Access.getRef();//Implementation Object
            ob.m1(x);
            ob.m2(y);
            ob.m3(z);
          }catch(Exception e) {
            e.printStackTrace();
          }
        }
}
```

o/p:

Enter the value for x:

21

Enter the value for y:

22

Enter the value for z:

23

-----Implemented method m1(x)-----

The value x:21

-----Implemented method m2(y)-----

The value y:22

-----default concrete m3(z)-----

The value z:23

Dt : 4/1/2025(Day-6)

=>we use getConnection()-method from 'java.sql.DriverManager' class to create implementation object for 'Connection-Interface'.

=>This getConnection()-method internally holding 'Anonymous Local InnerClass as implementation class of Connection-Interface' and the Connection-Object is generated.

=>This Connection-Object internally connected to Database product.

Method Signature of getConnction():

public static java.sql.Connection getConnection(java.lang.String, java.lang.String, java.lang.String) throws java.sql.SQLException;

syntax:

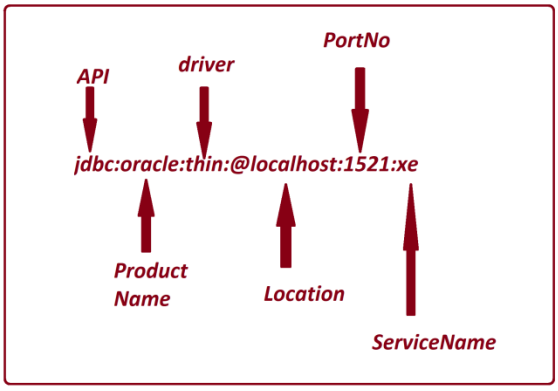
Connection con = DriverManager.getConnction("DB-URL","DB-UName","DB-PWord");

DB-URL => jdbc:oracle:thin:@localhost:1521:xe

DB-UName => system

DB-PWord => tiger

Diagram:



---

*imp

JDBC statements:

=>The statements which specify the actions(operations) to be performed on database product are known as JDBC-statements:

=>These JDBC-statements are categorized into three types:

1.Statement

2.PreparedStatement

3.CallableStatement

1.Statement:

=>'Statement' is an interface from java.sql package and which is used execute Normal queries without IN-Parameters.

(Normal queries means Create,Insert,Select(retrieve),update and delete)

=>we use createStatement()-method from 'Connection-Interface' to create implementation object for 'Statement-Interface'

=>This createStatement()-method internally holding 'Anonymous Local InnerClass as

implementation class of Statement-Interface' and generates Statement-Object.

Method Signature of createStatement():

public abstract java.sql.Statement createStatement() throws java.sql.SQLException;

syntax:

Statement stm = con.createStatement();

=>The following are two important methods of 'Statement':

(a)executeQuery()

(b)executeUpdate()

(a)executeQuery():

=>executeQuery()-method is used to execute select-queries.

Method Signature:

public abstract java.sql.ResultSet executeQuery(java.lang.String)

throws java.sql.SQLException;

syntax:

ResultSet rs = stm.executeQuery("select-query");

(b)executeUpdate():

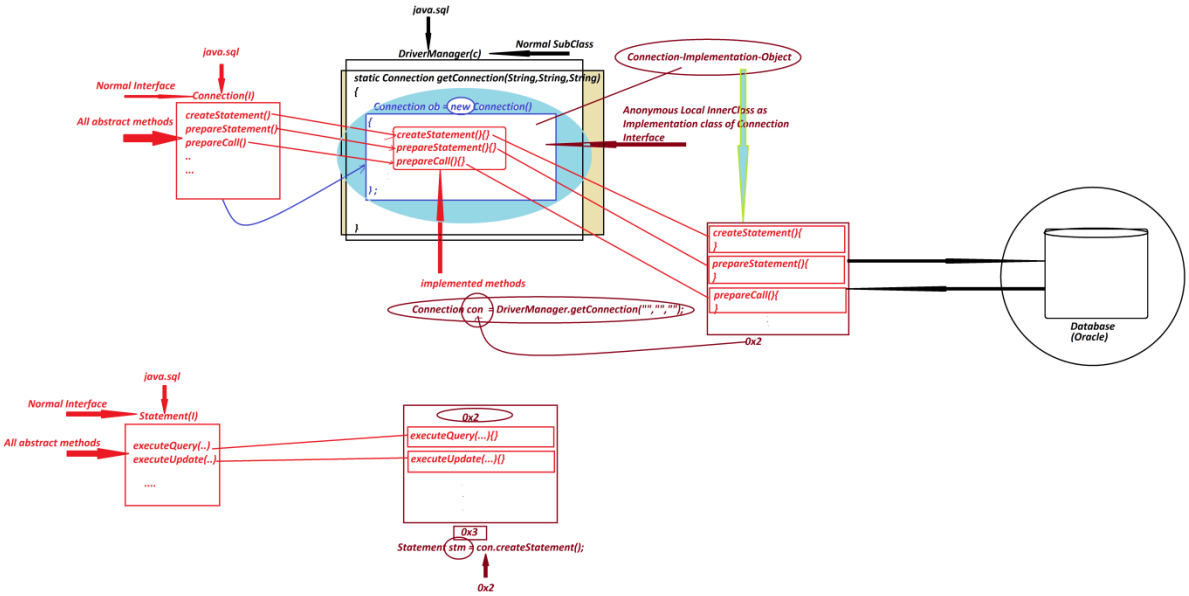
=>executeUpdate()-method is used to execute NonSelect-queries.

Method Signature:

public abstract int executeUpdate(java.lang.String) throws java.sql.SQLException;

syntax:

int k = stm.executeUpdate("NonSelect-query");



Dt : 6/1/2025

=>We use the following steps to perform communication with Database product:

Step-1 : Loading driver

Step-2 : Creating Connection

Step-3 : Creating JDBC-statement

Step-4 : Executing query

Step-5 : Closing the Connection

**imp*

Construct JDBC Application using IDE Eclipse:

step-1 : Open IDE Eclipse,while opening name the WorkSpace and Click 'Launch'

step-2 : Create Java Project

step-3 : ADD DB-Jar-File to Java-Project through 'Build path'

RightClick on JavaProject->Build path->Configure Build Path->Libraries->select 'Classpath'

and click 'Add External JARS'->Browse and select DB-Jar-File->Open->Apply->Apply and Close.

step-4 : Create package in 'src'

step-5 : Create class in package and write JDBC-Code to display all Customer details.

Program : DBCon1.java

```
package test;
import java.sql.*;
public class DBCon1
{
```

```
	public static void main(String[] args)
	{
	 try {
	  // Step-1 : Loading driver
	    Class.forName("oracle.jdbc.driver.OracleDriver");

	    //Step-2 : Creating Connection
	    Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");

	    //  Step-3 : Creating JDBC-statement
	    Statement stm = con.createStatement();

	    //Step-4 : Executing query
	    ResultSet rs = stm.executeQuery("select * from Customer70");
	    System.out.println("-------Customer Details------");
	    while(rs.next())
	    {
	          System.out.println(rs.getInt(1)+"\t"+rs.getString(2)+"\t"+
	         rs.getString(3)+"\t"+rs.getString(4)+"\t"+rs.getLong(5));
	    }//end of loop

	    //  Step-5 : Closing the Connection
	    con.close();
	  }catch(Exception e) {
	    e.printStackTrace();
	  }
	}

}
```

o/p:

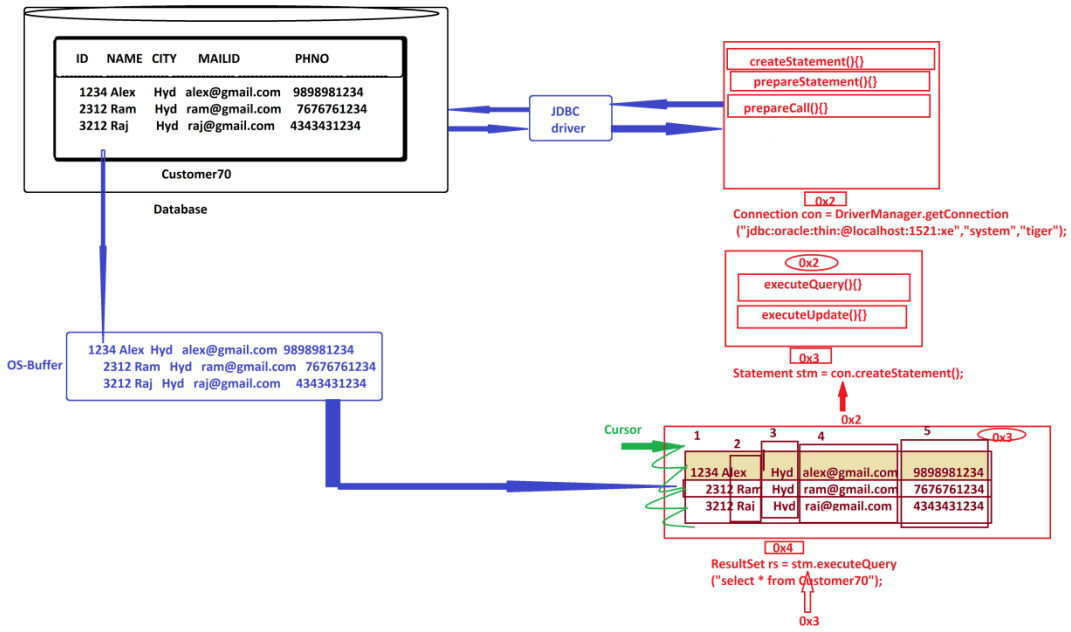
-------Customer Details------

1234 Alex Hyd alex@gmail.com 9898981234

2312 Ram Hyd ram@gmail.com 7676761234

3212 Raj Hyd raj@gmail.com 4343431234

Diagram:



==========

Assignment-1 :

step-1 : Construct DB table with name Product70

(code,name,price,qty)

Primary Key : code

Step-2 : Insert Min 5 Product details....

step-3 : Construct JDBC Application to display all product details.

Dt : 7/1/2025(Day-8)

faq:

wt is the internal execution behaviour of executeQuery() method?

=>This executeQuery()-method internally creates implementation object for ResultSet-Interface,and this ResultSet-Interface-Object will hold the result generated from select-query.

=>This executeQuery()-method also generate one cursor pointing before the first-row.

=>we move the cursor in forward direction using 'next()' method and which is boolean return_type.

syntax:

ResultSet rs = stm.executeQuery("select-query");

Ex-2:(Construct JDBC Application to perform the following Operations on Customer-table)

1.AddCustomer

2.ViewAllCustomers

3.ViewCustomerBasedOnId

Program : DBCon2.java

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon2
{
      public static void main(String[] args)
      {
         Scanner s = new Scanner(System.in);
         try(s;){
             Class.forName("oracle.jdbc.driver.OracleDriver");
             Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
             Statement stm = con.createStatement();
```

```
		while(true) {
			System.out.println("******Operations Choice*****");
			System.out.println("\t1.AddCustomer"
					+ "\n\t2.ViewAllCustomers"
					+ "\n\t3.ViewCustomerById"
					+ "\n\t4.Exit");
			System.out.println("Enter your choice:");
			int choice = Integer.parseInt(s.nextLine());
			switch(choice)
			{
			case 1:
				System.out.println("--------Cutomer
Details---------");
				System.out.println("Enter the Cust-Id:");
				int id = Integer.parseInt(s.nextLine());
				System.out.println("Enter the Cust-Name:");
				String name = s.nextLine();
				System.out.println("Enter the Cust-City:");
				String city = s.nextLine();
				System.out.println("Enter the Cust-MailId:");
				String mId = s.nextLine();
				System.out.println("Enter the Cust-PhNo:");
				long phNo = Long.parseLong(s.nextLine());

				int k = stm.executeUpdate
				("insert into Customer70
values("+id+",'"+name+"','"+city+"','"+mId+"',"+phNo+")");
				if(k>0) {
					System.out.println("Custome Added
Successfully...");
				}
				break;
			case 2:
				ResultSet rs1 = stm.executeQuery("select * from
Customer70");
				System.out.println("------Customer Details-----");
				while(rs1.next()) {
					System.out.println(rs1.getInt(1)+"\t"
							+rs1.getString(2)+"\t"
							+rs1.getString(3)+"\t"
							+rs1.getString(4)+"\t"
							+rs1.getLong(5));
				}//end of loop
				break;
			case 3:
				System.out.println("Enter the Cust-Id to display the
details...");
				int cId = Integer.parseInt(s.nextLine());
				ResultSet rs2 = stm.executeQuery
```

```
                                    ("select * from Customer70 where
id="+cId+"");
                        if(rs2.next()) {
                              System.out.println(rs2.getInt(1)+"\t"
                                          +rs2.getString(2)+"\t"
                                          +rs2.getString(3)+"\t"
                                          +rs2.getString(4)+"\t"
                                          +rs2.getLong(5));
                        }else {
                              System.out.println("Invalid Customer Id....");
                        }
                        break;
                  case 4:
                        System.out.println("Operations Stopped....");
                        System.exit(0);
                  default:
                        System.out.println("Invalid Choice...");
                  }//end of switch
            }//end of loop
          }catch(SQLIntegrityConstraintViolationException sq) {
            System.out.println("Customer details already available...");
          }catch(Exception e) {
            e.printStackTrace();
          }
      }
}
```

o/p:

*******Operations Choice*****

1.AddCustomer

2.ViewAllCustomers

3.ViewCustomerById

4.Exit

Enter your choice:

2

------Customer Details-----

1234 Alex Hyd alex@gmail.com 9898981234

2312 Ram Hyd ram@gmail.com 7676761234

3212 Raj Hyd raj@gmail.com 4343431234

5454 DERE Hyd d@gmail.com 565656123

*******Operations Choice******

1.AddCustomer

2.ViewAllCustomers

3.ViewCustomerById

4.Exit

Enter your choice:

3

Enter the Cust-Id to display the details...

3212

3212 Raj Hyd raj@gmail.com 4343431234

*******Operations Choice******

1.AddCustomer

2.ViewAllCustomers

3.ViewCustomerById

4.Exit

Enter your choice:

2

------Customer Details-----

1234 Alex Hyd alex@gmail.com 9898981234

2312 Ram Hyd ram@gmail.com 7676761234

3212 Raj Hyd raj@gmail.com 4343431234

5454 DERE Hyd d@gmail.com 565656123

*******Operations Choice******

1.AddCustomer

2.ViewAllCustomers

3.ViewCustomerById

4.Exit

Enter your choice:

3

Enter the Cust-Id to display the details...

5412

Invalid Customer Id....

*******Operations Choice*****

1.AddCustomer

2.ViewAllCustomers

3.ViewCustomerById

4.Exit

Enter your choice:

1

--------Cutomer Details---------

Enter the Cust-Id:

6712

Enter the Cust-Name:

TREW

Enter the Cust-City:

Hyd

Enter the Cust-MailId:

c@gmail.com

Enter the Cust-PhNo:

676767123

Custome Added Successfully...

*******Operations Choice*****

1.AddCustomer

2.ViewAllCustomers

3.ViewCustomerById

4.Exit

Enter your choice:

2

------Customer Details-----

1234 Alex Hyd alex@gmail.com 9898981234

2312 Ram Hyd ram@gmail.com 7676761234

3212 Raj Hyd raj@gmail.com 4343431234

5454 DERE Hyd d@gmail.com 565656123

6712 TREW Hyd c@gmail.com 676767123

*******Operations Choice*****

1.AddCustomer

2.ViewAllCustomers

3.ViewCustomerById

4.Exit

Enter your choice:

4

Operations Stopped....

Assignment-2:

Construct JDBC Application to perform the following operations on Product table

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

Assignment-3:

step-1 : Create table with name Employee70

(eid,ename,edesg,bsal,hra,da,totsal)

primary key : eid

step-2 : Construct JDBC Application to perform the following operations on Employee table

1.AddEmployee

2.ViewAllEmployees

3.ViewEmployeeById

hra = 96% of bSal

da = 61% of bsal

totSal = bSal+hra+da;

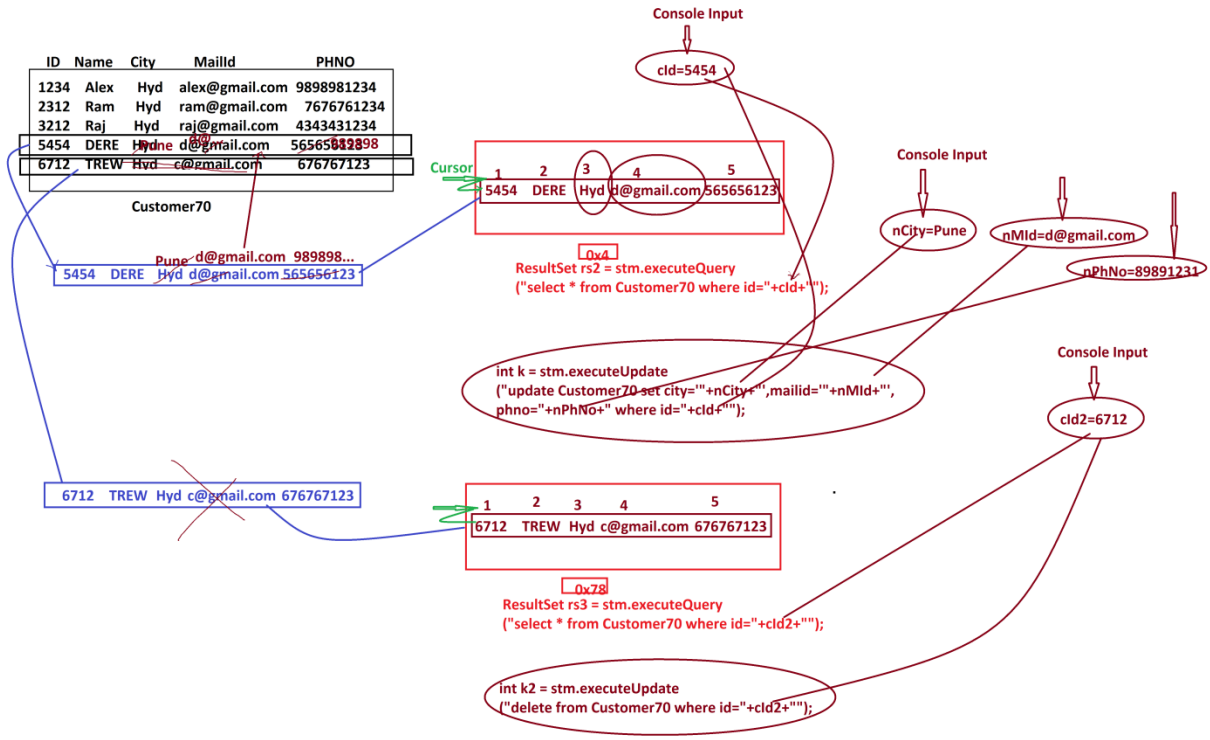
Console Input : eId,eName,eDesg,bSal

Exception Condition : raise exception when bSal is less than 12000.

==========

Dt : 8/1/2025(Day-9)

Ex:(Demonstrating Update and Delete Operations using 'Statement');



Program : DBCon3.java

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon3
{
	public static void main(String[] args)
	{
	   Scanner s = new Scanner(System.in);
	   try(s;){
	       Class.forName("oracle.jdbc.driver.OracleDriver");
	       Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
	       Statement stm = con.createStatement();
	       ResultSet rs = stm.executeQuery("select * from Customer70");
	       System.out.println("-----Customer Details------");
	       while(rs.next())
	       {
	             System.out.println(rs.getInt(1)+"\t"
	                         +rs.getString(2)+"\t"
```

```
                              +rs.getString(3)+"\t"
                              +rs.getString(4)+"\t"
                              +rs.getLong(5));
                }//end of loop
                System.out.println("-----Operations Choice-----");
                System.out.println("\t1.UpdateCustomerById"
                           + "\n\t2.DeleteCustomerById");
                System.out.println("Enter your Choice:");
                int choice = Integer.parseInt(s.nextLine());
                switch(choice)
                {
                case 1:
                      System.out.println("Enter the Cust-Id to perform Update
Operation:");
                      int cId = Integer.parseInt(s.nextLine());
                      ResultSet rs2 =
                                  stm.executeQuery("select * from Customer70
where id="+cId+"");
                      if(rs2.next()) {
                            System.out.println("Existing
City:"+rs2.getString(3));
                            System.out.println("Enter New City:");
                            String nCity = s.nextLine();
                            System.out.println("Existing
MailId:"+rs2.getString(4));
                            System.out.println("Enter the New MailId:");
                            String nMId = s.nextLine();
                            System.out.println("Existing
PhoneNo:"+rs2.getLong(5));
                            System.out.println("Enter the new PhoneNo:");
                            long nPhNo = Long.parseLong(s.nextLine());

                            int k = stm.executeUpdate
                           ("update Customer70 set
city='"+nCity+"',mailid='"+nMId+"',phno="+nPhNo+" where id="+cId+"");
                            if(k>0) {
                                  System.out.println("Customer details Updated
Successfully....");
                            }
                      }else {
                            System.out.println("Invalid Cust-Id...");
                      }
                      break;
                case 2:
                      System.out.println("Enter the Cust-Id to perform Delete
Operation:");
                      int cId2 = Integer.parseInt(s.nextLine());
                      ResultSet rs3 = stm.executeQuery
                                  ("select * from Customer70 where
id="+cId2+"");
```

```
                    if(rs3.next()) {
                          int k2 = stm.executeUpdate
                                      ("delete from Customer70 where
id="+cId2+"");
                          if(k2>0) {
                                System.out.println("Customer details deleted
Successfully....");
                          }
                    }else {
                          System.out.println("Invalid Customer id....");
                    }
                    break;
              default:
                    System.out.println("Invalid Choice....");
              }//end of switch
          }catch(Exception e) {
              e.printStackTrace();
          }
      }
}
```

o/p:(Update Operation)

-----Customer Details------

1234 Alex Hyd alex@gmail.com 9898981234

2312 Ram Hyd ram@gmail.com 7676761234

3212 Raj Hyd raj@gmail.com 4343431234

5454 DERE Hyd d@gmail.com 565656123

6712 TREW Hyd c@gmail.com 676767123

-----Operations Choice-----

1.UpdateCustomerById

2.DeleteCustomerById

Enter your Choice:

1

Enter the Cust-Id to perform Update Operation:

5454

Existing City:Hyd

Enter New City:

Pune

Existing MailId:d@gmail.com

Enter the New MailId:

d@gmail.com

Existing PhoneNo:565656123

Enter the new PhoneNo:

9898983213

Customer details Updated Successfully....

o/p:(Delete Operation)

-----Customer Details------

1234 Alex Hyd alex@gmail.com 9898981234

2312 Ram Hyd ram@gmail.com 7676761234

3212 Raj Hyd raj@gmail.com 4343431234

5454 DERE Pune d@gmail.com 9898983213

6712 TREW Hyd c@gmail.com 676767123

-----Operations Choice-----

1.UpdateCustomerById

2.DeleteCustomerById

Enter your Choice:

2

Enter the Cust-Id to perform Delete Operation:

6712

Customer details deleted Successfully....

==========

Assignment:(Update above applications)

Assignment-2:

Construct JDBC Application to perform the following operations on Product table

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(price-qty)

5.DeleteProductByCode

Assignment-3:

step-1 : Create table with name Employee70

(eid,ename,edesg,bsal,hra,da,totsal)

primary key : eid

step-2 : Construct JDBC Application to perform the following operations on Employee table

1.AddEmployee

2.ViewAllEmployees

3.ViewEmployeeById

4.UpdateEmployeeById(bSal,hra,da,totSal)

5.DeleteEmployeeById

hra = 96% of bSal

da = 61% of bsal

totSal = bSal+hra+da;

Console Input : eId,eName,eDesg,bSal

Exception Condition : raise exception when bSal is less than 12000.

****

Dt : 9/1/2025(Day-10)

**imp*

2.PreparedStatement:

=>'PreparedStatement' is an interface from java.sql package and which is used to execute normal queries with IN-Parameters.

=>we use prepareStatement()-method from Connection-Interface is used to create implementation object for 'PreparedStatement' Interface,because the prepareStatement()-method internally holding 'Anonymous Local InnerClass as implementation class of PreparedStatement Interface'.

Method Signature of prepareStatement():

public abstract java.sql.PreparedStatement prepareStatement(java.lang.String) throws java.sql.SQLException;

syntax:

PreparedStatement ps = con.prepareStatement("query-structure");

=>The following are two important methods of PreparedStatement:

(i)executeQuery()

(ii)executeUpdate()

(i)executeQuery():

=>executeQuery()-method is used to execute select-queries.

Method Signature:

public abstract java.sql.ResultSet executeQuery() throws java.sql.SQLException;

syntax:

ResultSet rs = ps.executeQuery();

(ii)executeUpdate():

=>executeUpdate()-method is used to execute NonSelect-queries.

Method Signature:

public abstract int executeUpdate() throws java.sql.SQLException;

syntax:

int k = ps.executeUpdate();

Assignment-2:(Solution with PreparedStatement)

Construct JDBC Application to perform the following operations on Product table

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(price-qty)

5.DeleteProductByCode

Construct DB table with name Product70

(code,name,price,qty)

Primary Key : code

```
create table Product70(code varchar2(10),name varchar2(15),price number(10,2),
qty number(10),primary key(code));
```

Dt : 10/1/2025

Program : DBCon4.java

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon4
{
        public static void main(String[] args)
        {
    Scanner s = new Scanner(System.in);
    try(s;){
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection
                        ("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
        PreparedStatement ps1 = con.prepareStatement
                        ("insert into Product70 values(?,?,?,?)");//Compilation Completed
        PreparedStatement ps2 = con.prepareStatement
                        ("select * from Product70");//Compilation Completed
```

```
PreparedStatement ps3 = con.prepareStatement
                ("select * from Product70 where code=?");//Compilation Completed
PreparedStatement ps4 = con.prepareStatement
                ("update Product70 set price=?,qty=qty+? where code=?");
                 ////Compilation Completed
PreparedStatement ps5 = con.prepareStatement
                ("delete from Product70 where code=?");
while(true) {
        System.out.println("*******Operations Choice******");
        System.out.println("\t1.AddProduct"
                        + "\n\t2.ViewAllProducts"
                        + "\n\t3.ViewProductByCode"
                        + "\n\t4.UpdateProductByCode(Price-Qty)"
                        + "\n\t5.DeleteProductByCode"
                        + "\n\t6.Exit");
        System.out.println("Enter your Choice:");
        int choice = Integer.parseInt(s.nextLine());
        switch(choice)
        {
        case 1:
                 //data loaded to Local variables of main()-method
                System.out.println("======Product Details=====");
                System.out.println("Enter the Prod-Code:");
                String code = s.nextLine();
                System.out.println("Enter the Prod-Name:");
```

```
String name = s.nextLine();
System.out.println("Enter the Prod-Price:");
float price  = Float.parseFloat(s.nextLine());
System.out.println("Enter the Prod-Qty:");
int qty = Integer.parseInt(s.nextLine());

 //Loading data to PreparedStatement Object from Local variables
ps1.setString(1, code);
ps1.setString(2, name);
ps1.setFloat(3, price);
ps1.setInt(4, qty);

int k = ps1.executeUpdate();
if(k>0) {
        System.out.println("Product added Successfully....");
}
break;
case 2:
ResultSet rs1 = ps2.executeQuery();
System.out.println("------Product Details-----");
while(rs1.next())
{
        System.out.println(rs1.getString(1)+"\t"
                        +rs1.getString(2)+"\t"
                        +rs1.getFloat(3)+"\t"
```

```
                    +rs1.getInt(4));
            }//end of loop
            break;
case 3:
            System.out.println("Enter the Prod-Code to retrieve details:");
            String pc1 = s.nextLine();
            ps3.setString(1, pc1);
            ResultSet rs2 = ps3.executeQuery();
            if(rs2.next()) {
                  System.out.println(rs2.getString(1)+"\t"
                              +rs2.getString(2)+"\t"
                              +rs2.getFloat(3)+"\t"
                              +rs2.getInt(4));
            }else {
                  System.out.println("Invalid Product code....");
            }
            break;
case 4:
            System.out.println("Enter the Prod-Code to update Product details:");
            String pc2 = s.nextLine();
            ps3.setString(1, pc2);
            ResultSet rs3 = ps3.executeQuery();
            if(rs3.next()) {
                  System.out.println("Existing product price:"+rs3.getFloat(3));
                  System.out.println("Enter the new price:");
```

```
                float nPrice = Float.parseFloat(s.nextLine());
                System.out.println("Existing product qty:"+rs3.getInt(4));
                System.out.println("Enter the new qty:");
                int nQty = Integer.parseInt(s.nextLine());
                ps4.setFloat(1,nPrice);
                ps4.setInt(2, nQty);
                ps4.setString(3, pc2);
                int k1 = ps4.executeUpdate();
                if(k1>0) {
                        System.out.println("Product Updated Successfully....");
                }
        }else {
                System.out.println("Invalid Prod-Code....");
        }
        break;
case 5:
        System.out.println("Enter the Prod-Code to delete product details:");
        String pc3 = s.nextLine();
        ps3.setString(1, pc3);
        ResultSet rs4 = ps3.executeQuery();
        if(rs4.next()) {
                ps5.setString(1, pc3);
                int k3 = ps5.executeUpdate();
                if(k3>0) {
                        System.out.println("Product details deleted Successfully..");
```

```
                    }
                }else {
                    System.out.println("Invalid Prod-Code...");
                }
                break;
            case 6:
                System.out.println("Operations Stopped...");
                System.exit(0);
            default:
                System.out.println("Invalid Choice..");
            }//end of switch
        }//end of loop
    }catch(Exception e) {
        e.printStackTrace();
    }
        }
}
```

o/p:

```
*******Operations Choice******
        1.AddProduct
        2.ViewAllProducts
        3.ViewProductByCode
        4.UpdateProductByCode(Price-Qty)
        5.DeleteProductByCode
        6.Exit
```

Enter your Choice:

2

------Product Details-----

A21 Mous 1200.0 12

C12 CDR 700.0 10

********Operations Choice*******

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(Price-Qty)

5.DeleteProductByCode

6.Exit

Enter your Choice:

5

Enter the Prod-Code to delete product details:

C12

Product details deleted Successfully..

********Operations Choice*******

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(Price-Qty)

5.DeleteProductByCode

6.Exit

Enter your Choice:

2

------Product Details-----

A21 Mous 1200.0 12

********Operations Choice*******

1.AddProduct

2.ViewAllProducts

3.ViewProductByCode

4.UpdateProductByCode(Price-Qty)

5.DeleteProductByCode

6.Exit

Enter your Choice:

6

Operations Stopped...

Assignment:

step-1 : Create table with name Student70

(rollNo,name,branch,mid,phno,totMarks,per,Result)

step-2 : Construct JDBC Application to perform the following Operations

1.AddStudent

2.ViewAllStudents

3.ViewStudentByRollNo

4.UpdateStudentByRollNo(mid-phno)

5.DeleteStudentByRollNo

Calculations:

=>totMarks based on 6 Subject Marks.

(Exception Condition : The marks must be in b/w 0 to 100,else 'Exception')

==========

Dt : 11/1/2025

**imp*

'ResultSet' in JDBC:

=>'ResultSet' is an interface from java.sql package and which is instatiated to hold the data retrieved using select-queries.

(ResultSet-Objects will hold data retrived using select-queries)

=>ResultSet-Objects are categorized into two types:

1.NonScrollable ResultSet Objects

2.Scrollable ResultSet Objects

1.NonScrollable ResultSet Objects:

=>The ResultSet-Objects in which,the cursor can be moved only in one direction are known as NonScrollable ResultSet Objects.

(In NonScrollable ResultSet Objects,the cursor can be moved from top-of-table data to bottom-of-table data)

=>We use the following syntaxes to create NonScrollable ResultSet Objects:

syntax-1 : Using 'Statement'

```
Statement stm = con.createStatement();
ResultSet rs = stm.executeQuery("select-query");
```

syntax-2 : Using 'PreparedStatement'

```
PreparedStatement ps = con.prepareStatement("select-query-structure");
ResultSet rs = ps.executeQuery();
```

2.Scrollable ResultSet Objects:

=>The ResultSet-Objects in which,the cursor can be moved in both directions (forward-backward) are known as Scrollable ResultSet Objects.

=>we use the following syntaxes to create Scrollable ResultSet Objects:

syntax-1 : Using 'Statement'

```
Statement stm = con.createStatement(type,mode);
ResultSet rs = stm.executeQuery("select-query");
```

syntax-2 : Using 'PreparedStatement'

```
PreparedStatement ps = con.prepareStatement("select-query-structure",type,mode);
ResultSet rs = ps.executeQuery();
```

define 'type'?

=>'type' specifies the direction of cursor on ResultSet-Objects.

=>The following fields from 'ResultSet' represent "type":

```
public static final int TYPE_FORWARD_ONLY;
public static final int TYPE_SCROLL_INSENSITIVE;
public static final int TYPE_SCROLL_SENSITIVE;
```

define 'mode'?

=>'mode' specifies the action performed on ResultSet-Objects.

=>The following fields from 'ResultSet' represent "mode":

public static final int CONCUR_READ_ONLY;

public static final int CONCUR_UPDATABLE;

=>we use the following some important methods to control cursor on ResultSet-Objects:

1.afterLast()

2.beforeFirst()

3.previous()

4.next()

5.first()

6.last()

7.absolute(int)

8.relative(int)

1.afterLast():

=>afterLast()-method will move the cursor pointing after the last-row.

2.beforeFirst():

=>beforeFirst()-method will move the cursor pointing before the first-row

3.previous():

=>previous()-method will move the cursor in backward direction.

4.next():

=>next()-method will move the cursor in forward direction.

5.first():

=>first()-method will make the cursor point to the first-row

6.last():

=>last()-method will make the cursor point to the last-row.

7.absolute(int):

=>absolute(row_no) method will make the cursor point to the specified row_no.

8.relative(int):

=>relative(int) method will move the cursor forward or backward from the current position.

Dt : 20/1/2025

Ex:

Program : DBCon5.java

```
package test;
import java.sql.*;
public class DBCon5
{
	public static void main(String[] args)
	{
	  try {
		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
		System.out.println("***************Statement*************");
		Statement stm =
con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
				ResultSet.CONCUR_READ_ONLY);
		ResultSet rs1 = stm.executeQuery("select * from Product70");
		System.out.println("------absolute(3)---------");
		rs1.absolute(3);
		System.out.println(rs1.getString(1)+"\t"
				+rs1.getString(2)+"\t"
				+rs1.getFloat(3)+"\t"
				+rs1.getInt(4));
		System.out.println("------relative(-1)--------");
		rs1.relative(-1);
		System.out.println(rs1.getString(1)+"\t"
				+rs1.getString(2)+"\t"
				+rs1.getFloat(3)+"\t"
				+rs1.getInt(4));
		System.out.println("------last()-------");
		rs1.last();
		System.out.println(rs1.getString(1)+"\t"
				+rs1.getString(2)+"\t"
				+rs1.getFloat(3)+"\t"
				+rs1.getInt(4));
		System.out.println("-----first()------");
		rs1.first();
		System.out.println(rs1.getString(1)+"\t"
				+rs1.getString(2)+"\t"
				+rs1.getFloat(3)+"\t"
				+rs1.getInt(4));
	       System.out.println("********PreparedStatement********");
	       PreparedStatement ps = con.prepareStatement("select * from
Product70",
```

```
ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_READ_ONLY);
            ResultSet rs2 = ps.executeQuery();
            System.out.println("--------Products in reverse------");
            rs2.afterLast();
            while(rs2.previous()) {
                  System.out.println(rs2.getString(1)+"\t"
                              +rs2.getString(2)+"\t"
                              +rs2.getFloat(3)+"\t"
                              +rs2.getInt(4));
            }
        }catch(Exception e) {
            e.printStackTrace();
        }
      }
}
```

o/p:

```
***************Statement*************

------absolute(3)---------

E11     CDR     1100.0  11

------relative(-1)--------

G32     MDD     1800.0  18

------last()-------

F34     FDD     1000.0  10

-----first()------

A21     Mous    1200.0  12

********PreparedStatement********

--------Products in reverse------

F34     FDD     1000.0  10

E11     CDR     1100.0  11

G32     MDD     1800.0  18

A21     Mous    1200.0  12
```

---

*imp

Streams with Database:(Multi-Media data with Database)

define stream?(normal definition)

=>The continuous flow of data is known as stream.

Types of streams:

=>Java Support,the following two types of streams:

1.Byte Stream

2.Character Stream

1.Byte Stream:

=>The continuous flow of data in the form of 8-bits is known as Byte Stream or Binary Stream.

=>Byte Stream supports all Multi-Media data formats.

2.Character Stream:

=>The continuous flow of data in the form of 16-bits is known as Character Stream or Text Stream.

=>Character Stream is preferable for Text data,which means not preferable for Audio, Video,Image and Animation data.

=>we use the following sql-types to store Stream data:

(a)BLOB

(b)CLOB

(a)BLOB:

=>BLOB stands for 'Binary Large OBjects' and which is used to store Byte Stream data.

(b)CLOB:

=>CLOB stands for 'Character Large OBjects' and which is used to store Character Stream data.

Dt : 21/1/2025

**imp*

Loading Stream data to database Product:

step-1 : Create table with name StreamTab70(id,mfile)

Primary key : id

```
create table StreamTab70(id varchar2(10),mfile BLOB,primary key(id));
```

step-2 : Construct JDBC Application to load Image onto database Table.

Program : DBCoon6.java

```
package test;
import java.io.*;
```

```
import java.util.*;
import java.sql.*;
public class DBCon6 {
	public static void main(String[] args) {
	Scanner s = new Scanner(System.in);
	try(s;){
	Class.forName("oracle.jdbc.driver.OracleDriver");
	Connection con = DriverManager.getConnection
			("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
	PreparedStatement ps = con.prepareStatement
			("insert into StreamTab70 values(?,?)");
	System.out.println("Enter the Id to Store Image:");
	String id = s.nextLine();
	System.out.println("Enter the fPath&fName(source of Image)");
	String path = s.nextLine();
	File f = new File(path);
	if(f.exists()) {
		FileInputStream fis = new FileInputStream(path);
		ps.setString(1, id);
		ps.setBinaryStream(2, fis, f.length());
		int k = ps.executeUpdate();
		if(k>0) {
			System.out.println("Image Stored Successfully...");
		}
	}else {
```

```
                    System.out.println("Invalid fPath or fName....");
            }
        }catch(Exception e) {
            e.printStackTrace();
        }
            }
}
```

o/p:

Enter the Id to Store Image:

A11

Enter the fPath&fName(source of Image)

C:\Images\c-tara.jpg

Image Stored Successfully...

**imp*

Retrieving Image from database product:

Program : DBCon7.java

```
package test;
import java.util.*;
import java.io.*;
import java.sql.*;
public class DBCon7 {
        public static void main(String[] args) {
```

```
Scanner s = new Scanner(System.in);
try(s;){
        Class.forName("oracle.jdbc.driver.OracleDriver");
      Connection con = DriverManager.getConnection
                        ("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
      PreparedStatement ps = con.prepareStatement
                        ("select * from StreamTab70 where id=?");
      System.out.println("Enter the id to retrieve Image:");
      String id = s.nextLine();
      ps.setString(1, id);
      ResultSet rs = ps.executeQuery();
      if(rs.next()) {
            Blob b = rs.getBlob(2);
            byte by[] = b.getBytes(1, (int)b.length());
            System.out.println("Enter the fPath&fName(destination to store image)");
            String path = s.nextLine();
            FileOutputStream fos = new FileOutputStream(path);
            fos.write(by);
            System.out.println("Image retrieved Successfully...");
            fos.close();
      }else {
            System.out.println("Invalid id....");
      }
}catch(Exception e) {
        e.printStackTrace();
```

```
    }
        }
}
```

o/p:

Enter the id to retrieve Image:

A11

Enter the fPath&fName(destination to store image)

E:\Images\XYZ.jpg

Image retrieved Successfully...

Diagram:

==========

Dt : 22/1/2025

faq:

define 'FileInputStream'?

=>'FileInputStream' is a class from java.io package and which is instantiated to link (connect) the file to read Byte stream data.

syntax:

```
FileInputStream fis = new FileInputStream(path);
```

faq:

define 'FileOutputStream'?

=>'FileOutputStream' is a class from java.io package and which is instantiated to create a new file with zero KB and which is linked(connected) to write Byte Stream data.

syntax:

```
FileOutputStream fos = new FileOutputStream(path);
```

faq:

define setBinaryStream()?

=>setBinaryStream()-method will set the stream to parameter-index-field of PreparedStatement Object,and which specifies para-index,location and length.

syntax:

```
ps.setBinaryStream(2, fis, f.length());
```

faq:

define getBlob()?

=>getBlob()-method is from 'ResultSet' and which is used to create implementation object for 'Blob' interface,and this Blob-Object internally linked to stream-column of ResultSet.

syntax:

```
Blob b = rs.getBlob(2);
```

**imp*

faq:

define getBytes()?

=>getBytes()-method is from 'Blob' and which is used to convert Byte Stream into byte-array

syntax:

```
byte by[] = b.getBytes(1, (int)b.length());
```

**imp*

(3)CallableStatement:

=>'CallableStatement' is an interface from java.sql package and which is used to execute Procedures and functions on Database product.

=>we use prepareCall()-method from Connection-Interface to create implementation object for CallableStatement-Interface.

Method Signature of prepareCall():

```
public abstract java.sql.CallableStatement prepareCall(java.lang.String)
                                        throws java.sql.SQLException;
```

syntax:

```
CallableStatement cs = con.prepareCall("{call Proc/Func}");
```

faq:

wt is the diff b/w

(i)Functions

(ii)Member Functions

(iii)Methods

(i)Functions:

=>The part of program which is executed out of main-program in C-Lang is known as Function.

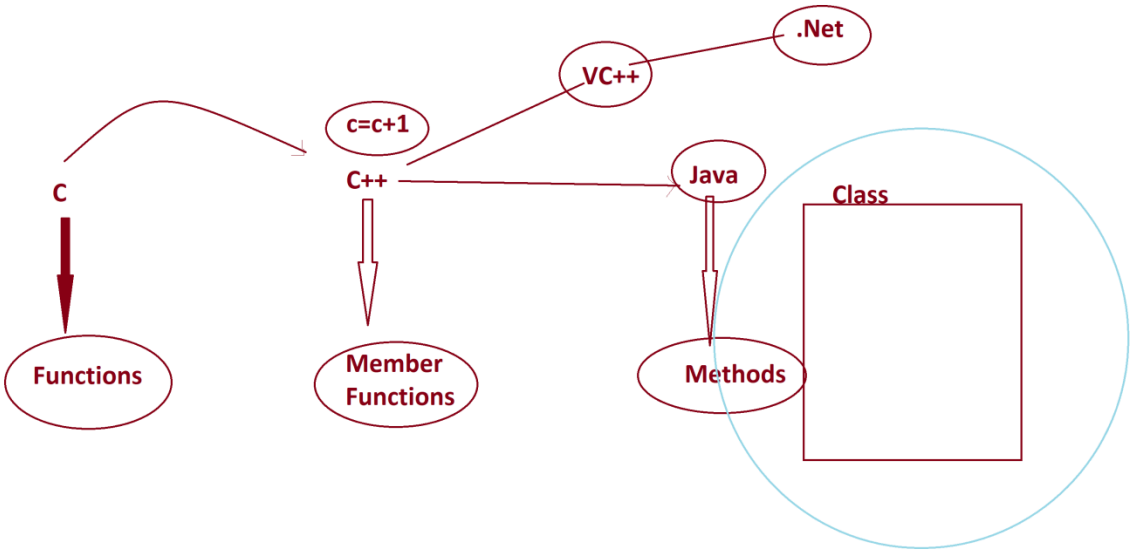
(ii)Member Functions:

=>The functions which are declared as members of class in C++ are known as Member Functions.

(Member Functions can be declared inside the class or outside the class with class reference)

(iii)Methods:

=>The functions which are declared only inside the class in Java are known as Methods.



Dt : 23/1/2025

faq:

define Procedure?

=>Procedure is a set-of-queries executed on database product and which will not return any value.

(Procedure in SQL is NonReturn-Type)

=>Procedures are also known as Stored Procedures.

structure of Procedure:

```
create or replace procedure Procedure_name
(para_list) is
begin
 query-1
 query-2
 ...
end;
/
```

Types of Procedures:

=>According to JDBC,procedures are categorized into three types:

(i)IN-Parameter Procedures

(ii)OUT-Parameter Procedures

(iii)IN-OUT-Parameter Procedures

(i)IN-Parameter Procedures:

=>The Procedures which take the data from Java-Program and send to Database product are known as IN-Parameter Procedures.

(ii)OUT-Parameter Procedures:

=>The Procedures which take the data from Database product and send to Java-Program are known as OUT-Parameter Procedures.

(iii)IN-OUT-Parameter Procedures:

=>The Procedures which perform both operations are known as IN-OUT-Parameter Procedures.

**imp*

Creating and Executing Procedure(Stored Procedure):

step-1 : Create the following DB Tables

BankCustomer70(accno,name,bal,acctype)

CustAddress70(accno,city,state,pincode)

CustContact70(accno,mid,phno)

```
create table BankCustomer70(accno number(15),name varchar2(15),bal number(10,2),
acctype varchar2(15),primary key(accno));

create table CustAddress70(accno number(15),city varchar2(15),state varchar2(15),
pincode number(10),primary key(accno));
```

create table CustContact70(accno number(15),mid varchar2(25),phno number(15),

primary key(accno));

step-2 : Construct Procedure to insert Customer details

create or replace procedure CreateAccount70

(ano number,nm varchar2,bl number,atype varchar2,cty varchar2,st varchar2,pcode number,

md varchar2,pno number) is

begin

insert into BankCustomer70 values(ano,nm,bl,atype);

insert into CustAddress70 values(ano,cty,st,pcode);

insert into CustContact70 values(ano,md,pno);

end;

/

step-3 : Construct JDBC application to execute procedure.

syntax:

CallableStatement cs = con.prepareCall

("{call CreateAccount70(?,?,?,?,?,?,?,?,?)}");

Program : DBCon8.java

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon8 {
      public static void main(String[] args) {
       Scanner s = new Scanner(System.in);
```

```
        try(s;){
          Class.forName("oracle.jdbc.driver.OracleDriver");
              Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
              CallableStatement cs = con.prepareCall
                          ("{call CreateAccount70(?,?,?,?,?,?,?,?,?)}");
              System.out.println("Enter the AccNo:");
              long accNo = Long.parseLong(s.nextLine());
              System.out.println("Enter the Cust-Name:");
        String name = s.nextLine();
        System.out.println("Enter the Cust-Bal:");
        float bal = Float.parseFloat(s.nextLine());
        System.out.println("Enter the Cust-AccType:");
        String accType = s.nextLine();
        System.out.println("Enter the Cust-City:");
        String city = s.nextLine();
        System.out.println("Enter the Cust-State:");
        String state = s.nextLine();
        System.out.println("Enter the Cust-PinCode:");
        int pinCode = Integer.parseInt(s.nextLine());
        System.out.println("Enter the Cust-MailId:");
        String mId = s.nextLine();
        System.out.println("Enter the Cust-PhoneNo:");
        long phNo = Long.parseLong(s.nextLine());

        //Loading data to CallableStatement Object
        cs.setLong(1, accNo);
        cs.setString(2, name);
        cs.setFloat(3, bal);
        cs.setString(4, accType);
        cs.setString(5, city);
        cs.setString(6, state);
        cs.setInt(7, pinCode);
        cs.setString(8, mId);
        cs.setLong(9, phNo);

        cs.execute();//Procedure Executed...
        System.out.println("Customer Account created Successfully....");
        }catch(Exception e) {
          e.printStackTrace();
        }
       }
}
```

o/p:

Enter the AccNo:

123456

Enter the Cust-Name:

Alex

Enter the Cust-Bal:

1000

Enter the Cust-AccType:

Savings

Enter the Cust-City:

Hyd

Enter the Cust-State:

TG

Enter the Cust-PinCode:

506112

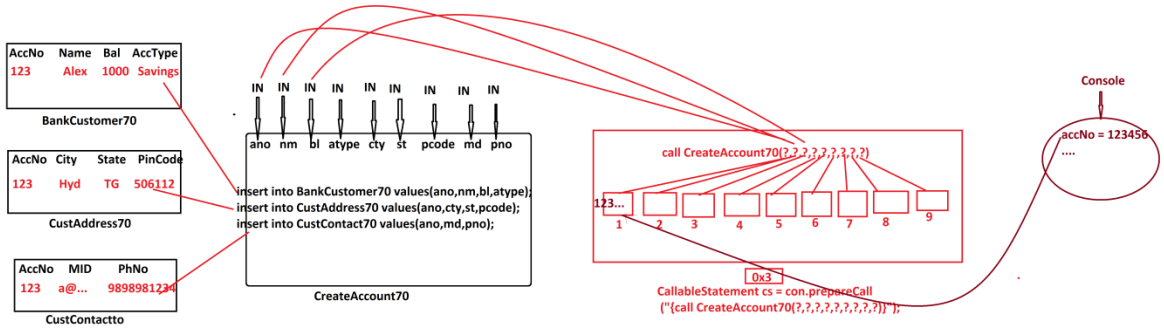
Enter the Cust-MailId:

a@gmail.com

Enter the Cust-PhoneNo:

9898981234

Customer Account created Successfully....



SQL> select * from BankCustomer70;

ACCNO	NAME	BAL	ACCTYPE
123456	Alex	1000	Savings

SQL> select * from CustAddress70;

ACCNO	CITY	STATE	PINCODE
123456	Hyd	TG	506112

SQL> select * from CustContact70;

ACCNO	MID	PHNO
123456	a@gmail.com	9898981234

SQL>

Assignment:

step-1 : Create the following DB Tables

StuData70(rollno,name,branch)

StuAddress70(rollno,city,state,pincode)

StuContact70(rollno,mid,phno)

StuMarks70(rollno,sub1,sub2,sub3,sub4,sub5,sub6)

StuResult70(rollno,totmarks,per,result)

step-2 : Construct Procedure to insert Student details

step-3 : Construct JDBC Application to execute Procedure

Dt : 24/1/2025

**imp*

Construct Procedure to retrieve Bank Customer details based on accNo.

Procedure:

```
create or replace procedure RetrieveDetails70
(ano number,nm OUT varchar2,bl OUT number,atype OUT varchar2,cty OUT varchar2,
st OUT varchar2,pcode OUT number,md OUT varchar2,pno OUT number) is
begin
  select name,bal,acctype into nm,bl,atype from BankCustomer70 where accno=ano;
  select city,state,pincode into cty,st,pcode from CustAddress70 where accno=ano;
  select mid,phno into md,pno from CustContact70 where accno=ano;
end;
/
```

Program:

Construct JDBC Program to execute Procedure.

Program : DBCon9.java

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon10 {
        public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
```

```
try(s;){
        Class.forName("oracle.jdbc.driver.OracleDriver");
      Connection con = DriverManager.getConnection
                        ("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
      CallableStatement cs = con.prepareCall
                        ("{call RetrieveDetails70(?,?,?,?,?,?,?,?,?)}");
      System.out.println("Enter the Cust-AccNo to retrieve details:");
      long accNo = Long.parseLong(s.nextLine());
      cs.setLong(1, accNo);
      cs.registerOutParameter(2, Types.VARCHAR);
      cs.registerOutParameter(3, Types.FLOAT);
      cs.registerOutParameter(4, Types.VARCHAR);
      cs.registerOutParameter(5, Types.VARCHAR);
      cs.registerOutParameter(6, Types.VARCHAR);
      cs.registerOutParameter(7, Types.INTEGER);
      cs.registerOutParameter(8, Types.VARCHAR);
      cs.registerOutParameter(9, Types.BIGINT);
      cs.execute();
      System.out.println("--------Customer Details-------");
      System.out.println("Cust-Name : "+cs.getString(2));
     System.out.println("Cust-Bal : "+cs.getFloat(3));
     System.out.println("Cust-AccType : "+cs.getString(4));
     System.out.println("Cust-City : "+cs.getString(5));
     System.out.println("Cust-State : "+cs.getString(6));
     System.out.println("Cust-PinCode : "+cs.getInt(7));
```

```
            System.out.println("Cust-MailId : "+cs.getString(8));
            System.out.println("Cust-PhoneNo : "+cs.getLong(9));
             con.close();
        }catch(Exception e) {
              e.printStackTrace();
        }
            }
}
```

o/p:

Enter the Cust-AccNo to retrieve details:

123456

--------Customer Details-------

Cust-Name : Alex

Cust-Bal : 1000.0

Cust-AccType : Savings

Cust-City : Hyd

Cust-State : TG

Cust-PinCode : 506112

Cust-MailId : a@gmail.com

Cust-PhoneNo : 9898981234

Assignment:

Construct Procedure to retrieve Complete Student details based on RollNo.

---

faq:

define Function?

=>Function is a set-of-queries executed on Database product and which will return the value.

(Function in SQL means return_type)

=>In SQL-Function also we use 'return' statement to return the value

stucture of Function:

```
create or replace Function Function_name
(para_list) return data_type as var data_type;
begin
  queries
  return var;
end;
/
```

**imp*

Construct Function to display Customer Balance based on accNo.

Function:

```
create or replace Function RetrieveBalance70
(ano number) return number as x number;
begin
  select bal into x from BankCustomer70 where accno=ano;
```

```
  return x;
end;
/
```

Program : DBCon10.java

```
package test;
import java.util.*;
import java.sql.*;
public class DBCon10 {
   public static void main(String[] args) {
          Scanner s = new Scanner(System.in);
          try(s;){
                Class.forName("oracle.jdbc.driver.OracleDriver");
          Connection con = DriverManager.getConnection
                        ("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
          CallableStatement cs = con.prepareCall
                        ("{call ?:=RetrieveBalance70(?)}");
          System.out.println("Enter the AccNo to retrieve Balance:");
          long accNo = s.nextLong();
          cs.setLong(2, accNo);
          cs.registerOutParameter(1, Types.FLOAT);
          cs.execute();
          System.out.println("Cust-Balance:"+cs.getFloat(1));
          con.close();
          }catch(Exception e) {
```

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

o/p:

Enter the AccNo to retrieve Balance:

123456

Cust-Balance:1000.0

Assignment:

Construct Function to retrieve Student Percentage based on RollNo.

DT : 27/1/2025

*imp

Transaction Management in JDBC:

define Transaction?

=>The set-of-statements which are executed on a single resource or multiple resources using ACID properties,is known as Transaction.

A - Atomicity

C - Consistency

I - Isolation

D - Durability

A - Atomicity

=>The process in which all statements in Transaction are executed at-a-time or not-at-all,is known as Atomicity.

C - Consistency

=>The process in which the select state-of-resources remain same until the transaction is completed,is known as Consistency.

I - Isolation

=>The process in which multiple users execute independently,is known as Isolation.

D - Durability

=>The process in which storing the transaction details and making it available for user,is known as Durability.

---

faq:

define Transaction Management?

=>The process of controlling the transcation from starting to ending is known as Transaction Management.

=>We use the following methods in Transaction Management:

1.getAutoCommit()

2.setAutoCommit()

3.setSavepoint()

4.releaseSavepoint()

5.commit()

6.rollback()

1.getAutoCommit():

=>getAutoCommit()-method is used to check the status of commit-operation.

Method Signature:

public abstract boolean getAutoCommit() throws java.sql.SQLException;

syntax:

boolean b = con.getAutoCommit();

2.setAutoCommit():

=>setAutoCommit()-method is used to stop the auto commit operation.

Method Signature:

public abstract void setAutoCommit(boolean) throws java.sql.SQLException;

syntax:

con.setAutoCommit(false);

3.setSavepoint():

=>setSavepoint()-method will give specification for rollback operation when transaction failed.

Method Signature:

public abstract java.sql.Savepoint setSavepoint() throws java.sql.SQLException;

syntax:

Savepoint sp = con.setSavepoint();

4.releaseSavepoint():

=>releaseSavepoint()-method is used to delete the savepoint.

Method Signature:

public abstract void releaseSavepoint(java.sql.Savepoint) throws java.sql.SQLException;

syntax:

con.releaseSavepoint(sp);

5.commit():

=>commit()-method is used to perform commit-operation when the transaction is Successfull,which the save permanently to database when all statements in Transaction are executed Successfully.

Method Signature:

public abstract void commit() throws java.sql.SQLException;

syntax:

con.commit();

6.rollback():

=>rollback()-method will perform rollback operation,which means reset the buffer and move the control to savepoint.

Method Signature:

public abstract void rollback() throws java.sql.SQLException;

public abstract void rollback(java.sql.Savepoint) throws java.sql.SQLException;

syntax:

con.rollback(sp);

Note:

(i)JDBC-Applications will perform auto-commit operation.

(ii)To perform Transaction Management,we have stop auto-commit-operation.

Diagram:

Transaction : Book Ticket using 'Book My Show'

1.Login process
2.Region
3.Movie
4.Date
5.No Tickets
6.Theater
7.Select the seats
8.Payment
  (i)Card
  (ii)UPI
  ....
9.Tickects Confirmation MSg
  (Mobile/MailId)
10.Logout

Dt : 28/1/2027

DB Table : Bank70(accno,name,balance,acctype)

Primary Key : accno

create table Bank70(accno number(15),name varchar2(15),balance number(10,2),

acctype varchar2(15),primary key(accno));

insert into Bank70 values(6123456,'Alex',12000,'Savings');

insert into Bank70 values(313131,'Ram',500,'Savings');

*SQL> select * from Bank70;*

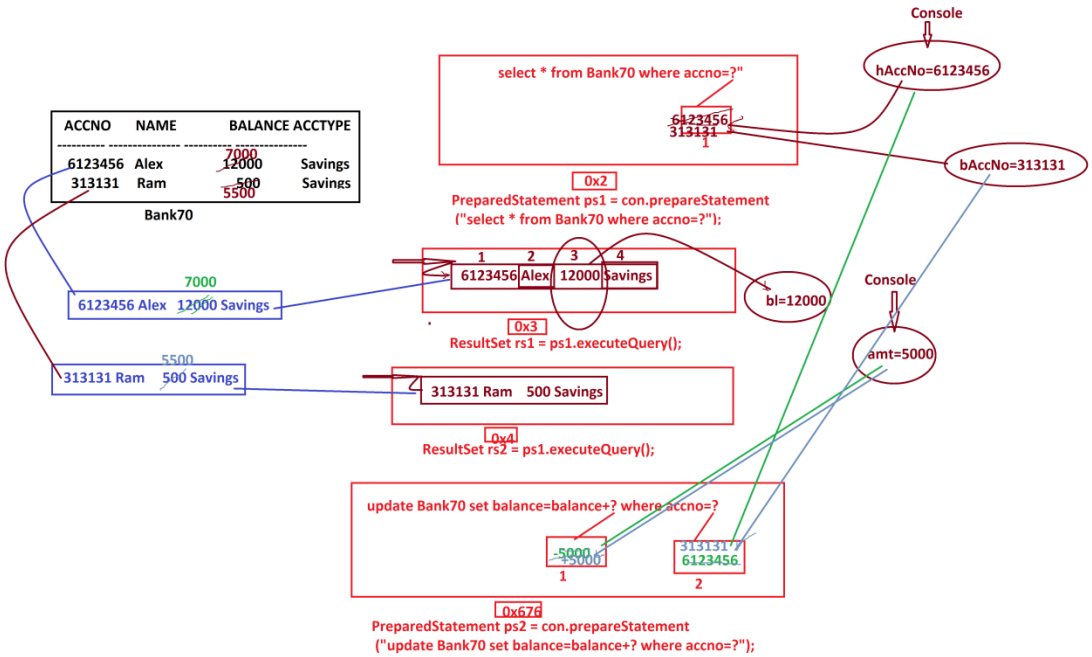
```
    ACCNO NAME            BALANCE ACCTYPE
---------- --------------- ---------- ---------------
   6123456 Alex             12000 Savings
    313131 Ram                500 Savings
```

SQL>

Transaction : Transfer amt:5000/- from accNo:6123456 to accNo:313131

Statement-1 : Subtract amt:5000/- from accNo:6123456

Statement-2 : Add amt:5000/- to accNo:313131



Program : DBCon11.java

```
package test;

import java.util.*;

import java.sql.*;

public class DBCon11 {

        public static void main(String[] args) {

    Scanner s = new Scanner(System.in);

    try(s;){

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

```
Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
PreparedStatement ps1 = con.prepareStatement
                ("select * from Bank70 where accno=?");
PreparedStatement ps2 = con.prepareStatement
                ("update Bank70 set balance=balance+? where accno=?");
System.out.println("commit status : "+con.getAutoCommit());
con.setAutoCommit(false);
System.out.println("commit status : "+con.getAutoCommit());
Savepoint sp = con.setSavepoint();
System.out.println("Enter the HomeAccNo:");
long hAccNo = s.nextLong();
ps1.setLong(1, hAccNo);
ResultSet rs1 = ps1.executeQuery();
if(rs1.next()) {
        float bl = rs1.getFloat(3);
        System.out.println("Enter the benificieryAccNo:");
        long bAccNo = s.nextLong();
        ps1.setLong(1, bAccNo);
        ResultSet rs2 = ps1.executeQuery();
        if(rs2.next()) {
                System.out.println("Enter the amt to be transferred:");
                float amt = s.nextFloat();
                if(amt<=bl) {
                        //Statement-1 : Subtract amt:5000/- from accNo:6123456
```

```
                    ps2.setFloat(1,-amt);
                    ps2.setLong(2, hAccNo);
                    int p = ps2.executeUpdate();//Buffer Updated

                    //Statement-2 : Add amt:5000/- to accNo:313131
                    ps2.setFloat(1, +amt);
                    ps2.setLong(2, bAccNo);
                    int q = ps2.executeUpdate();//Buffer Updated

                    if(p==1 && q==1) {
                        con.commit();//Updating Database
                        System.out.println("Transaction Successfull...");
                    }else {
                        System.out.println("Transaction failed...");
                    }
                }else {
                    con.rollback(sp);
                    System.out.println("InSufficient Fund...");
                }
            }else {
                System.out.println("Invalid bAccNo...");
            }
        }else {
            System.out.println("Invalid hAccNo...");
        }
```

```
        }catch(Exception e) {
          e.printStackTrace();
    }
        }
}
```

o/p:

commit status : true

commit status : false

Enter the HomeAccNo:

6123456

Enter the benificieryAccNo:

313131

Enter the amt to be transferred:

5000

Transaction Successfull...

Assignment:

(i)Update above program by generated else-block-messages from 'catch-block'

(ii)Create Table : TransLog70(hccno,baccno,amt,datetime)

Primary Key : haccno

when Transaction Successfull,then record the details.

**imp*

Batch Processing in JDBC:

=>The process of collecting multiple queries as batch and executing at-a-time is known as Batch Processing or Batch Update Processing.

=>In Batch Processing we cannot perform select-operation.

=>We use 'Statement' to perform batch processing,because we can update multiple db tables at-a-time

=>We use the following methods to perform Batch Processing:

1.addBatch()

2.executeBatch()

3.clearBatch()

1.addBatch():

=>addBatch()-method is used to add queries to batch.

Method Signature:

public abstract void addBatch(java.lang.String) throws java.sql.SQLException;

2.executeBatch():

=>executeBatch()-method is used to execute all queries from the batch.

Method Signature:

public abstract int[] executeBatch() throws java.sql.SQLException;

3.clearBatch():

=>clearBatch()-method will delete all queries from the batch and destroy the batch

Method Signature:

public abstract void clearBatch() throws java.sql.SQLException;

Dt : 29/1/2025

Program : DBCon12.java

```
package test;

import java.sql.*;

import java.util.*;

public class DBCon12 {

        public static void main(String[] args) {

    Scanner s = new Scanner(System.in);

    try(s;){

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

        Connection con = DriverManager.getConnection

                        ("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");

        Statement stm = con.createStatement();


        System.out.println("------Add Customer to Customer70------");

        System.out.println("Enter the Cust-No:");

        int cId = Integer.parseInt(s.nextLine());

        System.out.println("Enter the Cust-Name:");

        String cName = s.nextLine();

        System.out.println("Enter the Cust-City:");

        String cCity = s.nextLine();

        System.out.println("Enter the Cust-MailId:");

        String mId = s.nextLine();

        System.out.println("Enter the Cust-PhNo:");

        long phNo = Long.parseLong(s.nextLine());
```

```
        stm.addBatch
        ("insert into Customer70 values("+cId+",'"+cName+"','"+cCity+"','"+mId+"',"+phNo+")");

        System.out.println("------Update operation on Product70------");
        System.out.println("Enter the Prod-Code to update price and qty:");
        String pCode = s.nextLine();
        System.out.println("Enter the NewPrice for Product:");
        float price = Float.parseFloat(s.nextLine());
        System.out.println("Enter the NewQty for Product:");
        int qty = Integer.parseInt(s.nextLine());

        stm.addBatch
        ("update Product70 set price="+price+",qty="+qty+" where code='"+pCode+"'");

        int k[] = stm.executeBatch();
        for(int i : k)
        {
                System.out.println("Query executed : "+i);
        }//end of loop
        stm.clearBatch();
        con.close();
    }catch(Exception e) {
        e.printStackTrace();
    }
```

```
        }
}
```

o/p:

------Add Customer to Customer70------

Enter the Cust-No:

6666

Enter the Cust-Name:

DERT

Enter the Cust-City:

Hyd

Enter the Cust-MailId:

d@gmail.com

Enter the Cust-PhNo:

7675431234

------Update operation on Product70------

Enter the Prod-Code to update price and qty:

F34

Enter the NewPrice for Product:

1500

Enter the NewQty for Product:

30

Query executed : 1

Query executed : 1

----

define MetaData?

=>The data which is holding information about other data is known as MetaData.

=>According to JDBC,Object holding information about other object is known as MetaData Object.

=>The following components in JDBC generate MetaData Objects:

1.DatabaseMetaData

2.ParameterMetaData

3.ResultSetMetaData

1.DatabaseMetaData:

=>DatabaseMetaData is an interface from java.sql and which is instatiated to hold the information about Connection-object.

=>we use getMetaData()-method from 'Connection' to create implementation object for DatabaseMetaData.

Method Signature:

public abstract java.sql.DatabaseMetaData getMetaData() throws java.sql.SQLException;

syntax:

DatabaseMetaData dmd = con.getMetaData();

2.ParameterMetaData:

=>ParameterMetaData is an interface from java.sql package and which is instantiated to hold information about PreparedStatement-Object.

=>we use getParameterMetaData()-method from PreparedStatement to create implementation Object for ParameterMetaData.

Method Signature:

public abstract java.sql.ParameterMetaData getParameterMetaData()

throws java.sql.SQLException;

syntax:

ParameterMetaData pmd = ps.getParameterMetaData();

**Imp*

3.ResultSetMetaData:

=>ResultSetMetaData is an interface from java.sql package and which is instantiated to hold information about ResultSet-Object

=>we use getMetaData()-method from ResultSet to create implementation object for ResultSetMetaData.

Method Signature:

public abstract java.sql.ResultSetMetaData getMetaData() throws java.sql.SQLException;

syntax:

ResultSetMetaData rsmd = rs.getMetaData();

Program : DBCon13.java

```
package test;
import java.sql.*;
public class DBCon13 {
      public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
                System.out.println("*******DatabaseMetaData******");
                DatabaseMetaData dmd = con.getMetaData();
                System.out.println("DB Name :
"+dmd.getDatabaseProductName());
                System.out.println("DB Version :
"+dmd.getDatabaseProductVersion());
                System.out.println("DB driver : "+dmd.getDriverName());
        }catch(Exception e) {
            e.printStackTrace();
```

```
            }
        }
}
```

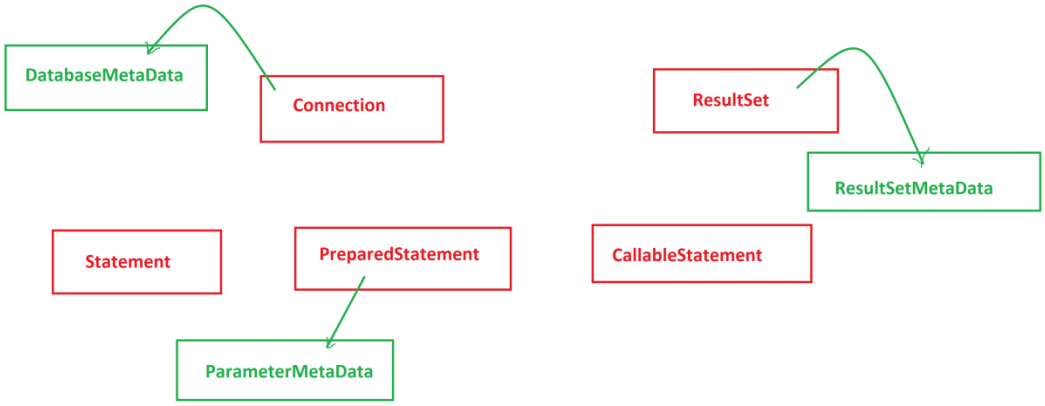
o/p:

********DatabaseMetaData*******

DB Name : Oracle

DB Version : Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

DB driver : Oracle JDBC driver



Dt : 30/1/2025

Program : DBCon14.java

```
package test;
import java.sql.*;
public class DBCon14
{
	public static void main(String[] args)
	{
	  try {
	  Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
		PreparedStatement ps1 = con.prepareStatement
				("insert into Product70 values(?,?,?,?)");
		System.out.println("********ParameterMetaData*******");
		ParameterMetaData pmd = ps1.getParameterMetaData();
		System.out.println("Para count :"+pmd.getParameterCount());
		PreparedStatement ps2 = con.prepareStatement
				("select id,mailid,phno from Customer70");
		ResultSet rs = ps2.executeQuery();
		System.out.println("------deatails-----");
		while(rs.next()) {
			System.out.println(rs.getInt(1)+"\t"
					+rs.getString(2)+"\t"
					+rs.getLong(3));
		}//end of loop
		System.out.println("*****ResultSetMetaData******");
		ResultSetMetaData rsmd = rs.getMetaData();
		System.out.println("Col Count : "+rsmd.getColumnCount());
		System.out.println("2nd Col Name :"+rsmd.getColumnName(2));
	  }catch(Exception e) {
	      e.printStackTrace();
	  }
	}
}
```

o/P:

*********ParameterMetaData*******

Para count :4

------deatails-----

1234 alex@gmail.com 9898981234

2312 ram@gmail.com 7676761234

3212 raj@gmail.com 4343431234

5454 d@gmail.com 9898983213

6666 d@gmail.com 7675431234

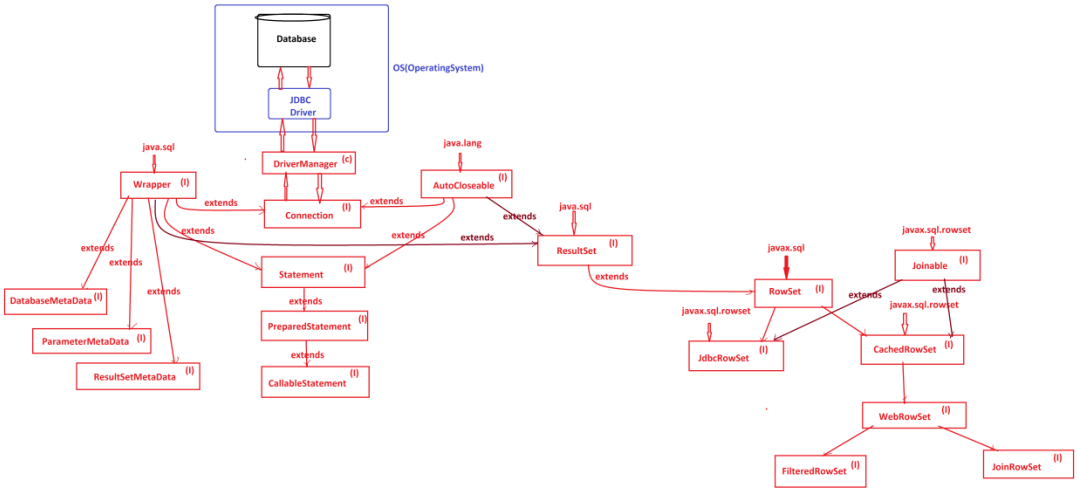
*****ResultSetMetaData******

Col Count : 3

2nd Col Name :MAILID

***imp**

Hierarchy of JDBC API:



***Dt : 31/1/2025**

***faq:**

***define 'Wrapper' in JDBC?**

=>'Wrapper' is an interface from java.sql package and which is used to check the JDBC-Component binded to Database or not.

=>The following are two important methods of 'Wrapper':

```
public abstract <T> T unwrap(java.lang.Class<T>) throws java.sql.SQLException;
public abstract boolean isWrapperFor(java.lang.Class<?>) throws java.sql.SQLException;
```

faq:

define 'AutoCloseable'?

=>'AutoCloseable' is an interface from java.lang package and which supports auto-closing operations.

=>The resource-components which are used in try-with-resource statement must be extended or implemented from java.lang.AutoCloseable interface.

Program : DBCon15.java

```
package test;
import java.sql.*;
import java.util.*;;
public class DBCon15 {
	public static void main(String[] args) {
	Scanner s = new Scanner(System.in);
	 try(s;)//Java9
	 {
		try
		{
			Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","tiger");
    try(con;){
        Statement stm = con.createStatement();
        try(stm;){
                System.out.println("Enter the query(Create/Insert/Update/Delete)");
                String qry = s.nextLine();
                int k = stm.executeUpdate(qry);
                System.out.println("Value k : "+k);
                if(k>=0) {
                        System.out.println("query executed successfully....");
                }
        }//end of try
    }//end of try
        }//end of try
        catch(SQLSyntaxErrorException qs) {
                System.out.println(qs.getMessage());
        }
        catch(SQLIntegrityConstraintViolationException se) {
                System.out.println(se.getMessage());
                System.out.println(se.getErrorCode());
        }
        catch(Exception e)
        {
                e.printStackTrace();
```

```
                }
        }//end of try with resource
        }
}
```

o/p:

Enter the query(Create/Insert/Update/Delete)

create table Emp70(id varchar2(10),name varchar2(15),desg varchar2(10),primary key(id))

Value k : 0

query executed successfully....

o/p:

Enter the query(Create/Insert/Update/Delete)

insert into Emp70 values('A121','Alex','SE')

Value k : 1

query executed successfully....

o/p:

Enter the query(Create/Insert/Update/Delete)

update Emp70 set desg='SE' where id='A231'

Value k : 1

query executed successfully....

o/p:

Enter the query(Create/Insert/Update/Delete)

delete from Emp70 where id='A231'

Value k : 1

query executed successfully....

---

faq:

define 'RowSet'?

=>'RowSet' is an interface from javax.sql package and which is extended from 'java.sql.ResultSet'

=>'RowSet' will hold data which is retrived from select-queries.

=>'RowSet-Objects' are automatically Scrollable-Objects.

=>'RowSet' categorized into two types:

1.JdbcRowSet

2.CachedRowSet

1.JdbcRowSet:

=>when we use JdbcRowSet,the connection to database willnot be disconnected after retriving the data.

2.CachedRowSet:

=>when we use CachedRowSet,the connection to database will be disconnected automatically after retrieving the data.

=>In realtim,we use CachedRowSet in the form 'WebRowSet' and which is categorized into two types:

(a)FilteredRowSet

(b)JoinRowSet

(a)FilteredRowSet:

=>FilteredRowSet will hold the data which is retrieved based on condition.

(b)JoinRowSet:

=>JoinRowSet will hold the data which is joined from more than one rowset objects.

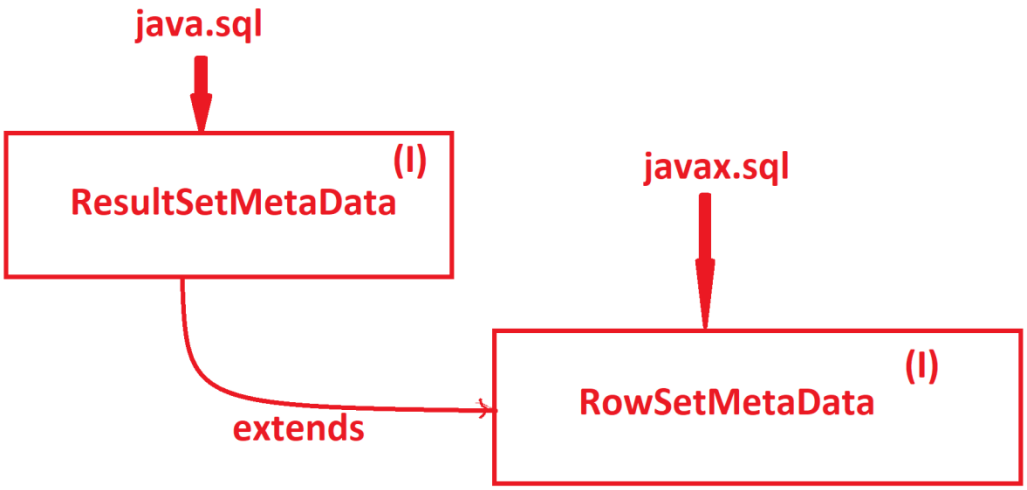
Dt : 1/2/2025

Note:

=>'javax.sql.RowSetMetaData' will hold information about 'RowSet' Objects.

=>'RowSetMetaData' is extended from 'ResultSetMetaData'

Diagram:



**imp*

define JDBC driver?

=>The driver which is used to establish communication b/w java-program and database product is known as JDBC driver(Java DataBase Connectivity driver)

Types of JDBC drivers:

=>JDBC drivers are categorized into four types:

1.JDBC-ODBC bridge driver(Type-1 driver)

2.Native API driver(Type-2 driver)

3.Network protocol driver(Type-3 driver)

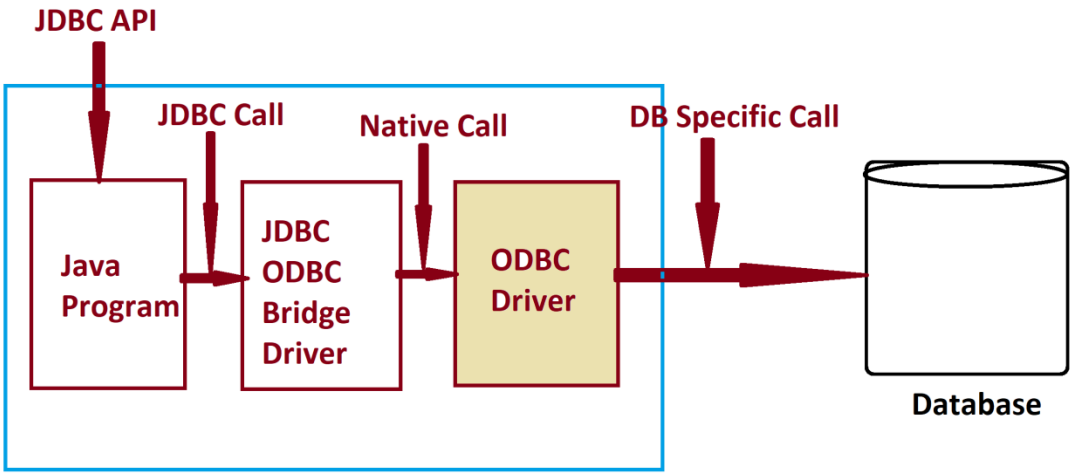
4.Thin driver(Type-4 driver)

1.JDBC-ODBC bridge driver(Type-1 driver):

=>The Type-1 driver will take the support of ODBC-driver to establish connection to Database product.

=>when we use Type-1 driver JDBC-Call is converted into Native call,and the Native Call is converted into DB Specific call for connetion.

Diagram:



DisAdvantage:

=>Type-1 driver internally uses more conversions,and which waste the execution time and degrades the performance of an application.

Note:

=>From Java8 version(2014) onwards Type-1 driver support is not available in Java.

faq:

define ODBC driver?

=>ODBC stands for 'Open DataBase Connectivity',and this driver will support to establish connection to any type of database.

=>This ODBC driver is PlatForm dependent driver,because which internally uses C/C++ codes.

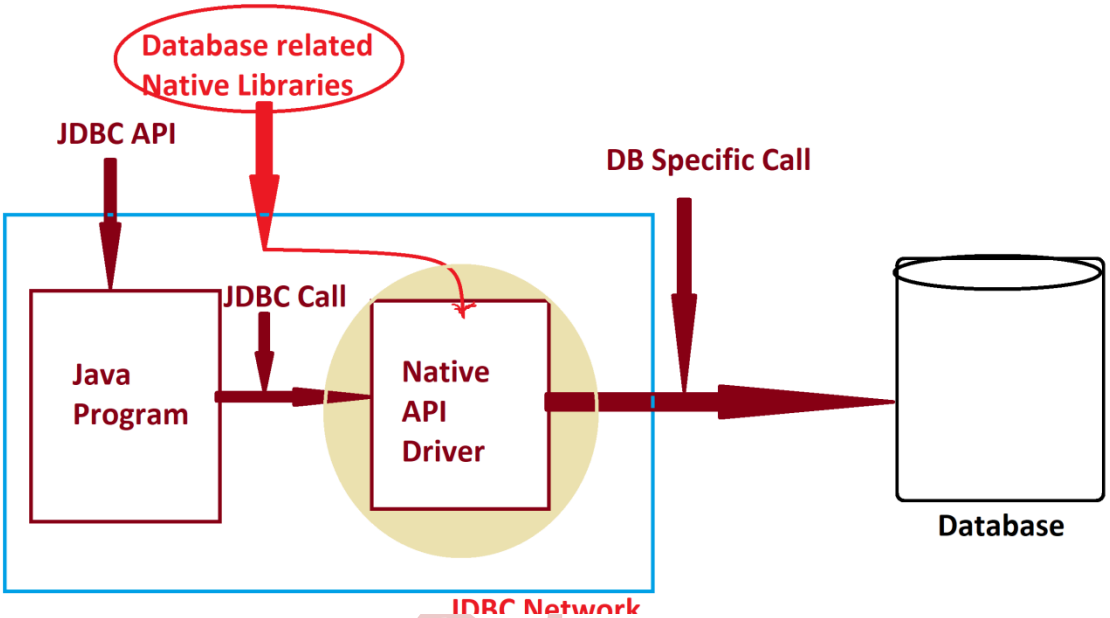
2.Native API driver(Type-2 driver):

=>Type-2 driver will take the support of Database related Native Libraries to establish

Connection to database product.

=>To Use Type-2 driver,the Client Computer must be installed with Database related Native libraries.

Diagram:



DisAdvantage:

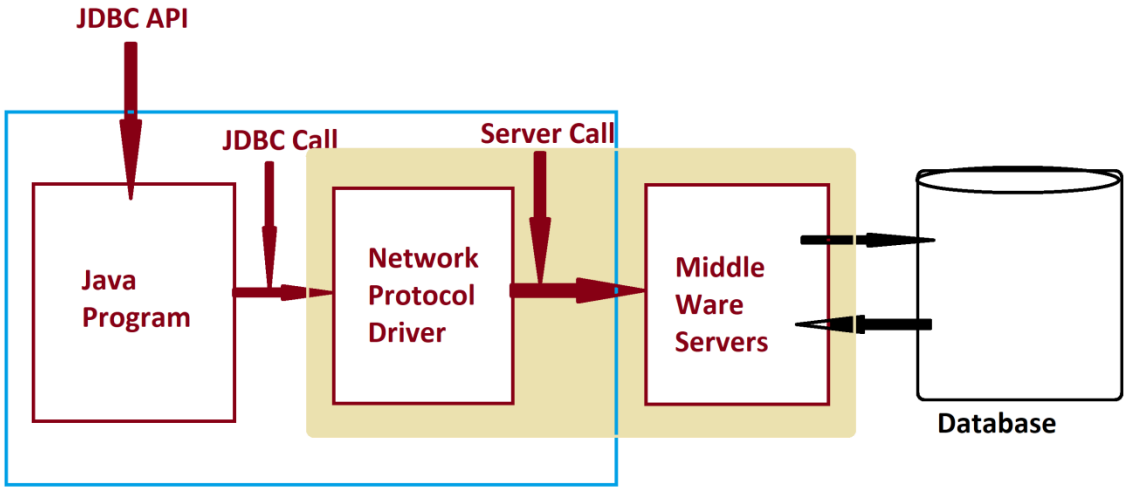
=>when we construct application with Type-2 driver,then the application will become Database dependent and which is not preferable in realtime.

3.Network protocol driver(Type-3 driver):

=>Type-3 driver will take the support of Intermediate MiddleWare server to establish connection to database product.

=>In this process Middleware Servers will hold database related connection code.

Diagram:



DisAdvantage:

=>when we want to use Type-3 driver,we have to make Network settings in ClientComputer and the Network components are involved in execution process and degrades the performance of an application.(Execution time increases)

****imp***

4.Thin driver(Type-4 driver):

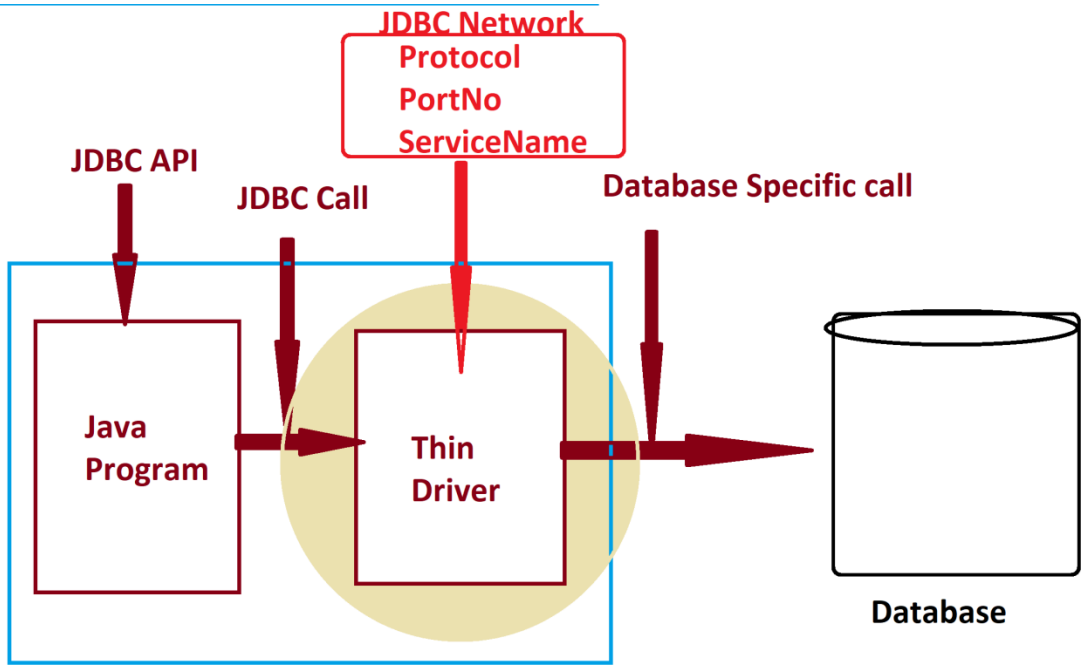
=>Type-4 driver will take the support of Database-Network-protocol to establish connection to to database product.

=>Type-4 driver is pure java-driver.

=>Type-4 driver is PlatForm independent driver.

=>Type-4 driver is high performance driver

Diagram:



==========

Summary of Objects generated from CoreJava:

1.User defined Class Objects

2.String-Objects

3.WrapperClass-Objects

4.Array-Objects

5.Collection<E>-Objects

6.Map<K,V>-Objects

7.Enun<E>-Objects

Summary of Objects generated from JDBC:

1.Connection-Object

2.Statement-Object

3.PreparedStatement-Object

4.CallableStatement-Object

5.ResultSet-Object

(i)Scrollable ResultSet Objects

(ii)NonScrollable ResultSet Objects

6.RowSet-Object

(a)JdbcRowSet Objects

(b)CachedRowSet Objects

=>WebRowSet Objects

(i)FilteredRowSet Objects

(ii)JoinRowSet Objects

7.DatabaseMetaData-Object

8.ParameterMetaData-Object

9.ResultSetMetaData-Object

10.RowSetMetaData-Object

Dt : 3/2/2025

faq:

define Serialization process?

=>The process of coverting Object-state into binary stream or byte stream is known as Serialization process.

define DeSerialization process?

=>The process of converting binary stream into Object state is known as DeSerialization process.

Note:

=>To perform Serialization and DeSerialization process,the class must be implemented from 'java.io.Serializable' interface.

faq:

wt is the advantage of Serialization process?

=>Using serialization process,we can make Object-state available in the form of stream and which can be moved on the Network from one location to another location.

Based on Serialization process,the Objects in Java are categorized into two types:

1.Serializable Objects

2.NonSerializable Objects

1.Serializable Objects:

=>The objects generated from the classes,which are implemented from 'Serializable'

interface are known as Serializable Objects.

Ex:

All CoreJava Objects

2.NonSerializable Objects:

=>The objects generated from the classes,which are not implemented from 'Serializable'

interface are knonw as NonSerializable Objects.

Ex:

JDBC Objects

Note:

=>According Vendor(Oracle Corporation),the java is available in the following:

1.JavaSE

2.JavaEE

3.JavaME

1.JavaSE:

=>JavaSE stands for 'Java Standard Edition' and which is used to develop NonServer

Applications,which means Stand-Alone-Applications.

Ex:

JavaSE includes 'CoreJava+JDBC'

2.JavaEE:

=>JavaEE stands for 'Java Enterprise Edition' and which is used to develop Server based

Applications.

Ex:

JavaEE includes

Servlet,JSP,WebServices

3.JavaME:

=>JavaME stands for 'Java Micro Edition' and which is used to develop Mobile Applications and Machine Application.

=>JavaME is also known as 'Mobile Edition' or 'Machine Edition'

---

faq:

define Server-based-Applications?

=>The Applications which are executed in server environment are known as Server-based Applications.

=>These Server based applications are categorized into two types:

1.Web Applications

2.Enterprise Applications

1.Web Applications:

=>The Server-based-applications which are constructed using JDBC,Servlet and JSP are known as Web Applications.

2.Enterprise Applications:

=>The applications which are executed in distributed environment and depending on the

features like 'Security','Load Balancing' and 'Clustering' are known as 'Enterprise applications' or 'Enterprise Distributed Applications'

Dt : 3/1/2025

**imp*

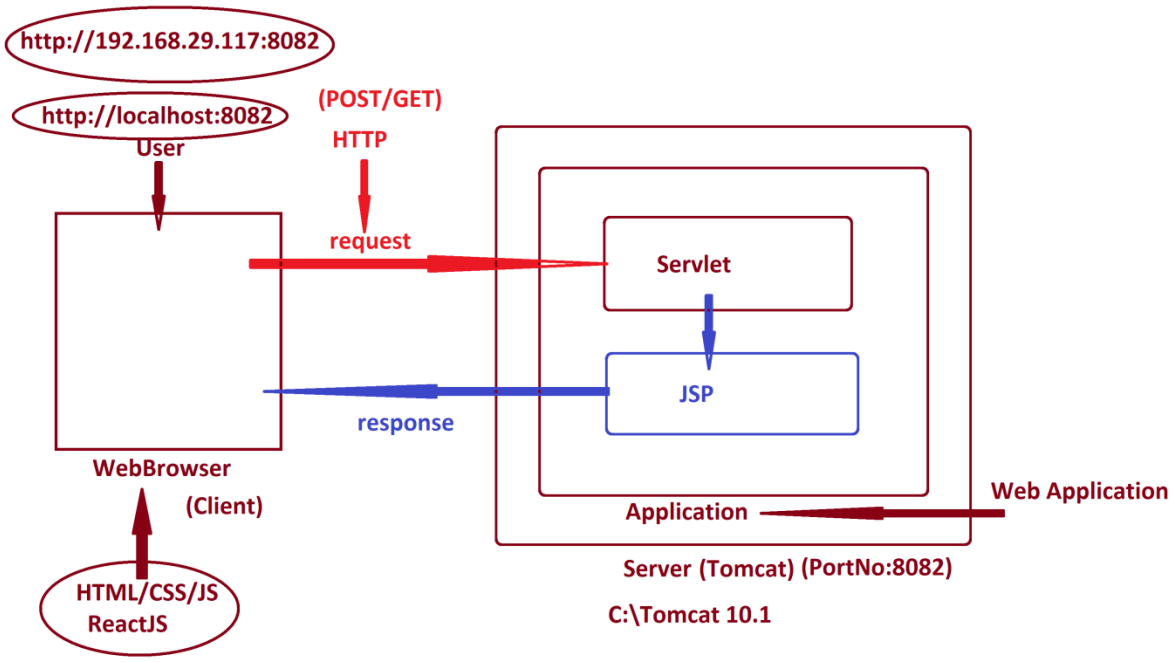
Servlet Programming:

faq:

define Servlet?

=>The PlatForm independent Java-Program which is executed in server environment and interacts with user through WebBrowser is known as Servlet Program or Server Program.

Diagram:



define Client?

=>Client means WebBrowser from where we raise request to server.

define Server?

=>Server means service-provider,which means accepts the request and provide the response.

=>Server will provide 'Web-Container' to execute Web-Applications.

=>According to realtime,servers are categorized into two types:

1.Web Servers

2.Application Servers

1.Web Servers:

=>Web Servers will provide only 'Web Container' to execute Web Applications.

=>Web Servers are preferable for static content data.(HTML)

=>Web Servers will accept the request from HTTP protocol.

Ex:

Tomcat

2.Application Servers:

=>Application Servers will provide both 'Web Container' and 'EJB Container' to execute Web Applications and Enterprise Applications.

(EJB - Enterprise Java Bean)

=>Application Servers are preferable for Dynamic content data.

=>Application Servers will accept the request from HTTP,RPC and RMI Protocols.

(RPC - Remote Procedure Call)

(RMI - Remote Method Invocation)

Ex:

WebLogic

WebSphere

---

*imp

Installing Tomcat Server:

step-1 : Download Tomcat10-version from APACHE Organization WebSite

Link:

https://tomcat.apache.org/download-10.cgi

step-2 : Install Tomcat Server

While Installation process,

Select the type of install : Full (Click Next)

Server Shutdown port : 8089

HTTP/1.1 Connector Port : 8081 or 8082 or 8083 or ...

User Name : V

Password : nit

(Click Next)

step-3 : Start the Tomcat Server

=>To start Tomcat server click 'startup' or 'Tomcat10w' file from 'bin' folder of Tomcat

C:\Tomcat 10.1\bin

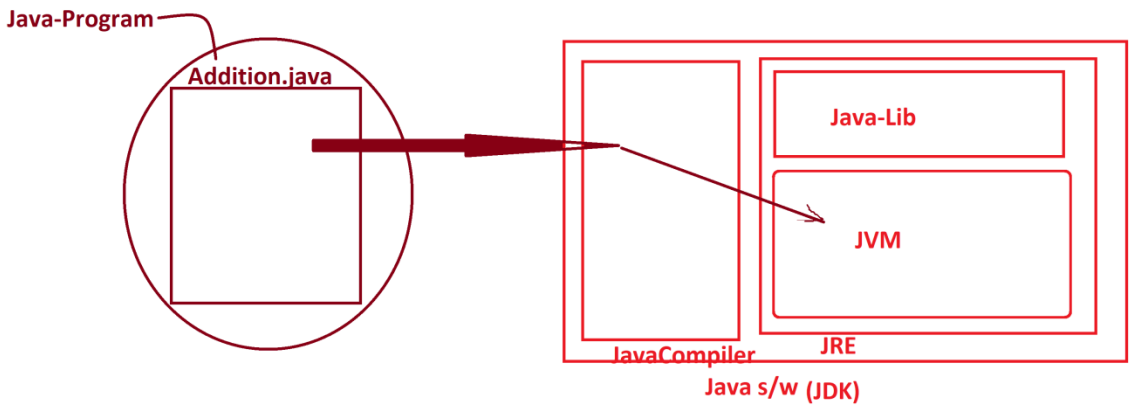
step-4 : Access the Tomcat Server from WebBrowser

http://localhost:8082

step-5 : Stop the Tomcat Server

=>To stop Tomcat server click 'shutdown' or 'Tomcat10w' file from 'bin' folder of Tomcat

C:\Tomcat 10.1\bin



Dt : 5/2/2025

**imp*

Servlet API:

=>'jakarta.servlet' package is known as Servlet-API,and which provide 'Classes and Interfaces' to develop Servlet-Application.

(From Tomcat10-version onwards the Java-Lib is referred by the word 'jakarta' and upto Tomcat9-version Java-Lib is reffered by the word 'javax')

=>'Servlet' interface from 'jakarta.servlet' package is the root of Servlet-API.

=>The following are some important methods of 'Servlet' interface:

1.init()

2.service()

3.destroy()

4.getServletConfig()

5.getServletInfo()

1.init():

=>init()-method is used to perform initialization process,which means making the programming components ready for service()-method.

Method Signature:

```
public abstract void init(jakarta.servlet.ServletConfig)
                    throws jakarta.servlet.ServletException;
```

2.service():

=>service()-method is used to accept the request and provide the response.

Method Signature:

public abstract void service(jakarta.servlet.ServletRequest,

jakarta.servlet.ServletResponse) throws jakarta.servlet.ServletException,

java.io.IOException;

3.destroy():

=>destroy()-method is used to close the resources and services which are opened while init() and service() methods.

Method Signature:

public abstract void destroy();

4.getServletConfig():

=>getServletConfig()-method is used to display the servlet-configuration details.

Method Signature:

public abstract jakarta.servlet.ServletConfig getServletConfig();

5.getServletInfo():

=>getServletInfo()-method will servlet information.

Method Signature:

public abstract java.lang.String getServletInfo();

Note:

=>The following three methods are known as Servlet Life-Cycle methods,because they are executed automatically in the same order:

(i)init()

(ii)service()

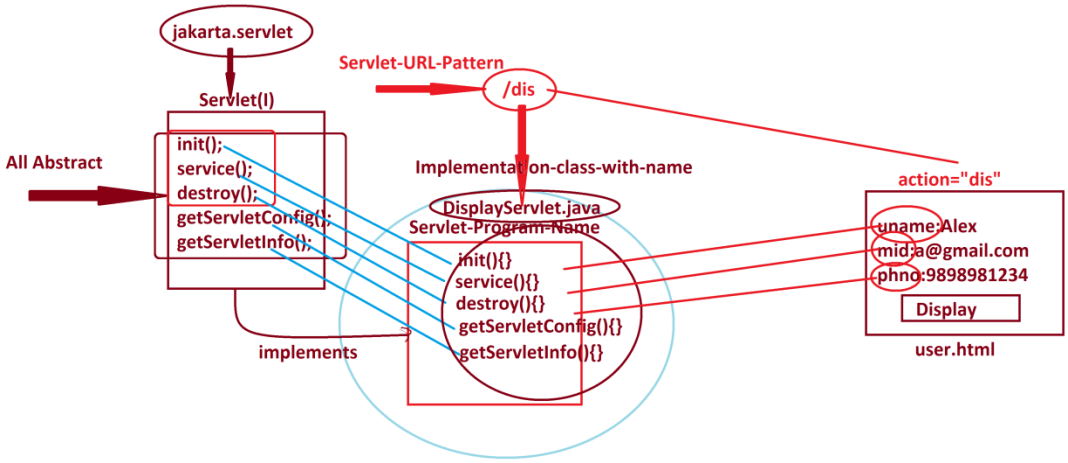
(iii)destroy()

Note:

=>In the process of constructing Servlet-Program,the Servlet-program must be implemented from 'Servlet' Interface and the Servlet-program must implement all abstract methods.

(Which means Construct bodies for all abstract methods of Servlet-Interface)

Diagram:



**imp*

Construct Servlet Application using IDE Eclipse:

step-1 : Open IDE Eclipse,while opening name the Workspace and click 'Launch'

step-2 : Create 'Dynamic Web Project'

Click on File->New->Project->Web->select 'Dynamic Web Project' and click 'Next'->

name the Project and Click 'Finish'

step-3 : Add 'servlet-api.jar' to Project through 'Build path'

RightClick on Dynamic Web Project->Build path->Configure Build Path->Libraries->

select 'Classpath' and click 'Add External JARs'->Browse and select 'servlet-api.jar' file

from 'lib' folder of Tomcat->Open->Apply->Apply and Close

step-4 : Add Web Server(Tomcat) to IDE Eclipse(one time process)

Click on Servers->click on 'Click this link to create a New Server'->select the server and

click 'Next'->Browse 'Tomcat Installation Directory'->select Folder->click 'Finish'

step-5 : Construct HTML file to read data to Servlet program

RightClick on webapp->New->HTML file->name the file->click 'Finish'

user.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="dis" method="post">
UserName:<input type="text" name="uname"><br>
MailId:<input type="text" name="mid"><br>
```

*PhoneNO:<input type="text" name="phno">
*

<input type="submit" value="Display">

</form>

</body>

</html>

step-6 : Construct 'web.xml' mapping file in 'WEB-INF'

RightClick on 'WEB-INF'->New->Other->XML->XML file->name the file as "web.xml" and click 'Finish'

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <welcome-file-list>
    <welcome-file>
        user.html
    </welcome-file>
  </welcome-file-list>
</web-app>
```

step-7 : create package in "src/main/java" of Java Resources.

step-8 : Create class(Servlet program) to display user details.

DisplayServlet.java

```
package test;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
@WebServlet("/dis")
public class DisplayServlet implements Servlet
{
	@Override
	public void init(ServletConfig scf)throws ServletException
	{
		//NoCode
	}
	@Override
	public void service(ServletRequest req,ServletResponse res)
			throws ServletException,IOException
	{
		String uName = req.getParameter("uname");
		String mId = req.getParameter("mid");
		long phNo = Long.parseLong(req.getParameter("phno"));
		PrintWriter pw = res.getWriter();
		res.setContentType("text/html");
		pw.println("******User Details*****<br>");
		pw.println("UserName:"+uName+"<br>");
```

```
            pw.println("MailId:"+mId+"<br>");
            pw.println("PhoneNo:"+phNo+"<br>");
        }
        @Override
        public void destroy()
        {
            //NoCode
        }
        @Override
        public ServletConfig getServletConfig()
        {
            return this.getServletConfig();
        }
        @Override
        public String getServletInfo()
        {
            return "This Servlet will display User details";
        }
}
```

step-9 : Execute the Application

RightClick on Application(Dynamic Web Project)->Run As->Run on Server->Select the Server->

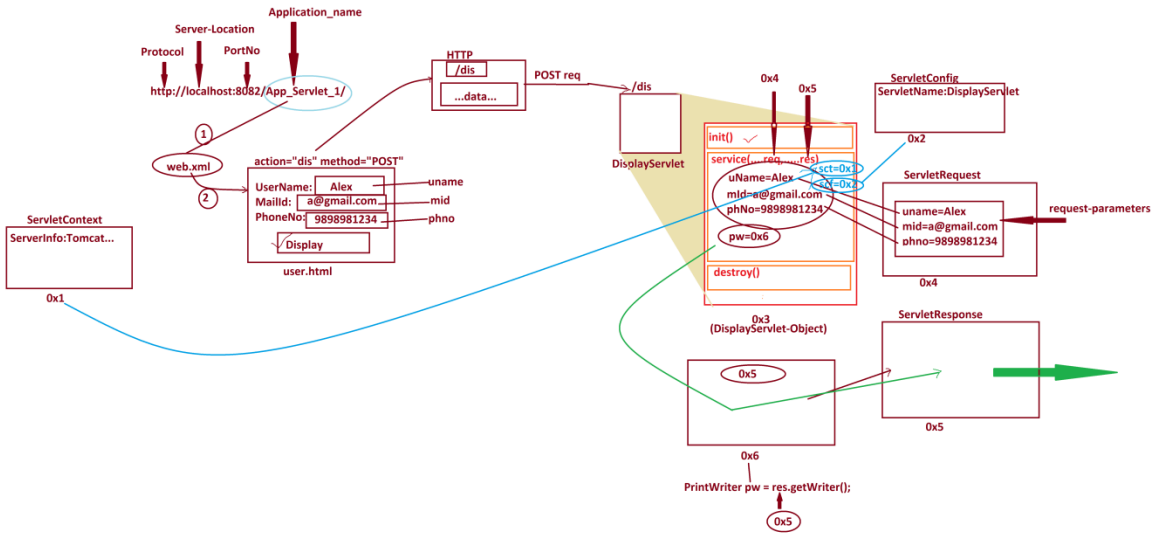
Click 'Finish'

http://localhost:8082/App_Servlet_1/

==========

Dt : 6/2/2025

Diagram:(Demonstrating Execution flow of Servlet Application)



ServletContext:

=>'ServletContext' is an interface from 'jakarta.servlet' package and which is instantiated automatically when the Web-App is deployed into Server.

=>This ServletContext-Object is recored with Server-Information.

=>we use getServletContext()-method from ServletRequest to access the reference of ServletContext-Object.

Method Signature:

public abstract jakarta.servlet.ServletContext getServletContext();

syntax and Ex:

ServletContext sct = req.getServletContext();

ServletConfig:

=>'ServletConfig' is an interface from 'jakarta.servlet' package and which is instantiated automatically when the Servlet-Program loaded for execution.

=>This ServletConfig-Object is recorded with Servlet-name.

=>we use getServletConfig()-method to access the reference of ServletConfig-Object

Method Signature:

public abstract jakarta.servlet.ServletConfig getServletConfig();

syntax and Ex:

ServletConfig scf = this.getServletConfig();

Note:

=>After ServletConfig-Object creation,the Servlet-program will be instantiated automatically.

=>After Servlet-Program instantiation,the execution-controls will identify the following Life-Cycle methods and they are executed automatically:

(i)init()

(ii)service()

(iii)destroy()

ServletRequest:

=>'ServletRequest' is an interface from 'jakarta.servlet' package and which is instantiated automatically while service()-method execution.

=>This ServletRequest-Object will hold the data submitted from HTML-form.

ServletResponse:

=>'ServletResponse' is an interface from 'jakarta.servlet' package and which is also

instantiated automatically while service()-method execution.

=>ServletResponse-Object will hold the data which we are sending as response.

faq:

define getParameter()-method?

=>getParameter()-method is from ServletRequest and which is used to access para-value from request object.

Method Signature:

public abstract java.lang.String getParameter(java.lang.String);

syntax and ex:

String uName = req.getParameter("uname");

faq:

define getWriter()-method?

=>getWriter()-method is from ServletResponse and which is used to create Object for 'java.io.PrintWriter' Class.

=>This 'PrintWriter' object will hold the reference of ServletResponse Object .

Method Signature of getWriter():

public abstract java.io.PrintWriter getWriter() throws java.io.IOException;

syntax and Ex:

PrintWriter pw = res.getWriter();

faq:

define setContentType()-method?

=>setContentType()-method is from ServletResponse and which specify the type of data we are adding to the response.

Method Signature:

public abstract void setContentType(java.lang.String);

syntax:

res.setContentType("text/html");

Assignment-1:

Construct Servlet Application to read and display Product details.

(Input using HTML form)

Assignment-2:

Construct Servlet Application to read and display BookDetails.

(Input Using HTML form)

Dt : 7/2/2025

faq:

Servlet Life-Cycle:

=>Servlet Life-Cycle demonstrates different states or stages of Servlet-Program from Starting to ending.

=>The following are some important stages of Servlet-Program:

1.Loading process

2.Instantiation process

3.Initialization process

4.Request Handling Process

5.Destroying Process

1.Loading process:

=>The process of identifying the Servlet-program based on url-pattern and loading for execution is known as Loading Process.

2.Instantiation process:

=>The process in which the Servlet-program automatically instantiated is known as Instantiation process.

Note:

=>After Instantiation process,execution-controls will identify the following LifeCycle methods:

(i)init()

(ii)service()

(iii)destroy()

3.Initialization process:

=>The process of making the programming components(Bean Objects,DAO Objects and Services) ready for service()-method is known as Initialization process.

=>we use init()-method to perform Initialization process.

4.Request Handling Process:

=>The process of accepting the request and providing the response,is known as Request Handling Process.

=>we use service()-method to perform Request Handling Process.

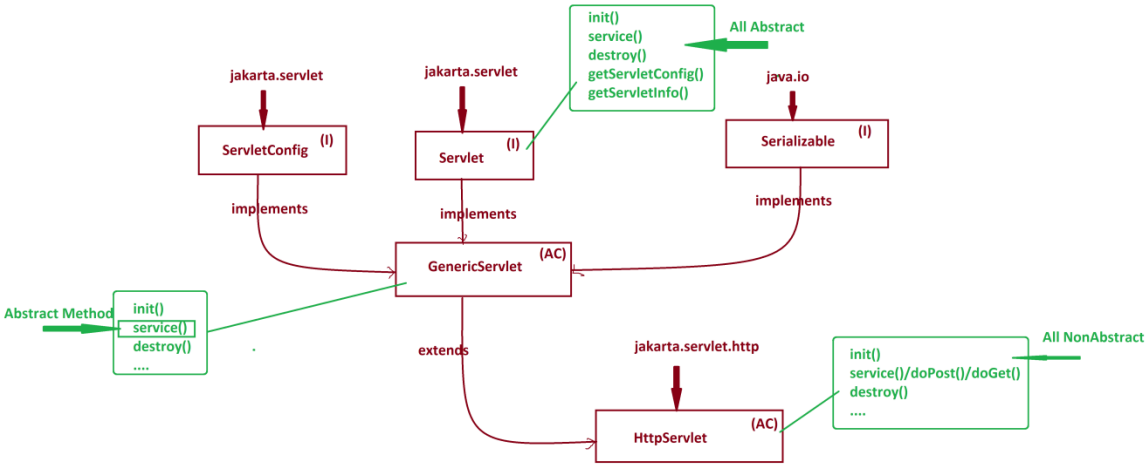
5.Destroying Process:

=>The process of closing the opening Components and services is known as Destroying process.

=>we use destroy()-method to perform Destroying process.

****imp***

Hierarchy of Servlet-API:



=>In the process of constructing Servlet-Program,we use any one of the following:

Model-1 : Implementing from 'Servlet-Interface'

Model-2 : Extending from 'GenericServlet-AbstractClass'

Model-3 : Extending from 'HttpServlet-AbstractClass'

Model-1 : Implementing from 'Servlet-Interface'

=>when the Servlet-program implemented from 'Servlet-Interface',then we have to construct body for all abstract methods.

Model-2 : Extending from 'GenericServlet-AbstractClass'

=>when the Servlet-Program extended from 'GenericServlet-AbstractClass',then we must construct body for only service()-method and 'init() and destroy()' are optional

Model-3 : Extending from 'HttpServlet-AbstractClass':

=>when the Servlet-program extended from 'HttpServlet-AbstractClass',then all methods are optional methods.

init()

service()/doPost()/doGet()

destroy()

Note:

service() : method will accept both POST and GET requests

doPost() : method will accept only POST request

doGet() : method will accept only GET request

Note:

=>The Object generated from Servlet-Interface is NonSerializable Object

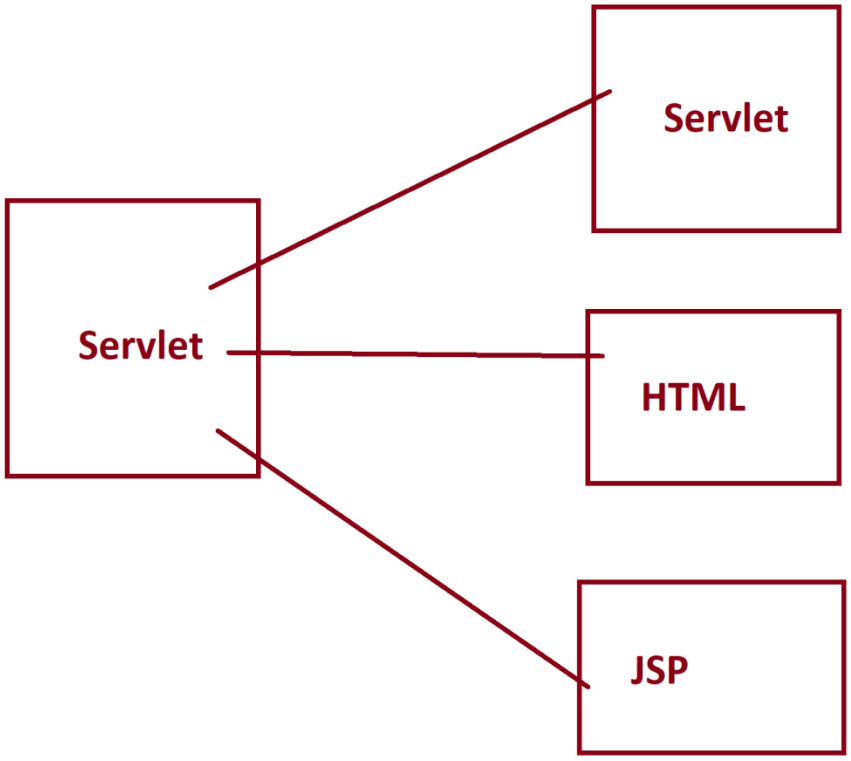
=>The Objects generated from GenericServlet and HttpServlet are Serializable Objects

==========

**imp*

'RequestDispatcher' in Servlet Programming:

=>RequestDispatcher is an interface from jakarta.servlet package and which is used for Servlet Communication like Servlet-Servlet Communication,Servlet-HTML Communication and Servlet-JSP Communication.



=>Servlet Communications are categorized into two types:

1.Forward Communication Process

2.Include Communication Process

1.Forward Communication Process:

=>In Forward Communication Process,the Servlet-1 will take the request and forwards the the request Servlet-2,in this process Servlet-2 will provide the response.

=>This Servlet-2 can be replaced with HTML/JSP.

=>we use forward()-method from 'RequestDispatcher' to perform Forward Communication

process.

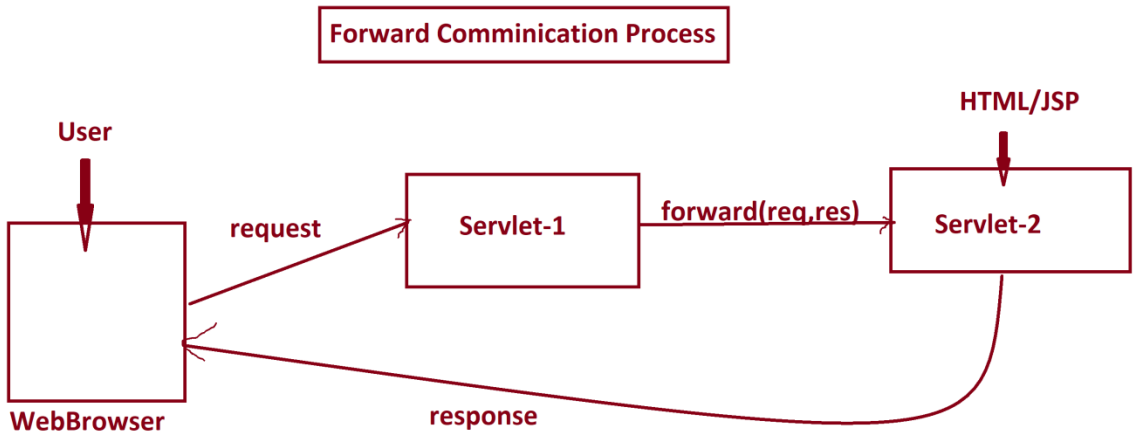
Method Signature:

public abstract void forward(jakarta.servlet.ServletRequest,

jakarta.servlet.ServletResponse) throws jakarta.servlet.ServletException,

java.io.IOException;

Diagram:



2.Include Communication Process:

=>In Include Communication Process,Servlet-1 will take the request and generate the response,but the response is included with the response of Servlet-2

=>This Servlet-2 can be replaced with HTML/JSP

=>we use include()-method from 'RequestDispatcher' to perform Include Communication process.

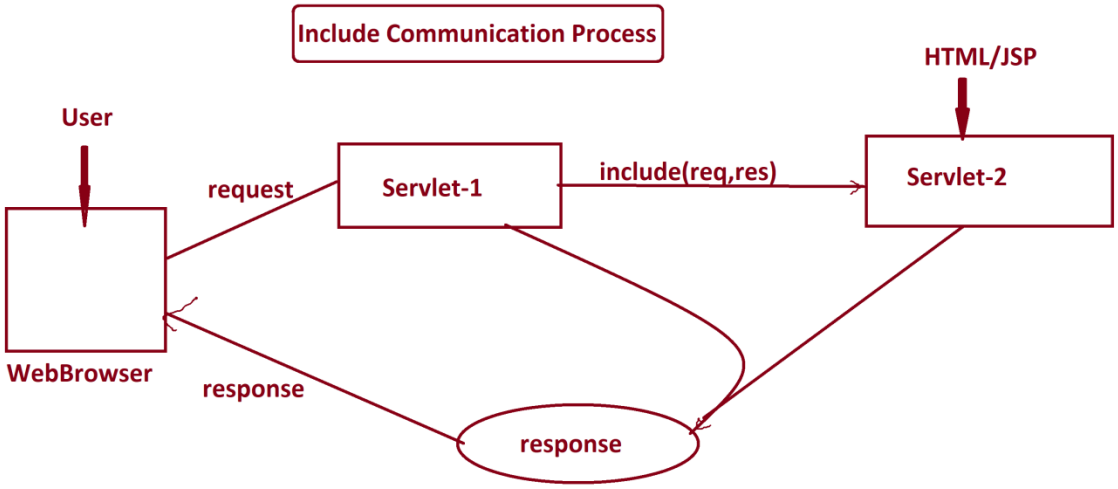
Method Signature:

public abstract void include(jakarta.servlet.ServletRequest,

jakarta.servlet.ServletResponse) throws jakarta.servlet.ServletException,

java.io.IOException;

Diagram:



=>we use getRequestDispatcher()-method from 'ServletRequest' to create implementation

Object for 'RequestDispatcher-Interface'

Method Signature:

public abstract jakarta.servlet.RequestDispatcher getRequestDispatcher(java.lang.String);

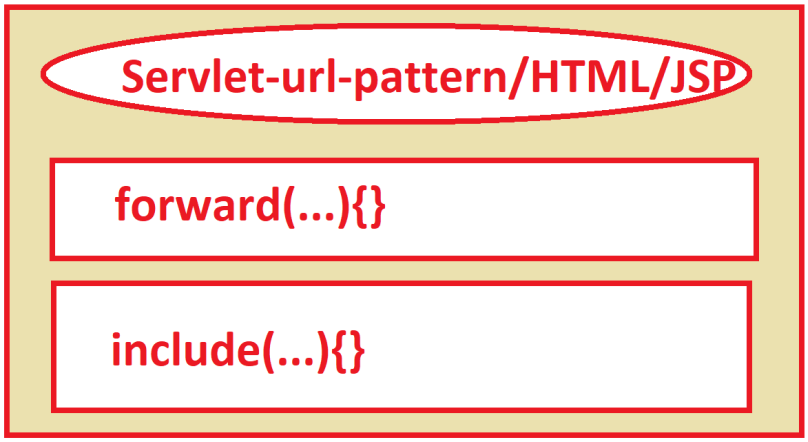
syntax:

RequestDispatcher rd = req.getRequestDispatcher("Servlet-url-pattern/HTML/JSP");

rd.forward(req,res);

rd.include(req,res);

Diagram:



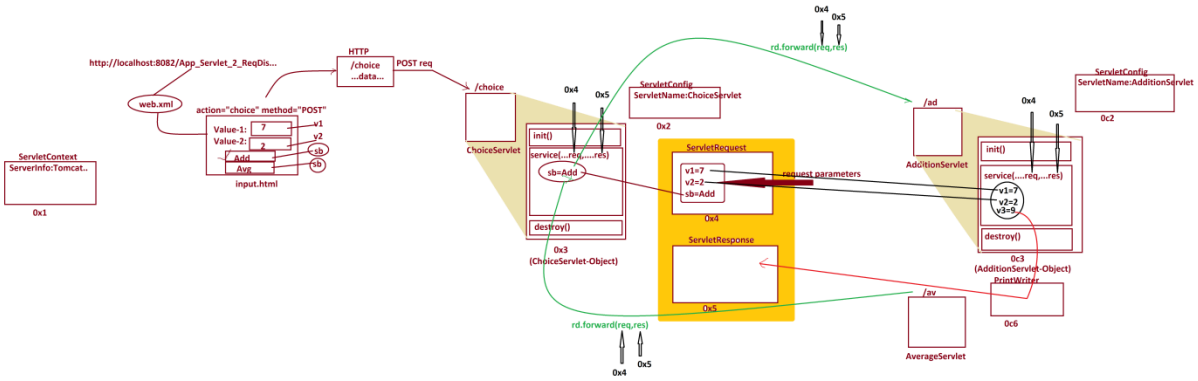
0x21

RequestDispatcher rd = req.getRequestDispatcher ("Servlet-url-pattern/HTML/JSP");

Dt : 10/2/2025

Application:(Demonstrating 'RequestDispatcher')

Layout:



input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="choice" method="post">
Enter the Value-1:<input type="text" name="v1"><br>
Enter the Value-2:<input type="text" name="v2"><br>
<input type="submit" value="Add" name="sb">
<input type="submit" value="Avg" name="sb">
</form>
</body>
</html>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app>
  <welcome-file-list>
     <welcome-file>input.html</welcome-file>
  </welcome-file-list>
</web-app>
```

ChoiceServlet.java

```
package test;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/choice")
public class ChoiceServlet extends GenericServlet
{
        @Override
 public void service(ServletRequest req,ServletResponse res)throws ServletException,
 IOException
 {
        String sb = req.getParameter("sb");
        if(sb.equals("Add")) {
                RequestDispatcher rd = req.getRequestDispatcher("ad");
                rd.forward(req, res);
        }else {
                RequestDispatcher rd = req.getRequestDispatcher("av");
                rd.forward(req, res);
        }
 }
}
```

AdditionServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/ad")

public class AdditionServlet extends GenericServlet

{

        @Override

  public void service(ServletRequest req,ServletResponse res)throws ServletException,

  IOException

  {

                PrintWriter pw = res.getWriter();

                res.setContentType("text/html");

                try {

                        int v1 = Integer.parseInt(req.getParameter("v1"));

                         int v2 = Integer.parseInt(req.getParameter("v2"));

                        int v3 = v1+v2;

                        pw.println("Sum:"+v3+"<br>");

                }catch(Exception e) {

                        pw.println("Enter Only Integer Values....<br>");

                }

                  RequestDispatcher rd = req.getRequestDispatcher("input.html");

                  rd.include(req, res);

  }
```

```
}
```

AverageServlet.java

```
package test;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/av")
public class AverageServlet extends GenericServlet
{
        @Override
  public void service(ServletRequest req,ServletResponse res)throws ServletException,
  IOException
  {
                PrintWriter pw = res.getWriter();
                res.setContentType("text/html");
                try {
                    int v1 = Integer.parseInt(req.getParameter("v1"));
                     int v2 = Integer.parseInt(req.getParameter("v2"));
                      float v3 = (float)(v1+v2)/2;
                      pw.println("Avg:"+v3+"<br>");
                }catch(Exception e) {
                        pw.println("Enter only Integer values...<br>");
                }
```

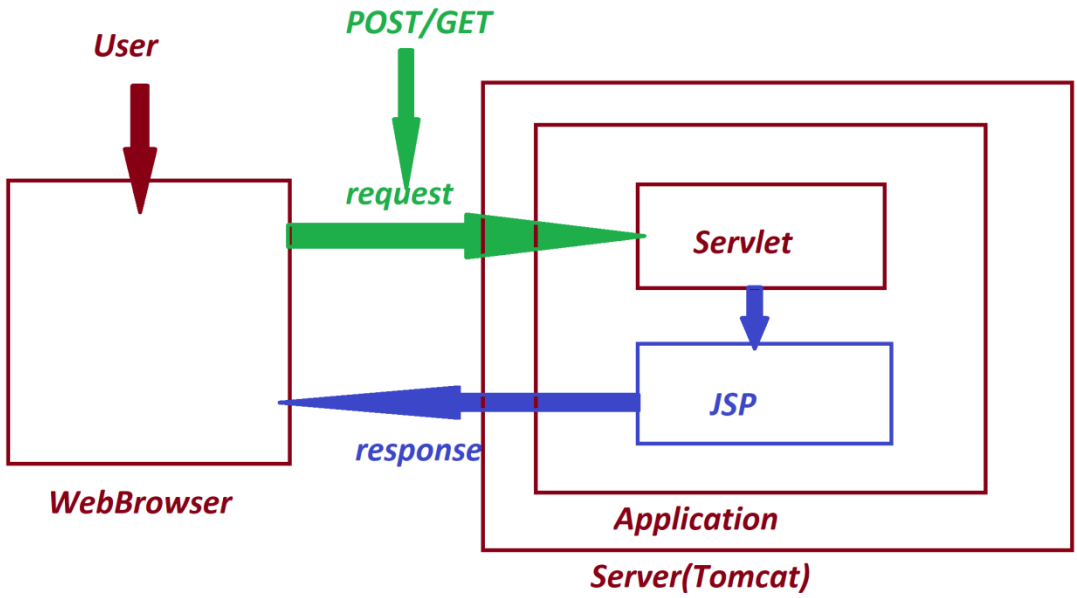
```
            RequestDispatcher rd = req.getRequestDispatcher("input.html");
            rd.include(req, res);
    }
}
```

Assignment:

Update above application with the following operations on two integer values:

- ***=>Sub***
- ***=>Mul***
- ***=>Div***
- ***=>ModDiv***
- ***=>Greater***
- ***=>Smaller***

Dt : 11/2/2025



**imp*

JSP Programming:

=>JSP stands for 'Java Server Page' and which is response from Web-Application.

=>JSP is tag-based-programming language and which is more simple when compared to Servlet Programming.

=>JSP program is combination of both HTML code and JavaCode(ServletCode)

=>JSP programs are saved with (.jsp) as extention.

Ex:

Read.jsp

=>We use the following JSP tags to construct JSP programs:

1.Scripting tags

2.Directive tags

3.Action tags.

1.Scripting tags:

=>Scripting tags are used to write noraml Java Code in jsp programs.

=>Types:

(a)Scriptlet tag

(b)Expression tag

(c)Declarative tag

2.Directive tags:

=>Directive tags will specify the information about current running JSP page(Program)

=>Types:

(a)@page

(b)@include

(c)@taglib

3.Action tags.:

=>Action tags will specify the actions performed while JSP Program execution.

=>Types:

(a)<jsp:forward>

(b)<jsp:include>

(c)<jsp:param>

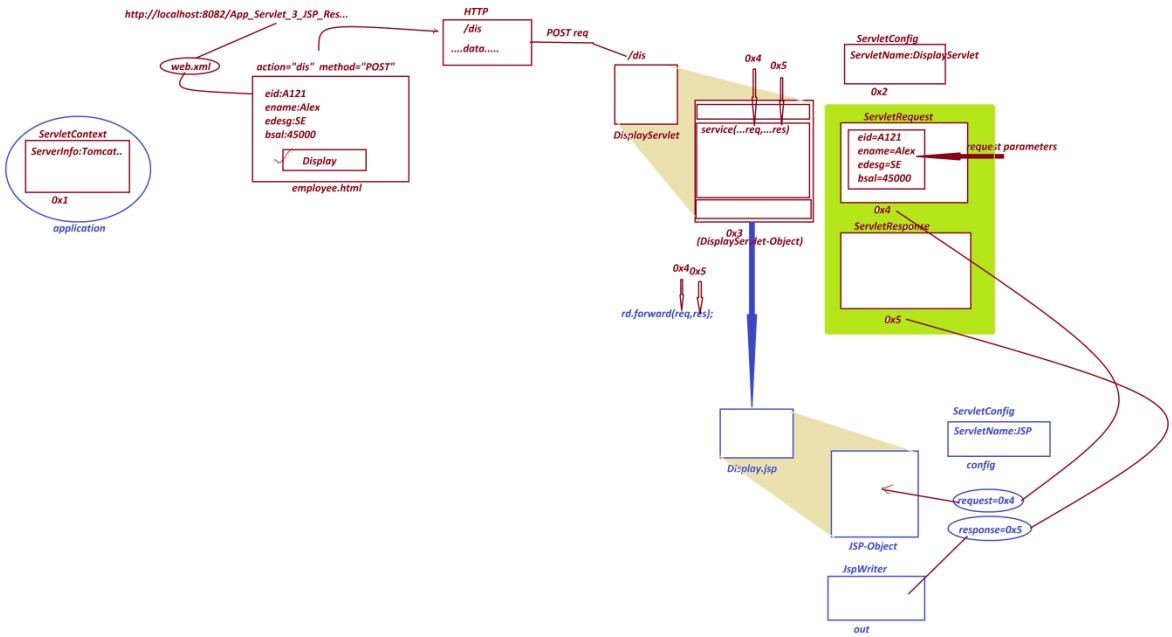
(d)<jsp:useBean>

(e)<jsp:setProperty>

(f)<jsp:getProperty>

Ex:(Construct Servlet Application to demonstrate JSP-response)

Layout:



employee.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="dis" method="post">
Employee-Id:<input type="text" name="eid"><br>
Employee-Name:<input type="text" name="ename"><br>
Employee-Desg:<input type="text" name="edesg"><br>
Employee-BSal:<input type="text" name="bsal"><br>
<input type="submit" value="Display">
</form>
</body>
```

```
</html>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <welcome-file-list>
     <welcome-file>employee.html</welcome-file>
  </welcome-file-list>
</web-app>
```

DisplayServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/dis")

public class DisplayServlet extends GenericServlet

{

        @Override

 public void service(ServletRequest req,ServletResponse res)throws ServletException,

 IOException

 {

       RequestDispatcher rd = req.getRequestDispatcher("Display.jsp");

        rd.forward(req, res);

 }

}
```

Display.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
```

```
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String eId = request.getParameter("eid");
String eName = request.getParameter("ename");
String eDesg = request.getParameter("edesg");
int bSal = Integer.parseInt(request.getParameter("bsal"));
float hra = 0.93F*bSal;
float da = 0.63F*bSal;
float totSal = bSal+hra+da;
out.println("Emp-Id:"+eId+"<br>");
out.println("Emp-Name:"+eName+"<br>");
out.println("Emp-Desg:"+eDesg+"<br>");
out.println("Emp-BSal:"+bSal+"<br>");
out.println("Emp-HRA:"+hra+"<br>");
out.println("Emp-DA:"+da+"<br>");
out.println("Emp-TotSal:"+totSal+"<br>");
%>
<%@include file="employee.html" %>
</body>
</html>
```

Assignment:

Construct Servlet Application to read and display User-registration details.

(JSP-response)

uname

pword

fname

lname

city

state

pincode

mid

phno

---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------

Dt : 12/2/2025

Note:

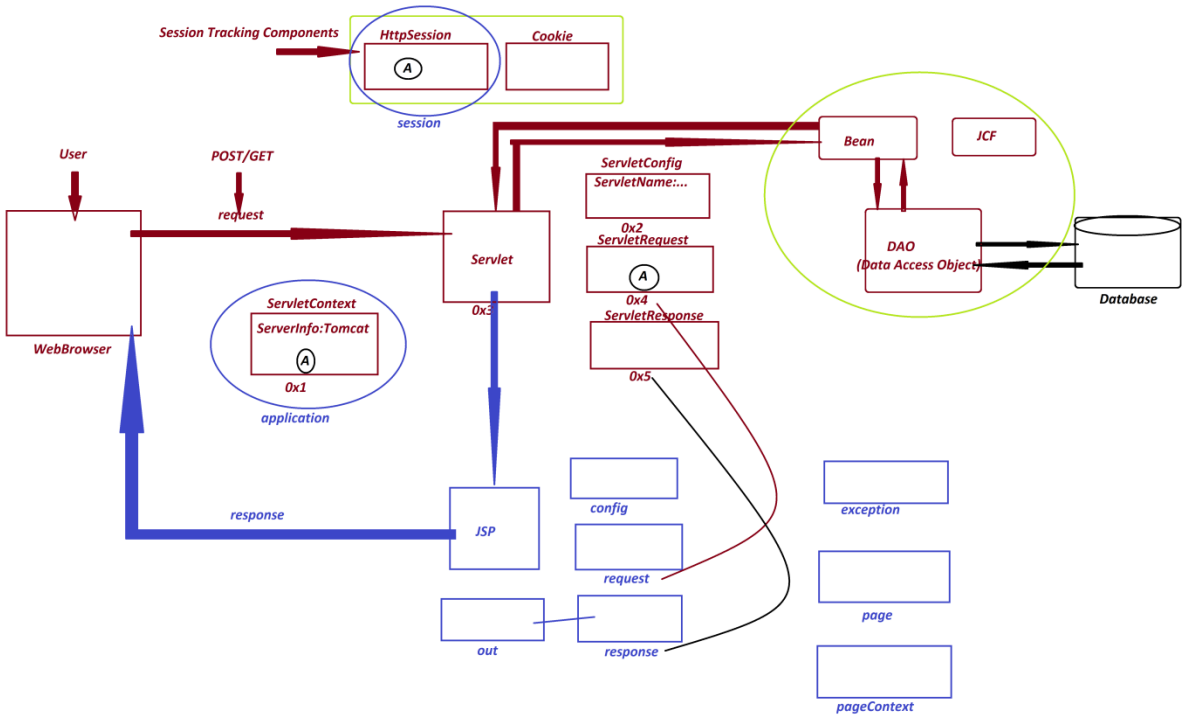
=>In Servlet forward communication,the request and response objects of Servlet-1 can be used by Servlet-2 or JSP-Program

=>WebContainer internally divided into the following two Sub-Containers:

(a)Servlet Container - Catelina

(b)JSP Container - Jasper

Complete Layout of Objects from Servlet and JSP:



---

faq:

define DAO?

=>DAO stands for 'Data Access Object' and which is separate layer in MVC

(Model View Controller) to hold database related codes(Persistent Logics).

=>DAO Layer will hold DB-Connection,DB-Insert,DB-Retrieve,DB-Update,DB-Delete,...

(All Database Operations)

=>In the process of establishing Communication b/w Servlet-Program and Database-Product,

the DB-JAR file must be copied into "lib" folder of 'WEB-INF'.

---

faq:

define Bean Class?

=>The class which is constructed with the following rules is known as Bean-Class:

Rule-1 : The class must be implemented from 'java.io.Serializable' interface.

Rule-2 : The variables which are declared within the class must be 'Private' variables.

Rule-3 : The class must be declared with 0-argument Constructor or 0-parameter Constructor

Rule-4 : The class must be declared with 'Getter' and 'Setter' methods.

Note:

=>These Bean-Classes will generate Bean-Objects.

=>Bean-Objects are the intermediate storages b/w Servlet-Program and Database Product.

**imp*

"attribute" in Servlet Programming:

=>"attribute" is a variable in Servlet Programming,which can be added to ServletContext Object,ServletRequest Object and HttpSession Object.

=>The following are the methods related to "attribute":

(a)setAttribute()

(b)getAttribute()

(c)removeAttribute()

(d)getAttributeNames()

(a)setAttribute():

=>setAttribute()-method is used to add attribute to Objects.

Method Signature:

public abstract void setAttribute(java.lang.String, java.lang.Object);

(b)getAttribute():

=>getAttribute()-method is used to get attribute from Objects.

Method Signature:

public abstract java.lang.Object getAttribute(java.lang.String);

(c)removeAttribute():

=>removeAttribute()-method will delete the attribute from the Objects.

Method Signature:

public abstract void removeAttribute(java.lang.String);

(d)getAttributeNames():

=>getAttributeNames()-method is used to get all attributes names from the Objects.

Method Signature:

public abstract java.util.Enumeration<java.lang.String> getAttributeNames();

Dt : 13/2/2025

faq:

define request?

=>The query which is raised by the user through WebBrowser is known as 'request'

=>'request' is categorized into two types:

1.POST request

2.GET request

1.POST request:

=>The request which is raised to send the data to server is known as POST request.

=>Through POST request we can send all types of data,which means we can send Text,Audio, Video,Image and Animation.

=>Through POST request we can send UnLimited data.

=>The data in POST request is secure,because the data is encapsulated into the body of HTTP.

=>To raise POST request,we write method="POST" in <form> tag.

syntax:

```
<form action="url" method="POST">
    ...
</form>
```

=>we use doPost()-method from 'HttpServlet' to accept POST request.

Method Signature of doPost():

```
protected void doPost(jakarta.servlet.http.HttpServletRequest,
 jakarta.servlet.http.HttpServletResponse) throws jakarta.servlet.ServletException,
            java.io.IOException;
```

2.GET request:

=>The request which is raised to get data from Server is known as GET request.

=>Through GET request we can send only Text data.

=>Through GET request we can send max up to 10KB data(Limited data)

=>The data in GET request is not secure,because the data is dispayed in address-bar directly.

=>We use the following syntaxes to raise GET request:

syntax-1 : we use method="GET" in <form> tag

```
<form action="url" method="GET">
 ...
</form>
```

syntax-2 : If we construct <form> tag without any method specificaation will raise GET request

```
<form action="url">
 ...
</form>
```

syntax-3 : GET request is raised through hyperlinks.

syntax-4 : The GET request can be raised through address-bar,by using servlet-url-pattern

=>We use doGet()-method from 'HttpServlet' to accept GET request.

Method Signature of doGet():

protected void doGet(jakarta.servlet.http.HttpServletRequest,

jakarta.servlet.http.HttpServletResponse) throws jakarta.servlet.ServletException,

java.io.IOException;

---

faq:

define Session?

=>The time interval b/w User-Login to User-Logout is known as Session.

faq:

define Session Management?

=>The process of recording the state-of-user and tracking the used in session,is known as Session Management.

Types of Session Tracking Techniques used in Session Management:

1.HttpSession

2.URL re-write

3.Hidden Form Fields

4.Cookie

1.HttpSession:

=>'HttpSession' is an interface from jakarta.servlet.http package and,which is instantiated and used in Session Tracking process.

=>The following are some important methods of HttpSession:

public abstract void setAttribute(java.lang.String, java.lang.Object);

public abstract java.lang.Object getAttribute(java.lang.String);

public abstract void removeAttribute(java.lang.String);

public abstract java.util.Enumeration<java.lang.String> getAttributeNames();

public abstract void invalidate();

public abstract jakarta.servlet.ServletContext getServletContext();

Note:

=>we use getSession()-method from 'HttpServletRequest' to create implementation Object for HttpSession-Interface.

Method Signatures of getSession():

public abstract jakarta.servlet.http.HttpSession getSession();

public abstract jakarta.servlet.http.HttpSession getSession(boolean);

==========

Application : ProductStore

DBTables:

Product70(code,name,price,qty)

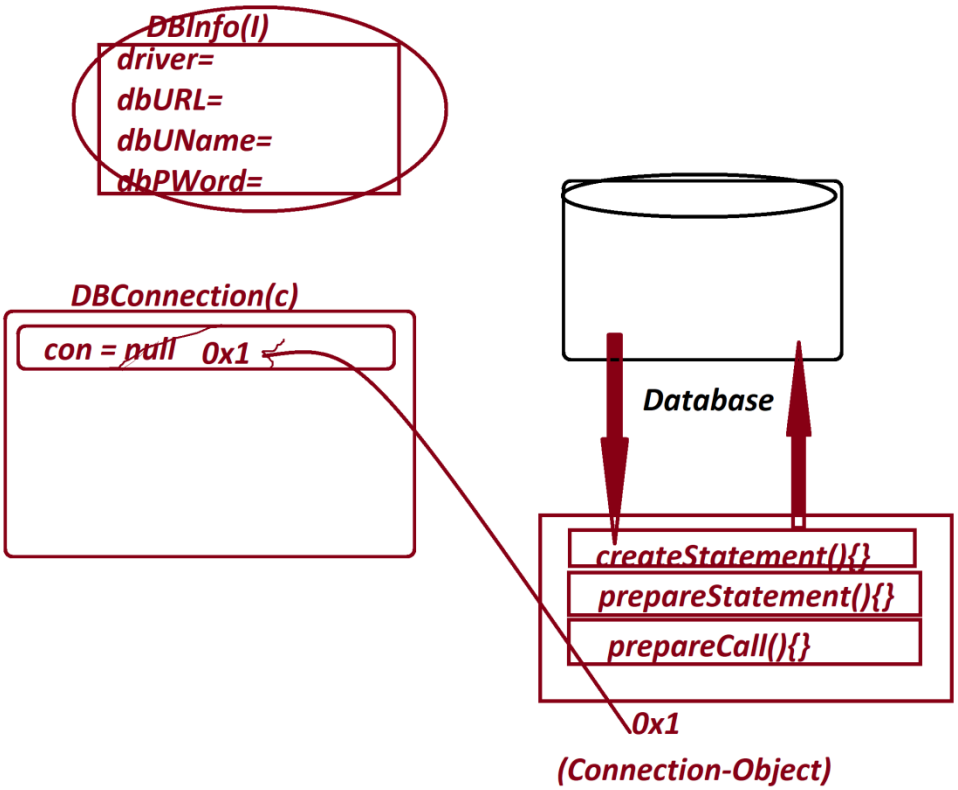
Primary Key : code

Admin70(uname,pword,fname,lname,city,mid,phno)

Primary Key : uname and pword

```
create table Admin70(uname varchar2(15),pword varchar2(15),fname varchar2(15),
lname varchar2(15),city varchar2(15),mid varchar2(25),phno number(15),
primary key(uname,pword));

insert into Admin70 values('nit.v','mzu672','V','M','Hyd','v@gmail.com',9898981234);
```

Layout:



DBInfo.java(Interface)

```
package test;
public interface DBInfo
{
   public static final String driver="oracle.jdbc.driver.OracleDriver";
   public static final String dbURL="jdbc:oracle:thin:@localhost:1521:xe";
   public static final String dbUName="system";
   public static final String dbPWord="tiger";
}
```

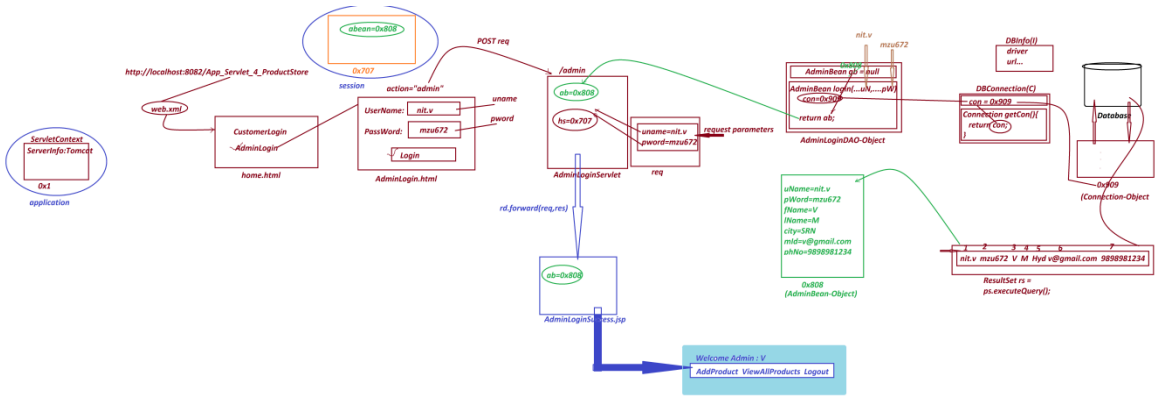
DBConnection.java

```
package test;
import java.sql.*;
public class DBConnection
```

```
{
	private static Connection con = null;
	private DBConnection() {}
	static
	{
	  try {
		Class.forName(DBInfo.driver);
		con = DriverManager.getConnection
				(DBInfo.dbUName,DBInfo.dbUName,DBInfo.dbPWord);
	  }catch(Exception e) {
		e.printStackTrace();
	  }
	}//end of block
	public static Connection getCon()
	{
	  return con;
	}
}
```

Dt : 15/2/2025

Layout:



home.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<a href="CustomerLogin.html">CustomerLogin</a>
<a href="AdminLogin.html">AdminLogin</a>
</body>
</html>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <welcome-file-list>
      <welcome-file>home.html</welcome-file>
  </welcome-file-list>
</web-app>
```

AdminLogin.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
```

```
</head>
<body>
<form action="admin" method="post">
UserName:<input type="text" name="uname"><br>
Password:<input type="password" name="pword"><br>
<input type="submit" value="Login">
</form>
</body>
</html>
```

AdminBean.java

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class AdminBean implements Serializable
{
   private String uName,pWord,fName,lName,city,mId;
   private long phNo;
   public AdminBean() {}
public String getuName() {
      return uName;
}
public void setuName(String uName) {
      this.uName = uName;
}
public String getpWord() {
      return pWord;
}
public void setpWord(String pWord) {
      this.pWord = pWord;
}
public String getfName() {
      return fName;
}
public void setfName(String fName) {
      this.fName = fName;
}
public String getlName() {
      return lName;
}
public void setlName(String lName) {
      this.lName = lName;
}
public String getCity() {
      return city;
}
public void setCity(String city) {
      this.city = city;
```

```
}
public String getmId() {
	return mId;
}
public void setmId(String mId) {
	this.mId = mId;
}
public long getPhNo() {
	return phNo;
}
public void setPhNo(long phNo) {
	this.phNo = phNo;
}

}
```

AdminLoginDAO.java

```
package test;
import java.sql.*;
public class AdminLoginDAO
{
    public AdminBean ab = null;
    public AdminBean login(String uN,String pW)
    {
      try {
            Connection con = DBConnection.getCon();//Accessing database
connection
            PreparedStatement ps = con.prepareStatement
                        ("select * from Admin70 where uname=? and pword=?");
            ps.setString(1, uN);
            ps.setString(2, pW);
            ResultSet rs = ps.executeQuery();
            if(rs.next()) {
                  ab = new AdminBean();
                  ab.setuName(rs.getString(1));
                  ab.setpWord(rs.getString(2));
                  ab.setfName(rs.getString(3));
                  ab.setlName(rs.getString(4));
                  ab.setCity(rs.getString(5));
                  ab.setmId(rs.getString(6));
                  ab.setPhNo(rs.getLong(7));
            }
      }catch(Exception e) {
            e.printStackTrace();
      }
      return ab;
    }
}
```

AdminLoginServlet.java

```
package test;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/admin")
public class AdminLoginServlet extends HttpServlet
{
        @Override
  protected void doPost(HttpServletRequest req,HttpServletResponse res)throws
  ServletException,IOException
  {
            String uN = req.getParameter("uname");
            String pW = req.getParameter("pword");
            AdminBean ab = new AdminLoginDAO().login(uN, pW);
            if(ab==null) {

            }else {
                  HttpSession hs = req.getSession();//New Session Created
                  hs.setAttribute("abean", ab);
                  RequestDispatcher rd = req.getRequestDispatcher("AdminLoginSuccess.jsp");
                  rd.forward(req, res);
```

```
        }

  }

}
```

AdminLoginSuccess.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.AdminBean"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
AdminBean ab = (AdminBean)session.getAttribute("abean");
out.println("Welcome Admin : "+ab.getfName()+"<br>");
%>
<a href="Product.html">AddProduct</a>
<a href="view">ViewAllProducts</a>
<a href="logout">Logout</a>
</body>
</html>
```

==========

Dt : 17/2/2025

Product.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="add" method="post">
ProductCode:<input type="text" name="pcode"><br>
ProductName:<input type="text" name="pname"><br>
ProductPrice:<input type="text" name="pprice"><br>
ProductQty:<input type="text" name="pqty"><br>
<input type="submit" value="AddProduct">
</form>
</body>
</html>
```

ProductBean.java

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class ProductBean implements Serializable
{
  private String code,name;
  private float price;
  private int qty;
  public ProductBean() {}
public String getCode() {
      return code;
}
public void setCode(String code) {
      this.code = code;
}
public String getName() {
      return name;
}
public void setName(String name) {
      this.name = name;
}
public float getPrice() {
      return price;
}
public void setPrice(float price) {
      this.price = price;
```

```
}
public int getQty() {
	return qty;
}
public void setQty(int qty) {
	this.qty = qty;
}

}
```

AddProductDAO.java

```
package test;
import java.sql.*;
public class AddProductDAO
{
    public int k=0;
    public int insert(ProductBean pb)
    {
      try {
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                        ("insert into Product70 values(?,?,?,?)");
            ps.setString(1, pb.getCode());
            ps.setString(2, pb.getName());
            ps.setFloat(3, pb.getPrice());
            ps.setInt(4, pb.getQty());
            k = ps.executeUpdate();
      }catch(Exception e ) {
            e.printStackTrace();
      }
      return k;
    }
}
```

AddProductServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.http.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")
```

```
@WebServlet("/add")
public class AddProductServlet extends HttpServlet
{
 protected void doPost(HttpServletRequest req,HttpServletResponse res)throws
 ServletException,IOException
 {
        HttpSession hs = req.getSession(false);//Accessing existing session
        if(hs==null) {
            req.setAttribute("msg", "Session Expired...<br>");
            req.getRequestDispatcher("Msg.jsp").forward(req, res);
        }else {
             ProductBean pb = new ProductBean(); //Bean created
             pb.setCode(req.getParameter("pcode"));
             pb.setName(req.getParameter("pname"));
             pb.setPrice(Float.parseFloat(req.getParameter("pprice")));
             pb.setQty(Integer.parseInt(req.getParameter("pqty")));
             int k = new AddProductDAO().insert(pb);
             if(k>0) {
                 req.setAttribute("msg", "Product Added Successfully...<br>");
                 req.getRequestDispatcher("AddProduct.jsp").forward(req, res);
             }
        }
 }
}
```

AddProduct.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.AdminBean"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
AdminBean ab = (AdminBean)session.getAttribute("abean");
String msg = (String)request.getAttribute("msg");
out.println("Page belongs to Admin:"+ab.getfName()+"<br>");
out.println(msg);
%>
<a href="Product.html">AddProduct</a>
<a href="view">ViewAllProducts</a>
<a href="logout">Logout</a>
</body>
</html>
```

Msg.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String msg = (String)request.getAttribute("msg");
out.println(msg);
%>
<%@include file="home.html" %>
</body>
</html>
```

ViewAllProductsDAO.java

```
package test;

import java.util.*;

import java.sql.*;
```

```
public class ViewAllProductsDAO
{
  public ArrayList<ProductBean> al = new ArrayList<ProductBean>();
  public ArrayList<ProductBean> retrieve()
  {
        try {
              Connection con = DBConnection.getCon();
              PreparedStatement ps = con.prepareStatement
                        ("select * from Product70");
              ResultSet rs = ps.executeQuery();
              while(rs.next()) {
                    ProductBean pb = new ProductBean();
                    pb.setCode(rs.getString(1));
                    pb.setName(rs.getString(2));
                    pb.setPrice(rs.getFloat(3));
                    pb.setQty(rs.getInt(4));
                    al.add(pb);//Bean added to ArrayList
              }//end of loop
        }catch(Exception e) {
              e.printStackTrace();
        }
        return al;
  }
}
```

ViewAllProductsServlet.java

```
package test;

import java.io.*;

import java.util.*;

import jakarta.servlet.*;

import jakarta.servlet.http.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/view")

public class ViewAllProductsServlet extends HttpServlet

{

  protected void doGet(HttpServletRequest req,HttpServletResponse res)throws

  ServletException,IOException

  {

            HttpSession hs = req.getSession(false);

            if(hs==null) {

                        req.setAttribute("msg","Session Expired...<br>");

                        req.getRequestDispatcher("Msg.jsp").forward(req, res);

            }else {

                       ArrayList<ProductBean> al = new ViewAllProductsDAO().retrieve();

                        hs.setAttribute("alist", al);

                        req.getRequestDispatcher("ViewAllProducts.jsp").forward(req, res);

            }

  }

}
```

ViewAllProducts.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.*,java.util.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
AdminBean ab = (AdminBean)session.getAttribute("abean");
ArrayList<ProductBean> al =
(ArrayList<ProductBean>)session.getAttribute("alist");
out.println("Page Belongs to Admin : "+ab.getfName()+"<br>");
if(al.size()==0){
      out.println("Prodcuts not available...<br>");
}else{
      Iterator<ProductBean> it = al.iterator();
      while(it.hasNext()){
      ProductBean pb = (ProductBean)it.next();
      out.println(pb.getCode()+"  "
                  +pb.getName()+"  "
                  +pb.getPrice()+"  "
                  +pb.getQty()+"  "
                  +"<a href='edit'>Edit</a>"+"  "
                  +"<a href='delete'>Delete</a>"+"<br>");
      }
}
%>
<a href="Product.html">AddProduct</a>
<a href="Logout">Logout</a>
</body>
</html>
```

Dt : 18/2/2025

EditProductServlet.java

```
package test;
import java.io.*;
import java.util.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/edit")
public class EditProductServlet extends HttpServlet
{
        @SuppressWarnings("unchecked")
        @Override
 protected void doGet(HttpServletRequest req,HttpServletResponse res)throws
 ServletException,IOException
 {
         HttpSession hs = req.getSession(false);
         if(hs==null) {
            req.setAttribute("msg","Session Expired...<br>");
             req.getRequestDispatcher("Msg.jsp").forward(req, res);
         }else {
                 ArrayList<ProductBean> al = (ArrayList<ProductBean>)hs.getAttribute("alist");
                 String pC = req.getParameter("pcode");
                 Iterator<ProductBean> it = al.iterator();
```

```
                while(it.hasNext())
                {
                        ProductBean pb = (ProductBean)it.next();
                        if(pC.equals(pb.getCode())) {
                                req.setAttribute("pbean", pb);
                                req.getRequestDispatcher("EditProduct.jsp").forward(req, res);
                                break;
                        }//end of if
                }//end of while
        }
 }
}
```

EditProduct.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
AdminBean ab = (AdminBean)session.getAttribute("abean");
ProductBean pb = (ProductBean)request.getAttribute("pbean");
out.println("Page Belongs to Admin : "+ab.getfName()+"<br>");
%>
<form action="update" method="post">
ProductPrice:<input type="text" name="price" value=<%= pb.getPrice() %>><br>
ProductQty:<input type="text" name="qty" value=<%= pb.getQty() %>><br>
<input type="submit" value="UpdateProduct">
</form>
</body>
</html>
```

Dt : 19/2/2025

ViewAllProducts.jsp(Modified Code)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.*,java.util.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
AdminBean ab = (AdminBean)session.getAttribute("abean");
ArrayList<ProductBean> al =
(ArrayList<ProductBean>)session.getAttribute("alist");
out.println("Page Belongs to Admin : "+ab.getfName()+"<br>");
if(al.size()==0){
      out.println("Prodcuts not available...<br>");
}else{
      Iterator<ProductBean> it = al.iterator();
      while(it.hasNext()){
      ProductBean pb = (ProductBean)it.next();
      out.println(pb.getCode()+"  "
                  +pb.getName()+"  "
                  +pb.getPrice()+"  "
                  +pb.getQty()+"  "
                  +"<a
href='edit?pcode="+pb.getCode()+"'>Edit</a>"+"  "
                  +"<a
href='delete?pcode="+pb.getCode()+"'>Delete</a>"+"<br>");
      }
}
%>
<a href="Product.html">AddProduct</a>
<a href="Logout">Logout</a>
</body>
</html>
```

EditProduct.jsp(Modified Code)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.*"%>
<!DOCTYPE html>
<html>
```

```
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
AdminBean ab = (AdminBean)session.getAttribute("abean");
ProductBean pb = (ProductBean)request.getAttribute("pbean");
out.println("Page Belongs to Admin : "+ab.getfName()+"<br>");
%>
<form action="update" method="post">
<input type="hidden" name="pcode" value=<%= pb.getCode() %>>
ProductPrice:<input type="text" name="price" value=<%= pb.getPrice() %>><br>
ProductQty:<input type="text" name="qty" value=<%= pb.getQty() %>><br>
<input type="submit" value="UpdateProduct">
</form>
</body>
</html>
```

UpdateProductDAO.java

```
package test;
import java.sql.*;
public class UpdateProductDAO {
   public int k=0;
   public int update(ProductBean pb) {
         try {
               Connection con = DBConnection.getCon();
               PreparedStatement ps = con.prepareStatement
                           ("update Product70 set price=?,qty=? where
code=?");
               ps.setFloat(1, pb.getPrice());
               ps.setInt(2, pb.getQty());
               ps.setString(3, pb.getCode());
               k = ps.executeUpdate();
         }catch(Exception e) {
                e.printStackTrace();
         }
         return k;
   }
}
```

UpdateProductServlet.java

package test;

```
import java.io.*;
import java.util.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/update")
public class UpdateProductServlet extends HttpServlet
{
   @SuppressWarnings("unchecked")
        @Override
        protected void doPost(HttpServletRequest req,HttpServletResponse res)throws
   ServletException,IOException
   {
        HttpSession hs = req.getSession(false);
        if(hs==null) {
                req.setAttribute("msg","Session Expired...<br>");
                req.getRequestDispatcher("Msg.jsp").forward(req, res);
        }else {
                ArrayList<ProductBean> al = (ArrayList<ProductBean>)hs.getAttribute("alist");
                String pC = req.getParameter("pcode");
                Iterator<ProductBean> it = al.iterator();
                while(it.hasNext()) {
                        ProductBean pb = (ProductBean)it.next();
                        if(pC.equals(pb.getCode())) {
```

```
                                pb.setPrice(Float.parseFloat(req.getParameter("price")));

                                pb.setQty(Integer.parseInt(req.getParameter("qty")));

                                int k = new UpdateProductDAO().update(pb);

                                if(k>0) {

                                        req.setAttribute("msg","Product Updated Successfully...<br>");

                                        req.getRequestDispatcher("UpdateProduct.jsp").forward(req,
res);

                                }//end of

                                break;

                        }//end of if

                }//end of loop

        }

  }

}
```


UpdateProduct.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
AdminBean ab = (AdminBean)session.getAttribute("abean");
String msg = (String)request.getAttribute("msg");
```

```
out.println("Page belongs to Admin : "+ab.getfName()+"<br>");
out.println(msg);
%>
<a href="Product.html">AddProduct</a>
<a href="view">ViewAllProducts</a>
<a href="logout">Logout</a>
</body>
</html>
```

AdminLogoutServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.http.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/logout")

public class AdminLogoutServlet extends HttpServlet

{

        @Override

  protected void doGet(HttpServletRequest req,HttpServletResponse res)throws

  ServletException,IOException

  {

            HttpSession hs = req.getSession(false);

            if(hs==null) {

                    req.setAttribute("msg","Session Expired...<br>");
```

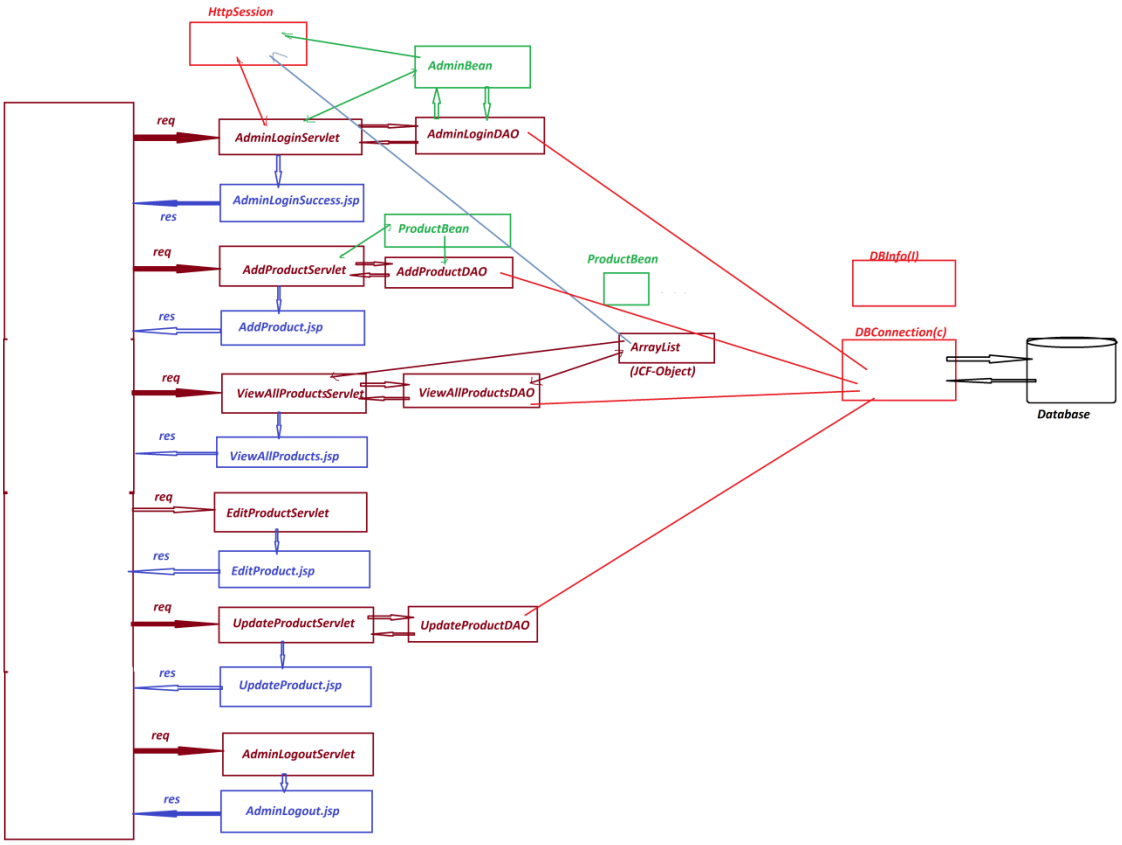
```
                req.getRequestDispatcher("Msg.jsp").forward(req, res);

        }else {

                hs.removeAttribute("abean");

                hs.removeAttribute("alist");

                req.setAttribute("msg","Admin LoggedOut Successfully...<br>");

                req.getRequestDispatcher("AdminLogout.jsp").forward(req, res);

        }

  }

}
```

AdminLogout.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String msg = (String)request.getAttribute("msg");
session.invalidate();
out.println(msg);
%>
<%@include file="home.html" %>
</body>
</html>
```

Dt : 20/2/2025

Summary of 'HttpSession' in Session Tracking process:



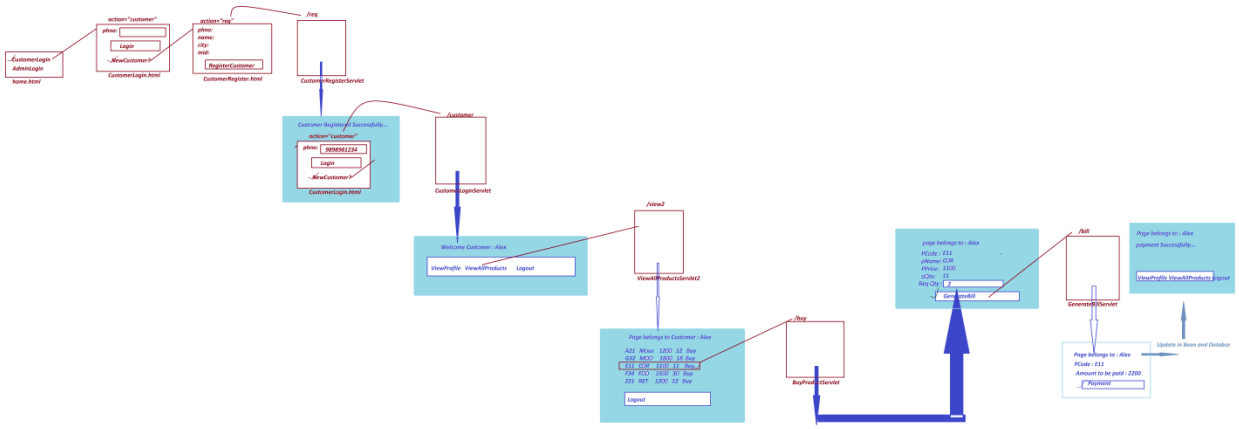
Assignment:

Construct CustomerLogin using the Following layout:

DBTable : CustomerDetails70(phno,cid,cname,city,mid)

Primary key : phno

Layout:



==========

Dt : 22/2/2025

faq:

wt is the diff b/w

(i)JAR

(ii)WAR

(iii)EAR

(i)JAR:

=>JAR means 'Java Archive' and which is compressed format of more number of class files.

=>Stand-Alone-Applications are converted into JAR files.

(ii)WAR:

=>WAR stands for 'Web Archive' and which is compressed format of Class files,JSP files, HTML files,XML files,external JARS and other files

=>Web Applications are converted into WAR files.

(iii)EAR:

=>EAR stands for 'Enterprise Archive' and which is compressed formats of JARs,WARs and other services.

=>Enterprise Applications are converted into EAR files

****imp***

Generating WAR file using IDE Eclipse and executing in Tomcat Server:

step-1 : Generate WAR file

RightClick on Project->Export->WAR file->Browse and select destination folder to save

WAR file->name the file and click 'Save'->Click on 'Finish'

step-2 : Deploy WAR file into Tomcat server for execution

start the Tomcat Server->Access Tomcat Server through WebBrowser->Click on 'Manager App'->

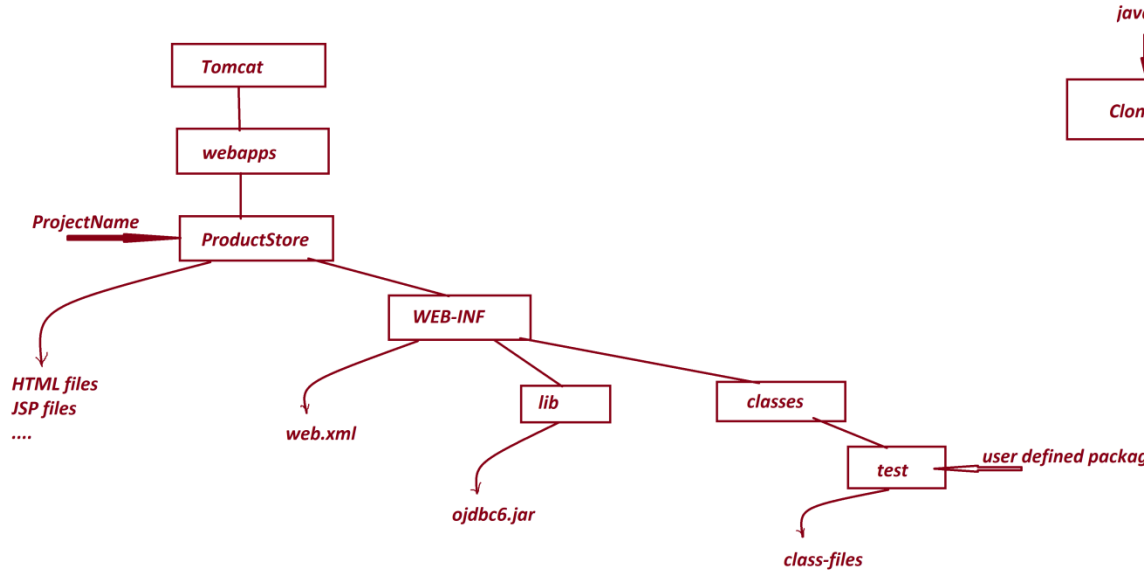
perform login process->Click on 'Choose File' from 'WAR file to deploy'->

Browse and select the file and click 'Open'->Click on 'Deploy'

http://localhost:8082/ProductStore/

**imp*

Deployment directory Structure of Tomcat Server:



==========

2.URL re-write:

=>The process of adding parameter-value to servlet-url-pattern is known as 'URL re-write'

=>Using 'URL re-write' process we can send some data from one servlet to another servlet in Session Tracking process.

syntax:

Servlet-url-pattern?para1=value&pare2=value&...

Note:

"?" - is the separator b/w Servlet-url-pattern and parameters

"&" - is the separator b/w parameters

https://www.google.com/search?

q=download+tomcat+server

&rlz=1C1JJTC_enIN1102IN1102

&oq=download+

&gs_lcrp=EgZjaHJvbWUqDAgDECMYJxiABBiKBTIGCAAQRRg5Mg4IARBFGCcYOxiABBiKBTIGCAIQRRg7MgwIAxAjGCcYgAQYigUyEAgEEAAYkQIYsQMYgAQYigUyEAgFEAAYkQIYsQMYgAQYigUyDQgGEAAYkQIYgAQYigUyEAgHEAAYkQIYsQMYgAQYigUyDQgIEAAYkQIYgAQYigUyDQgJEAAYgwEYsQMYgATSAQk3MzEwajBqMTWoAgiwAgE

&sourceid=chrome

&ie=UTF-8

3.Hidden Form Fields:

=>The process of declaring <input type="hidden" ..> in <form> tag is known as Hidden Form field.

=>The data available in Hidden form field is not displayed to the End-users on WebBrowser

=>Hidden Form fields also support to transfer the data from one servlet to another Servlet in Session tracking process.

syntax:

```
<form action="url" method="">
  <input type="hidden" name="nm" value="val">
  ...
</form>
```

**imp*

4.Cookie:

=>The piece of information which persisted(stored and available) b/w multiple requests is known as cookie.

=>cookie is created by the server,but stored in WebBrowser to track the user in Session

Tracking process.

=>Cookies are categorized into two types:

(i)Persistent Cookies

(ii)NonPersistent Cookies

(i)Persistent Cookies:

=>The cookies which are available in WebBrowser until we logout,are known as Persistent Cookies.

(ii)NonPersistent Cookies:

=>The Cookies which are destroyed automatically when the WebBrowser is closed,is known as NonPersistent cookies.

Note:

=>we use 'Cookie' class from jakarta.servlet.http package in Session Tracking process.

=>The following are some important methods of 'Cookie' Class:

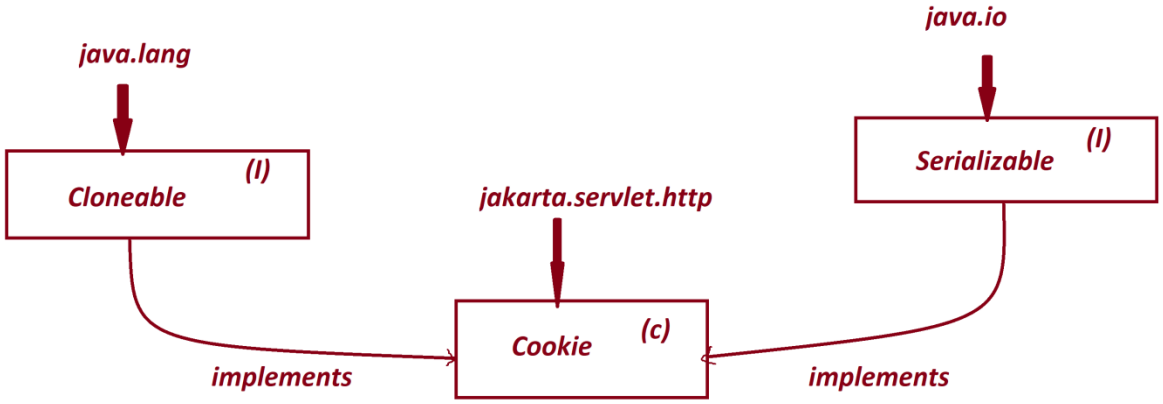
```
public jakarta.servlet.http.Cookie(java.lang.String, java.lang.String);

public void setMaxAge(int);
public int getMaxAge();

public java.lang.String getName();
public void setValue(java.lang.String);
public java.lang.String getValue();
```

public void setAttribute(java.lang.String, java.lang.String);

public java.lang.String getAttribute(java.lang.String);

public java.util.Map<java.lang.String, java.lang.String> getAttributes();

Hierarchy of 'Cookie':

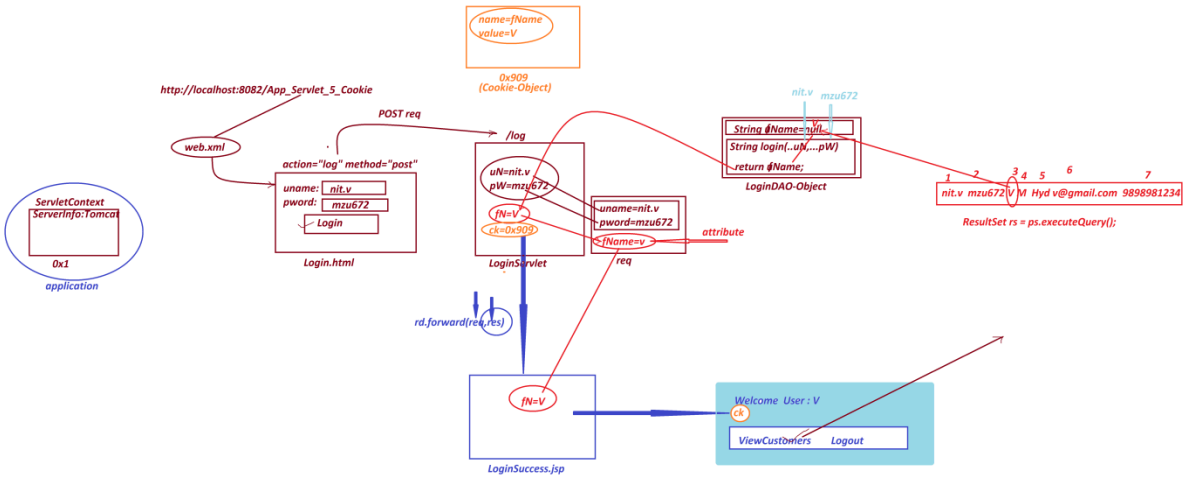


==========

Dt : 22/2/2025

Ex:(Demonstrating 'Cookie' in Session Tracking Process)

Layout:



DBInfo.java

```
package test;
public interface DBInfo
{
   public static final String driver="oracle.jdbc.driver.OracleDriver";
   public static final String dbURL="jdbc:oracle:thin:@localhost:1521:xe";
   public static final String dbUName="system";
   public static final String dbPWord="tiger";
}
```

DBConnection.java

```
package test;
import java.sql.*;
public class DBConnection
{
     private static Connection con = null;
     private DBConnection() {}
     static
     {
       try {
             Class.forName(DBInfo.driver);
             con = DriverManager.getConnection
                         (DBInfo.dbURL,DBInfo.dbUName,DBInfo.dbPWord);
       }catch(Exception e) {
             e.printStackTrace();
```

```
      }
    }//end of block
    public static Connection getCon()
    {
      return con;
    }
}
```

Login.html


```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {font-family: Arial, Helvetica, sans-serif;}
form {border: 3px solid #f1f1f1;}

input[type=text], input[type=password] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  box-sizing: border-box;
}

button {
  background-color: #04AA6D;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
}

button:hover {
  opacity: 0.8;
}

.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color: #f44336;
}

.imgcontainer {
  text-align: center;
```

```
  margin: 24px 0 12px 0;
}

img.avatar {
  width: 40%;
  border-radius: 50%;
}

.container {
  padding: 16px;
}

span.psw {
  float: right;
  padding-top: 16px;
}

/* Change styles for span and cancel button on extra small screens */
@media screen and (max-width: 300px) {
  span.psw {
     display: block;
     float: none;
  }
  .cancelbtn {
     width: 100%;
  }
}
</style>
</head>
<body>

<h2>Login Form</h2>

<form action="log" method="post">
  <div class="imgcontainer">
    <img src="C:\Images\img501.jpg" alt="Avatar" class="avatar">
  </div>

  <div class="container">
    <label for="uname"><b>Username</b></label>
    <input type="text" placeholder="Enter Username" name="uname" required>

    <label for="psw"><b>Password</b></label>
    <input type="password" placeholder="Enter Password" name="pword"
required>

    <button type="submit">Login</button>
    <label>
      <input type="checkbox" checked="checked" name="remember"> Remember me
    </label>
```

```
  </div>

  <div class="container" style="background-color:#f1f1f1">
    <button type="button" class="cancelbtn">Cancel</button>
    <span class="psw">Forgot <a href="#">password?</a></span>
  </div>
</form>

</body>
</html>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <welcome-file-list>
     <welcome-file>Login.html</welcome-file>
  </welcome-file-list>
</web-app>
```

LoginDAO.java

```
package test;
import java.sql.*;
public class LoginDAO
{
    public String fName=null;
    public String login(String uN,String pW)
    {
      try {
            Connection con = DBConnection.getCon();
            PreparedStatement ps = con.prepareStatement
                        ("select * from Admin70 where uname=? and pword=?");
            ps.setString(1, uN);
            ps.setString(2, pW);
            ResultSet rs = ps.executeQuery();
            if(rs.next()) {
                  fName = rs.getString(3);
            }
      }catch(Exception e) {
            e.printStackTrace();
      }
      return fName;
    }
}
```

LoginServlet.java

```
package test;

import java.io.*;

import jakarta.servlet.*;

import jakarta.servlet.http.*;

import jakarta.servlet.annotation.*;

@SuppressWarnings("serial")

@WebServlet("/log")

public class LoginServlet extends HttpServlet

{

        @Override

  protected void doPost(HttpServletRequest req,HttpServletResponse res)throws

  ServletException,IOException

  {

            String uN = req.getParameter("uname");

            String pW = req.getParameter("pword");

            String fN = new LoginDAO().login(uN, pW);

            if(fN==null) {

                    req.setAttribute("msg","Invalid Login Process...<br>");

                    req.getRequestDispatcher("Msg.jsp").forward(req, res);

            }else {

                    req.setAttribute("fName", fN);

                    Cookie ck = new Cookie("fname",fN);

                    res.addCookie(ck);//Adding Cookie Object to response

                    req.getRequestDispatcher("LoginSuccess.jsp").forward(req, res);

            }
```

```
 }

}
```

LoginSuccess.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String fN = (String)request.getAttribute("fName");
out.println("Welcome User : "+fN+"<br>");
%>
<a href="cview">ViewCustomers</a>
<a href="logout">Logout</a>
</body>
</html>
```

Dt : 24/2/2025

CustomerBean.java

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class CustomerBean implements Serializable{
   private int id;
   private String name,city,mId;
   private long phNo;
   public CustomerBean() {}
public int getId() {
      return id;
}
public void setId(int id) {
      this.id = id;
}
public String getName() {
      return name;
}
public void setName(String name) {
      this.name = name;
}
public String getCity() {
      return city;
}
public void setCity(String city) {
      this.city = city;
}
public String getmId() {
      return mId;
}
public void setmId(String mId) {
      this.mId = mId;
}
public long getPhNo() {
      return phNo;
}
public void setPhNo(long phNo) {
      this.phNo = phNo;
}

}
```

ViewCustomersDAO.java

package test;

```
import java.sql.*;
import java.util.*;
public class ViewCustomersDAO {
  public ArrayList<CustomerBean> al = new ArrayList<CustomerBean>();
  public ArrayList<CustomerBean> retrieve(){
        try {
              Connection con = DBConnection.getCon();
              PreparedStatement ps = con.prepareStatement
                        ("select * from Customer70");
              ResultSet rs = ps.executeQuery();
              while(rs.next()) {
                    CustomerBean cb = new CustomerBean();
                    cb.setId(rs.getInt(1));
                    cb.setName(rs.getString(2));
                    cb.setCity(rs.getString(3));
                    cb.setmId(rs.getString(4));
                    cb.setPhNo(rs.getLong(5));
                    al.add(cb);
              }//end og loop
        }catch(Exception e) {
              e.printStackTrace();
        }
        return al;
  }
}
```

ViewCustomersServlet.java

```
package test;
import java.io.*;
import java.util.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/cview")
public class ViewCustomersServlet extends HttpServlet
{
        @Override
  protected void doGet(HttpServletRequest req,HttpServletResponse res)throws
  ServletException,IOException
  {
            Cookie c[] = req.getCookies();
            if(c==null) {
                    req.setAttribute("msg", "Session Expired...<br>");
                    req.getRequestDispatcher("Msg.jsp").forward(req, res);
            }else {
                     String fN = c[0].getValue();
                     req.setAttribute("fname", fN);
                     ArrayList<CustomerBean> al = new ViewCustomersDAO().retrieve();
                     req.setAttribute("alist", al);
                     req.getRequestDispatcher("ViewCustomers.jsp").forward(req, res);
```

```
        }

 }

}
```

ViewCustomers.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.*,java.util.*"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String fN = (String)request.getAttribute("fname");
ArrayList<CustomerBean> al =
(ArrayList<CustomerBean>)request.getAttribute("alist");
out.println("Page belongs to : "+fN+"<br>");
if(al.size()==0){
      out.println("No Customers available...<br>");
}else{
      Iterator<CustomerBean> it = al.iterator();
      while(it.hasNext()){
            CustomerBean cb = (CustomerBean)it.next();
            out.println(cb.getId()+"  "
                        +cb.getName()+"  "
                        +cb.getCity()+"  "
                        +cb.getmId()+"  "
                        +cb.getPhNo()+"<br>");
      }
}
%>
<a href="logout">Logout</a>
</body>
</html>
```

LogoutServlet.java

```
package test;

import java.io.*;
```

```
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
@SuppressWarnings("serial")
@WebServlet("/logout")
public class LogoutServlet extends HttpServlet
{
        @Override
  protected void doGet(HttpServletRequest req,HttpServletResponse res)throws
  ServletException,IOException
  {
        Cookie c[] = req.getCookies();
        if(c==null) {
                req.setAttribute("msg", "Session Expired...<br>");
                req.getRequestDispatcher("Msg.jsp").forward(req, res);
        }else {
                c[0].setMaxAge(0);
                res.addCookie(c[0]);
                req.setAttribute("msg", "User Logged Out Successfully...<br>");
                req.getRequestDispatcher("Logout.jsp").forward(req, res);
        }
  }
}
```

Logout.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String msg = (String)request.getAttribute("msg");
out.println(msg);
%>
<%@include file="Login.html" %>
</body>
</html>
```

Msg.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String msg = (String)request.getAttribute("msg");
out.println(msg);
%>
<%@include file="Login.html" %>
</body>
</html>
```

Dt : 25/2/2025

faq:

wt is the diff b/w

(i)HttpSession

(ii)Cookie

(i)HttpSession:

=>'HttpSession' is an interface from jakarta.servlet.http package and which is instantiated to track the user in Session tracking process.

=>'HttpSession' is server dependent,because which is created and available in server until it is invalidated.

(ii)Cookie:

=>'Cookie' is a class from jakarta.servlet.http package and which is instantiated to track the user in session tracking process.

=>'Cookie' is client dependent,because which which is created by the server,but stored in WebBrowser to track the user.

---

faq:

wt is the diff b/w

(i)addCookie()

(ii)getCookies()

=>addCookie()-method will perform Serialization process and Cookie-stream to response.

=>getCookies()-method will count the number of cookies from the request object and create array of size equal to the count of cookies.

(This getCookies()-method internally perform DeSerialization)

**imp*

Web Application Architectures:

=>According to realtime,Web Application Architectures are categorized into two types:

1.Model-1 Architecture

2.Model-2 Architecture

1.Model-1 Architecture:

=>In Model-1 Architecture JSP is the controller of application,which means JSP will take the request and provide the response,in this process it controls Beans,DAOs and JCFs.

Diagram:

2.Model-2 Architecture:

=>Model-2 Architecture introduced to overcome the dis-advantages of Model-1 Architecture.

=>Model-2 Architecture is based on MVC Architecture.

M - Model

V - View

C - Controller

M - Model:

=>Model layer represents beans,where enterprise-data is available.

=>This Model layer also includes DAOs,JCfs and services.

V - View :

=>View represents JSP and which is presentation of Web-Application.

C - Controller:

=>Controller represents 'Servlet' and which controls all the layers in the application.

Diagram:

==========

**imp*

JSP Programming:

=>We use the following JSP tags to construct JSP programs:

1.Scripting tags

2.Directive tags

3.Action tags.

1.Scripting tags:

=>Scripting tags are used to write noraml Java Code in jsp programs.

=>Types:

(a)Scriptlet tag

(b)Expression tag

(c)Declarative tag

(a)Scriptlet tag:

=>Scriptlet tag is used to write normal ServletCode or JavaCode in JSP programs

syntax:

<%

-----ServletCode/JavaCode----

%>

(b)Expression tag:

=>Expression tag is used to assign the value to variable or used to send the data to

WebBrowser directly.

syntax:

<%=

-----value/Expression----

%>

(c)Declarative tag:

=>Declarative tag is used to declare variables and methods in JSP programs

syntrax:

<%!

-----Variables;methods------

%>

2.Directive tags:

=>Directive tags will specify the information about current running JSP page(Program)

=>Types:

(a)@page

(b)@include

(c)@taglib

(a)@page:

='@page' tag will give information about the Current JSP page.

syntax:

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"

import="test.,java.util.*"%>*

(b)@include:

=>'@include' tag is used to include the file in JSP-response

syntax:

<%@include file="home.html" %>

(c)@taglib:

=>'@taglib' tag is used to link external libraries to current JSP page

syntax:

<%@taglib .....%>

Dt : 26/2/2025

Ex:(Demonstrating Declarative tag)

Diagram:

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="Calculate.jsp" method="post">
Enter the Value:<input type="text" name="v"><br>
<input type="submit" value="Factorial">
</form>
</body>
</html>
```

web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<web-app>
  <welcome-file-list>
     <welcome-file>input.html</welcome-file>
  </welcome-file-list>
</web-app>
```

Calculate.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    errorPage="Error.jsp"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%!
int fact;
int factorial(int n)
{
      fact=1;
      for(int i=n;i>=1;i--)
      {
            fact = fact*i;
      }
      return fact;
}
%>
<%
int v = Integer.parseInt(request.getParameter("v"));
int res = factorial(v);
out.println("Factoral = "+res+"<br>");
%>
<%@include file="input.html" %>
</body>
</html>
```

Error.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
```

```
    pageEncoding="ISO-8859-1"
    isErrorPage="true"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
out.println("Enter only Integer value....<br>");
%>
<%= exception %>
<br>
<%@include file="input.html" %>
</body>
</html>
```

---

*imp

=>The following are the implicit objects of JSP:

1.application - jakarta.servlet.ServletContext

2.config - jakarta.servlet.ServletConfig

3.request - jakarta.servlet.http.HttpServletRequest

4.response - jakarta.servlet.http.HttpServletResponse

5.out - jakarta.servlet.jsp.JspWriter(AbstractClass)

6.session - jakarta.servlet.http.HttpSession

7.exception - java.lang.Exception

8.page - java.lang.Object

9.pageContext - jakarta.servlet.jsp.PageContext(AbstractClass)

---

faq:

define 'page' implicit Object of JSP?

=>'page' is an implicit object of java.lang.Object class.

=>'page' is used when we want to perform Thread-Communications and Cloning process in JSP.

faq:

define 'pageContext' implicit Object of JSP?

=>'pageContext' is an implicit Object generated from 'PageContext' AbstractClass

=>Using 'pageContext' Object we can access all other JSP Objects directly

==========

3.Action tags.:

=>Action tags will specify the actions performed while JSP Program execution.

=>Types:

(a)<jsp:forward>

(b)<jsp:include>

(c)<jsp:param>

(d)<jsp:useBean>

(e)<jsp:setProperty>

(f)<jsp:getProperty>

(a)<jsp:forward>:

=><jsp:forward> tag is used to perform forward communication in JSP pages.

syntax:

<jsp:forward page="JSP-File"/>

(b)<jsp:include>:

=><jsp:include> tag is used to perform include communication in JSP pages.

syntax:

<jsp:include page="JSP-File"/>

(c)<jsp:param> :

=><jsp:param> is the subtag of <jsp:forward> and which is used to send para-value from one JSP page to another JSP page.

syntax:

```
<jsp:forward page="JSP-File">
  <jsp:param value="<%=1000 %>" name="y"/>
</jsp:forward>
```

Ex:(Demonstrating JSP Communications)

input.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="Choice.jsp" method="post">
Enter the Value-1:<input type="text" name="v1"><br>
Enter the Value-2:<input type="text" name="v2"><br>
<input type="submit" name="sb" value="Add">
<input type="submit" name="sb" value="Sub">

</form>
</body>
</html>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <welcome-file-list>
     <welcome-file>input.html</welcome-file>
  </welcome-file-list>
</web-app>
```

Choice.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String sb = request.getParameter("sb");
if(sb.equals("Add")){
%>
<jsp:forward page="Addition.jsp">
   <jsp:param value="<%= 1000 %>" name="p"/>
</jsp:forward>
<%
}else{
%>
<jsp:forward page="Subtraction.jsp">
    <jsp:param value="<%= 2000 %>" name="q"/>
</jsp:forward>
<%
}
%>
</body>
</html>
```

Addition.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
```

```
<title>Insert title here</title>
</head>
<body>
<%
int v1 = Integer.parseInt(request.getParameter("v1"));
int v2 = Integer.parseInt(request.getParameter("v2"));
int v3 = v1+v2;
out.println("Sum="+v3+"<br>");
out.println("Para-Value="+request.getParameter("p")+"<br>");
%>
<jsp:include page="input.html"/>
</body>
</html>
```

Subtraction.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
int v1 = Integer.parseInt(request.getParameter("v1"));
int v2 = Integer.parseInt(request.getParameter("v2"));
int v3 = v1-v2;
out.println("Sub="+v3+"<br>");
out.println("Para-Value="+request.getParameter("q")+"<br>");
%>
<jsp:include page="input.html"/>
</body>
</html>
```

==========

Dt : 27/2/2025

(d)<jsp:useBean>

(e)<jsp:setProperty>

(f)<jsp:getProperty>

(d)<jsp:useBean>:

=><jsp:useBean> is used to instantiate bean object or access the existing bean object.

syntax:

<jsp:useBean id="ub" class="test.UserBean" scope="session"/>

<jsp:useBean id="ub" type="test.UserBean"></jsp:useBean>

(e)<jsp:setProperty>:

=><jsp:setProperty> is used to load the data to bean Object.

syntax:

<jsp:useBean property="pro-name" param="para-name" name="bean-name">

(f)<jsp:getProperty>:

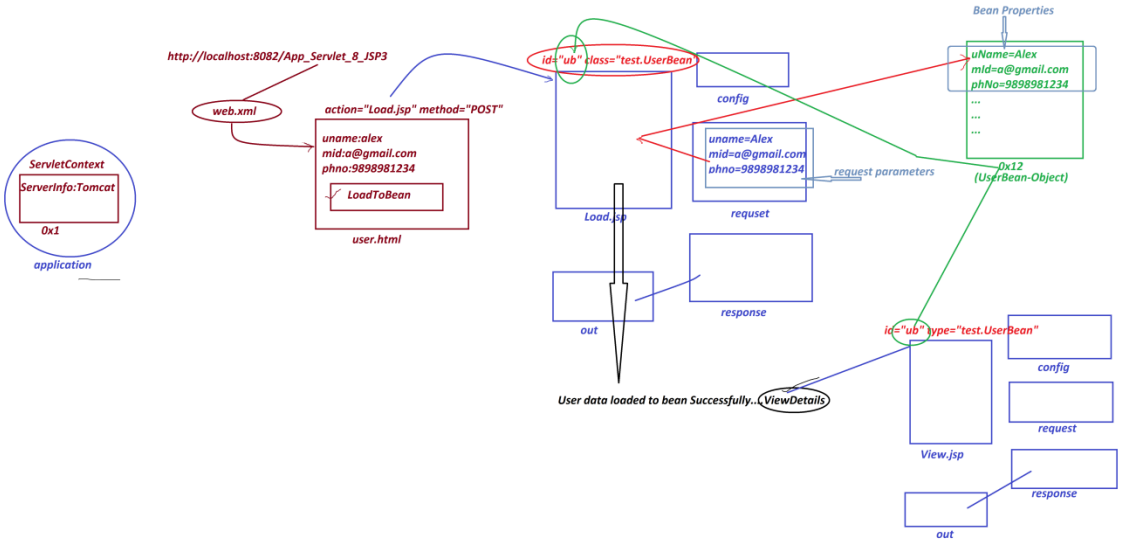
=><jsp:getProperty> is used to get the data from bean Object.

syntax:

=><jsp:getProperty property="pro-name" name="bean-name">

Ex:(Demonstrating <jsp:useBean>,<jsp:setProperty> and <jsp:getProperty>)

Layout:



user.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="Load.jsp" method="post">
UserName:<input type="text" name="uname"><br>
MailId:<input type="text" name="mid"><br>
PhoneNO:<input type="text" name="phno"><br>
<input type="submit" value="LoadToBean">
</form>
</body>
</html>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app>
   <welcome-file-list>
      <welcome-file>user.html</welcome-file>
```

```
   </welcome-file-list>
</web-app>
```

UserBean.java

```
package test;
import java.io.*;
@SuppressWarnings("serial")
public class UserBean implements Serializable
{
    private String uName,mId;
    private long phNo;
    public UserBean() {}
      public String getuName() {
            return uName;
      }
      public void setuName(String uName) {
            this.uName = uName;
      }
      public String getmId() {
            return mId;
      }
      public void setmId(String mId) {
            this.mId = mId;
      }
      public long getPhNo() {
            return phNo;
      }
      public void setPhNo(long phNo) {
            this.phNo = phNo;
      }

}
```

Load.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.UserBean"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<jsp:useBean id="ub" class="test.UserBean" scope="session"/>
<jsp:setProperty property="uName" param="uname" name="ub"/>
```

```
<jsp:setProperty property="mId" param="mid" name="ub"/>
<jsp:setProperty property="phNo" param="phno" name="ub"/>
<%
out.println("User data loaded to bean Successfully....");
%>
<a href="View.jsp">ViewDetails</a>
</body>
</html>
```

View.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"
    import="test.UserBean"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<jsp:useBean id="ub" type="test.UserBean" scope="session"/>
UserName:<jsp:getProperty property="uName" name="ub"/><br>
MailId:<jsp:getProperty property="mId" name="ub"/><br>
PhoneNo:<jsp:getProperty property="phNo" name="ub"/>
</body>
</html>
```

==========

**imp*

Expression Language(EL):

=>Expression Language(EL) introduced into JSP to make the code more simple and can access all the implicit objects of JSP.

syntax:

$(Expression)

=>The folowing are the implicit objects of EL:

1.applicationScope

2.sessionScope

3.requestScope

4.pageScope

5.param

6.paramValues

7.header

8.headerValues

9.cookie

10.initParam

11.pageContext

1.applicationScope:

=>applicationScope is used to access the attribute from ServletContext.

2.sessionScope:

=>sessionScope is used to access the attribute from HttpSession.

3.requestScope:

=>requestScope is used to access the attribute from request-object.

4.pageScope:

=>pageScope is used to access the attribute from PageContext Object.

5.param:

=>param is used to access single parameter-value from the request-Object

6.paramValues:

=>paramValues is used to access multiple parameter-values from the request object

7.header:

=>header is used retrive HTTP protocol header name.

8.headerValues:

=>headerValues is used retrieve HTTP protocol multiple header names.

9.cookie:

=>cookie is used to get the cookies from request-object.

10.initParam:

=>initParam is used to access the initilization parameter-values from config object.

11.pageContext:

=>pageContext is implicit name used by JSP-EL to access PageContext-object

---

Summary of Objects Generated from CoreJava:

1.User defined Class Objects

2.String-Objects

3.WrapperClass-Objects

4.Array-Objects

5.Collection<E>-Objects

6.Map<K,V>-Objects

7.Enum<E>-Objects

Summary of Objects generated from JDBC

1.Connection Object

2.Statement Object

3.PreparedStatement Object

4.CallableStatement Object

5.ResultSet Object

(i)Scrollable ResultSet Object

(ii)NonScrollable ResultSet Object

6.RowSet Object

(i)JdbcRowSet Object

(ii)CachedRowSet Object

7.DatabaseMetaData Object

8.ParameterMetaData Object

9.ResultSetMetaData Object

10.RowSetMetaData Object

Summary of Objects generated from Servlet Programming:

1.ServletContext Object

2.ServletConfig Object

3.ServletRequest/HttpServletRequest Object

4.ServletResponse/HttpServletResponse Object

5.PrintWriter Object

6.HttpSession Object

7.Cookie Object

8.Bean Object

9.JCF Object

10.DAO Object

Summary of implicit Objects generated from JSP programming:

1.application

2.config

3.request

4.response

5.out

6.session

7.exception

8.page

9.pageContext

JSP-EL:(Implicit Objects)

1.applicationScope

2.sessionScope

3.requestScope

4.pageScope

5.param

6.paramValues

7.header

8.headerValues

9.cookie

10.initParam

11.pageContext

==========

define JSTL?

=>JSTL stands for 'JSP Standard Tag Library' and,which provide some tags to make code more simple.

=>The following are some important JSTL tags:

1.Core Tags

2.Function Tags

3.Formatting Tags

4.XML Tags

5.SQL Tags

1.Core Tags:

=>The fundamental tags used for writing basic codes are known as CoreTags.

2.Function Tags:

=>The tags which support String functions are known as Function Tags.

3.Formatting Tags:

=>The tags which are used to format the data for persentation are known as Formatting tags.

4.XML Tags:

=>The tags which are used to write XML code in JSP are known as XML tags.

5.SQL Tags:

=>The tags which are used to write DB related Code in JSP are known as SQL Tags.

Note:

=>According to realtime application development using MVC,'SQL-Tags' and 'XML-Tags' are deprecated.

=>"@taglib" directive tag is used to include JSTL libraries into Current Running Program.

==========

Dt : 28/2/2025

Diagram:

Ex:(EL-JSTL Application)

step-1 : Download "jstl-1.2.jar" file from Internet Source.

step-2 : Copy the JSTL jar file into "lib" folder of WEB-INF

step-3 : Type and execute the program

Ex:

input.html

```
<form method="post" action="CoreTags.jsp">
 Enter the String:<input type="text" name="str"><br>
 <input type="submit" value="Display">
```

CoreTags.jsp

```
<%@ page language="java"  contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html> <html> <head>
<meta charset="ISO-8859-1">
```

```
<title>Insert title here</title>
</head>
<body>
<c:set var="n" value="${param.str}" />
<c:forEach var="j" begin="1" end="5">
<c:out value="${n}"/><p>
</c:forEach>
<c:forTokens items="${param.str}" delims=" " var="name">
<c:out value="${name}"/><p>
</c:forTokens>
</body>
</html>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <welcome-file-list>
    <welcome-file>input.html</welcome-file>
  </welcome-file-list>
</web-app>
```

==========

Note:

=>In the process of establishing Communication b/w applications in realtime,the Web-Application will collect the data into ServletContext and ServletConfig Objects

through web.xml.

=>The information in ServletContext Object can be shared by all Servlet and JSP programs of application.

=>The information in 'ServletConfig' is used by only one Servlet Program

---

faq:

define 'Filter' in Servlet Programming?

=>'Filter' is a pre-processing component executed before Servlet program,which means Filter will accept the request first and then sends to Servlet-program

=>Filter and Servlet programs are executed with same url-pattern,but Filter will have highest priority in execution than Servlet.

Note:

=>In realtime filters are used in WebServices.

---

faq:

define "listener" in Servlet Programming?

=>The program which is executed background to Servlet-Objects is known as Listener.

=>In Servlet Programming,we can add listeners to the following three Servlet Objects:

(a)ServletContext - ServletContextListener,ServletContextAttributeListener

(b)ServletRequest - ServletRequestListener,ServletRequestAttributeListener

(c)HttpSession - HttpSessionListener,HttpSessionAttributeListener

==========