*[Font/size: DejaVu Sans Mono/12] [ ForCode Font/size: Consolas/11] Date: 25/2/2025 (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) Class

(f) Interface

(g) Abstract Class

**2.Java Programming Concepts**

(a) Object Oriented Programming

(b) Exception Handling Process

(c) Java Collection Framework

(d) Multi-Threading Concept

(e) File Storage in Java

(f) Networking in Java

**3.Object Oriented Programming features**

(a) Class

(b) Object

(c) Abstraction

(d) Encapsulation

(e) Polymorphism

(f) Inheritance

**Note:**

* Using CoreJava Components, Concepts and Construction rules we can develop
* NonServer-Applications (which means Stand-Alone-Applications)

**Define Stand-Alone-Application?**

* The Application which is installed in one Computer and performs actions in the same computer, is known as Stand-Alone-Application or NonServer-Application.

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:

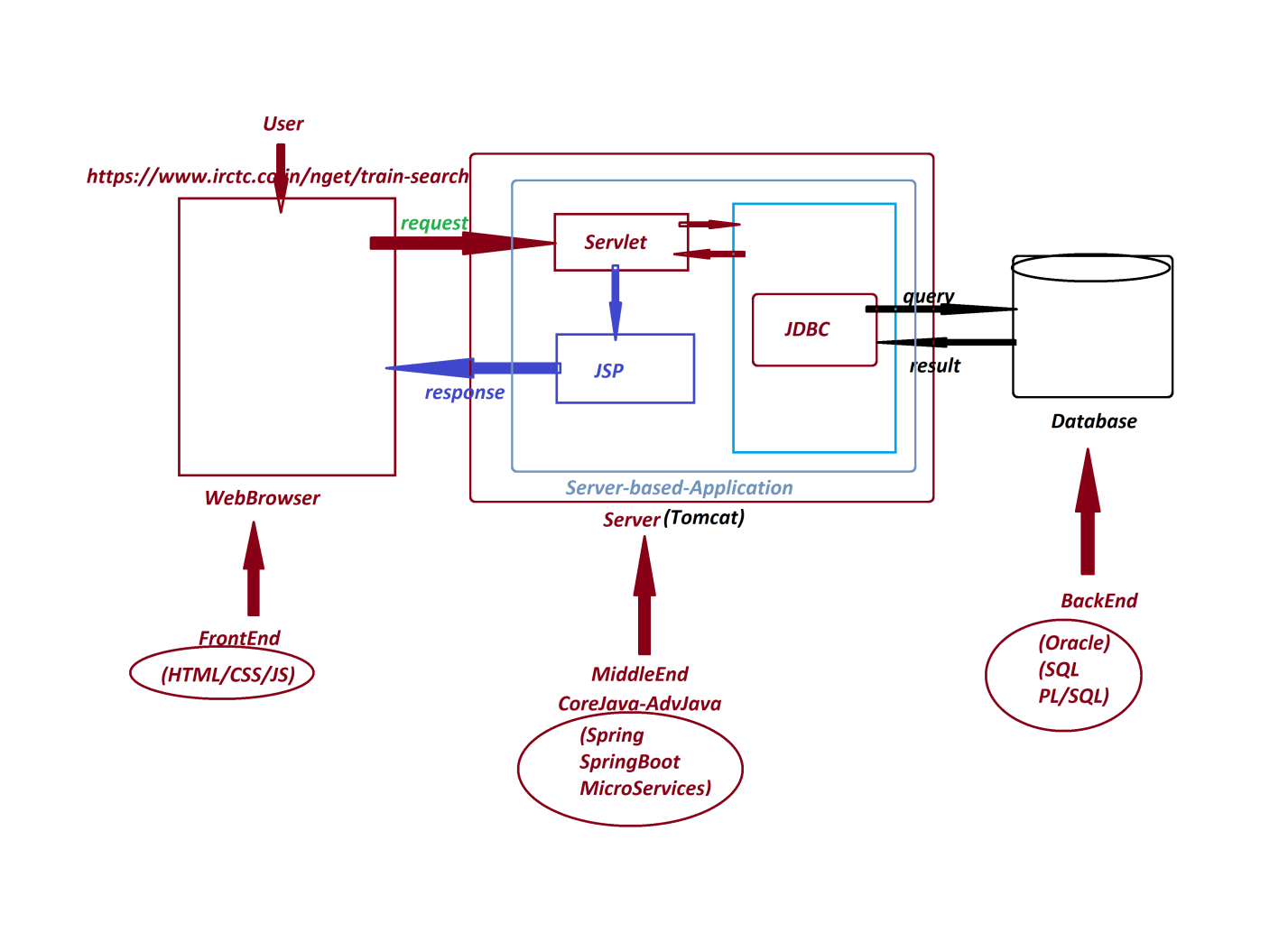
*1. Web Applications*

*2. Enterprise Applications*

**1. Web Applications:**

* The Applications which are constructed using AdvJava technologies like JDBC, Servlet and JSP are known as Web Applications.
* These Web Applications are available in 3-tier Architecture.

***Diagram:***



**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.
* Enterprise Applications are available in n-tier Architecture

*Ex: Java-Frameworks*

Java-Tools

-----

***\*imp***

**JDBC:(Part-1)**

* JDBC stands for ***'Java DataBase Connectivity***' and which is used to interact with database product.

-----

FAQ

##### Define Storage?

* The memory location where the data is available for access 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.
* When we use Primitive datatypes like byte, short, int, long, float, double, char and boolean will generate Field Storages.

**2. Object Storage:**

* The memory generated to hold group values is known as Object-Storage.
* when we use Non- Primitive datatypes like Class, Interface, Array and Enum will generate Object Storage.

*Date: 26/2/2025 (Day-2)*

Example

|  |
| --- |
| class Addition  {  static int a;      int b;      void add()      {         int c = a+b:        Sop(c);      }  }  Addition ad = new Addition();  ad.a = 12;  ad.b = 13;  ad.add();  Addition ob2 = ad; |
| Diagram |

FAQ

**What is the difference 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. It is created using the new keyword.

|  |
| --- |
| class Book {  String title;  double price;  }  public class ObjectExample {  public static void main(String[] args) {  Book b1 = new Book(); // Object creation  }  }  ------------------------------------------------------  -> Here, **new Book();** creates an **object** in memory. |

**(ii) Object reference:**

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

|  |
| --- |
| class Book {  String title;  double price;  }  public class ObjectReferenceExample {  public static void main(String[] args) {  Book b1 = new Book(); // 'b1' holds the reference of the object  System.out.println(b1); // Prints memory address (hashcode) of the object  }  }  -> Here, b1 is storing the **reference** (memory location) of the object. |

**(iii) Object reference Variable:**

* The Nonprimitive-data-type variable which is holding Object reference is known as Object reference Variable or Object name.

|  |
| --- |
| class Book {  String title;  double price;  }  public class ObjectReferenceVariableExample {  public static void main(String[] args) {  Book b1 = new Book(); // 'b1' is an Object Reference Variable  Book b2; // Declaring an Object Reference Variable without initializing  b2 = b1; // Assigning reference of 'b1' to 'b2'    System.out.println("b1 reference: " + b1);  System.out.println("b2 reference: " + b2);  }  }  /\*\* OUTPUT  b1 reference: Book@5e91993f  b2 reference: Book@5e91993f  \*/  -> Here, b1 and b2 are **Object Reference Variables** that hold the same **Object Reference**. |

-----

***\*imp***

**List of Objects generated from CoreJava:**

1. User defined Class Objects

2. String-Objects

3. WrapperClass-Objects

4. Array-Objects

5. Collection<E>-Objects

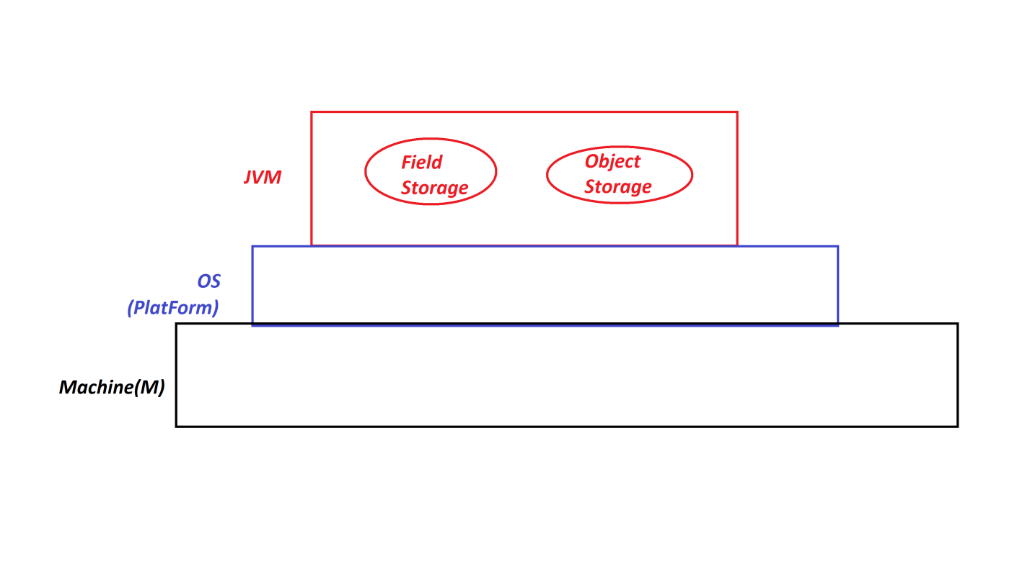
6. Map<K, V>-Objects

7. Enum<E>-Objects

-----

***Note:***

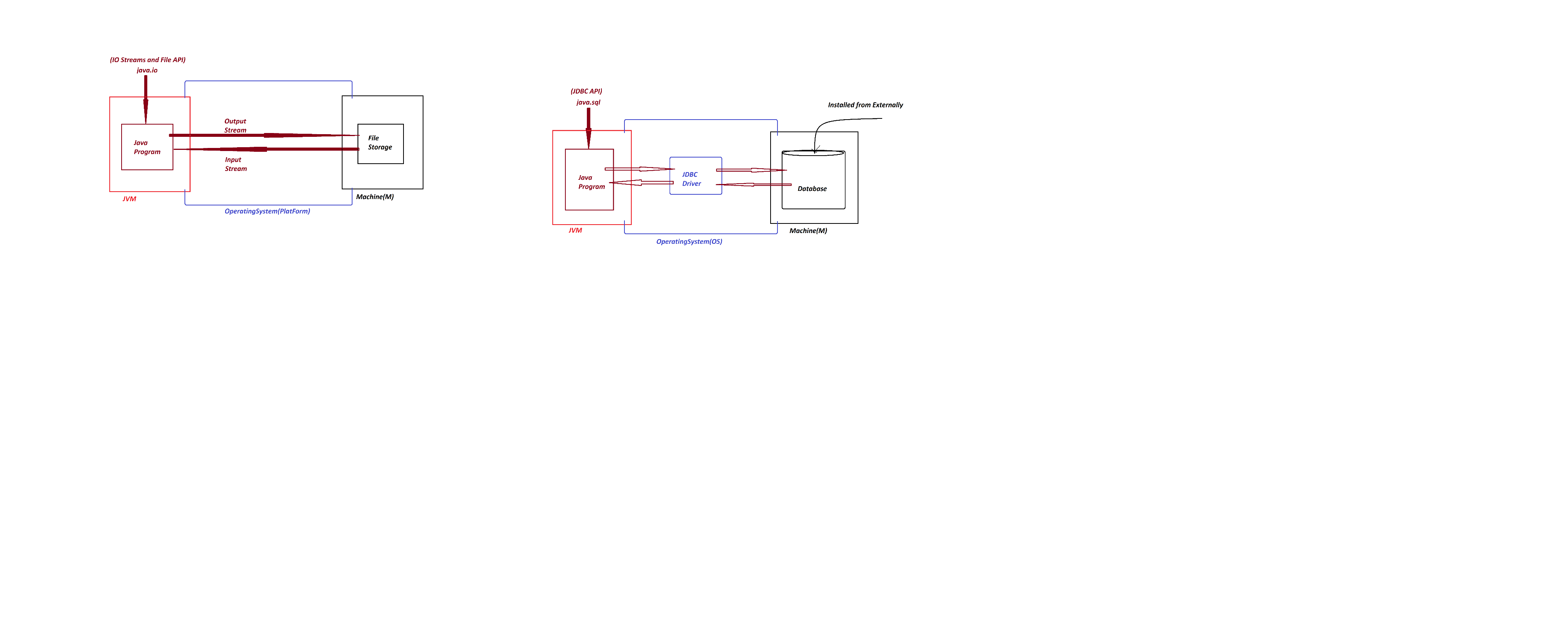
* The Field and Object Storages which are generated part of JVM while Application execution will be destroyed automatically when JVM Shutdowns.
* when we want to have permanent storage for Applications, then we have to take the support of any one of the following:
* File Storage
* Database Storage



*Date: 27/02/2025 (Day-2)*

**3. File Storage:**

* The smallest permanent storage of Computer System which is 'controlled and managed' by the Operating System is known as File Storage.
* In the process of establishing communication b/w Java-Program and File-Storage, the Java-Program must be Constructed using 'Classes and Interfaces' available from 'java.io' package (IO Streams and File API)



**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 known as Data redundancy. (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***

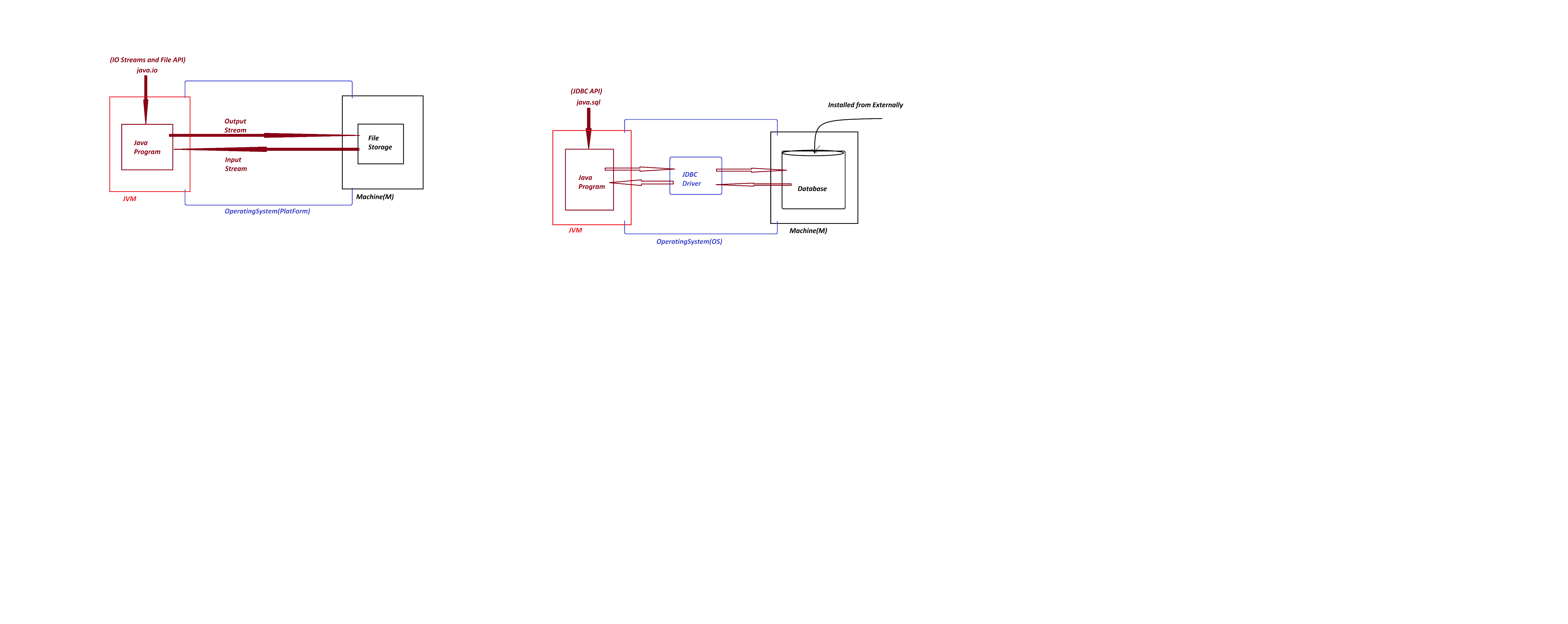
* Because of disadvantages in File-Storage, the File-Storage cannot be taken as major backend for Java-Applications.
* To Overcome all disadvantages of File-Storage, we use Database Storage.

=====

***\*imp***

**4. Database Storage:**

* The largest permanent storage of Computer System, which is installed from externally is known as Database Storage.
* In the process of establishing Communication b/w Java-Program and Database-Product, the Java-Program must be constructed using 'Classes and Interfaces' from 'java.sql' package (JDBC API) and the Java-Program must take the support of JDBC-Driver



------

FAQ

#### Define 'driver'?

* The small s/w program which is used to establish communication b/w two end-points is known as 'driver'.
* *Ex: Audio drivers, Video drivers, N/W drivers*

-----

***FAQ***

#### Define JDBC driver?

* The driver which is used to establish communication b/w Java-Program and DB-Product is known as JDBC driver.

**Types of drivers:**

* According Vendor the 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)*

-----

***Note***

* According to realtime application development, we use only 'Thin driver'

*Date: 28/02/2025 (Day-4)*

***\*imp***

**Creating System Environment ready to execute JDBC Applications:**

***step-1*** : Download and Install Database Product(Oracle)

***step-2*** : Perform Login process to Database Product

* DB UserName : system
* DB Password : tiger

***step-3*** : Create table with name Customer72

(phno, cid, name, city, mid) Primary Key : phno

*SQL> create table Customer72(*

*phno number(15),*

*cid varchar2(15),*

*name varchar2(15),*

*city varchar2(15),*

*mid varchar2(25),*

*primary key(phno)*

*);*

-----

***step-4*** : Insert min 5 Customer details from SQL-Command-Line

insert into Customer72 values(9898981234,'HM9898981234','Alex','Hyd','a@gmail.com');

insert into Customer72 values(7676761234,'HM7676761234','Raj','Hyd','rj@gmail.com');

insert into Customer72 values(8686861234,'HM8686861234','Ram','Hyd','rm@gmail.com');

-----

*SQL> Select \* from Customer72;*

*PHNO CID NAME CITY MID*

*---------------------------------------------------------------------------------*

*9898981234 HM9898981234 Alex Hyd a@gmail.com*

*7676761234 HM7676761234 Raj Hyd rj@gmail.com*

*8686861234 HM8686861234 Ram Hyd rm@gmail.com*

-----

***step-5***  :

* Copy DB-Jar file from "lib" folder of Oracle to User defined folder(on Desktop)

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

ojdbc6.jar - Oarcle11

FAQ

#### Define JAR?

#### JAR stands for 'Java Archive' and which is compressed format of more number of Class files.

***Note***

* This DB-Jar file will provide JDBC drivers.

-----

***step-6*** : Find the PortNo and ServiceName of Database Product(Oracle) PortNo and ServiceName is available from 'tnsnames.ora' file of 'Admin' folder of network

**C:\oraclexe\app\oracle\product\11.2.0\server\network\ADMIN**

*PortNo : 1521*

*ServiceName : XE (Express Edition)*

------

***\*imp***

***Steps used to establish communication to Database product:***

*step-1* : Loader driver

*step-2* : Creating Connection to Database Product

*step-3* : preparing JDBC-statement

*step-4* : Executing the query

*step-5* : Closing the connection from Database

-----

***\*imp***

**JDBC API:**

* 'java.sql' package is known as JDBC-API and which provide 'classes and Interfaces' to Construct JDBC-Applications.
* 'Connection' is a Normal interface from java.sql package and which is root of JDBC API.
* 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()

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

******

*Date: 03/03/2025 (Day-5)*

**\*imp**

**Application Development design Models:**

**Model-1 : Interface with Implementation class with name.**

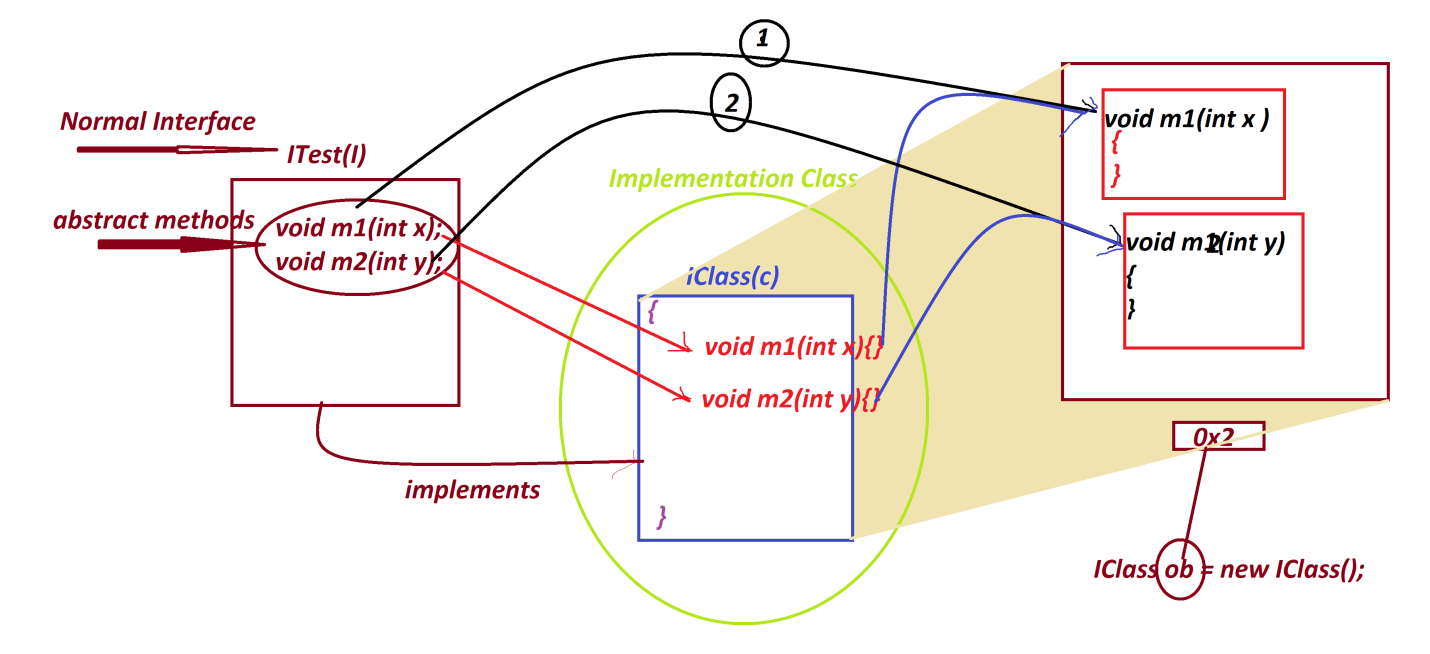
#### Model-2 : Interface without Implementation Class\_Name

**(Anonymous InnerClass as Implementation)**

**------**

**Model-1 : Interface with Implementation class with name.**

***Diagram:***

******

***ProjectName : CoreJava\_Model\_1***

|  |
| --- |
| ***//P1 : ITest.java***  **package p1;**  **public interface ITest**  **{**  **public abstract void m1(int x);**  **public abstract void m2(int y);**  **}**  ***//p1 : IClass.java***  **package p1;**  **public class IClass implements ITest**  **{**  **public void m1(int x)**  **{**  **System.out.println("\*\*\*\*\*Implemented m1(x)\*\*\*\*\*\*");**  **System.out.println("The value x:"+x);**  **}**    **public void m2(int y)**  **{**  **System.out.println("\*\*\*\*\*Implemented m2(xy\*\*\*\*\*\*");**  **System.out.println("The value y:"+y);**  **}**  **}** |

***p2 : DemoModel1.java(MainClass)***

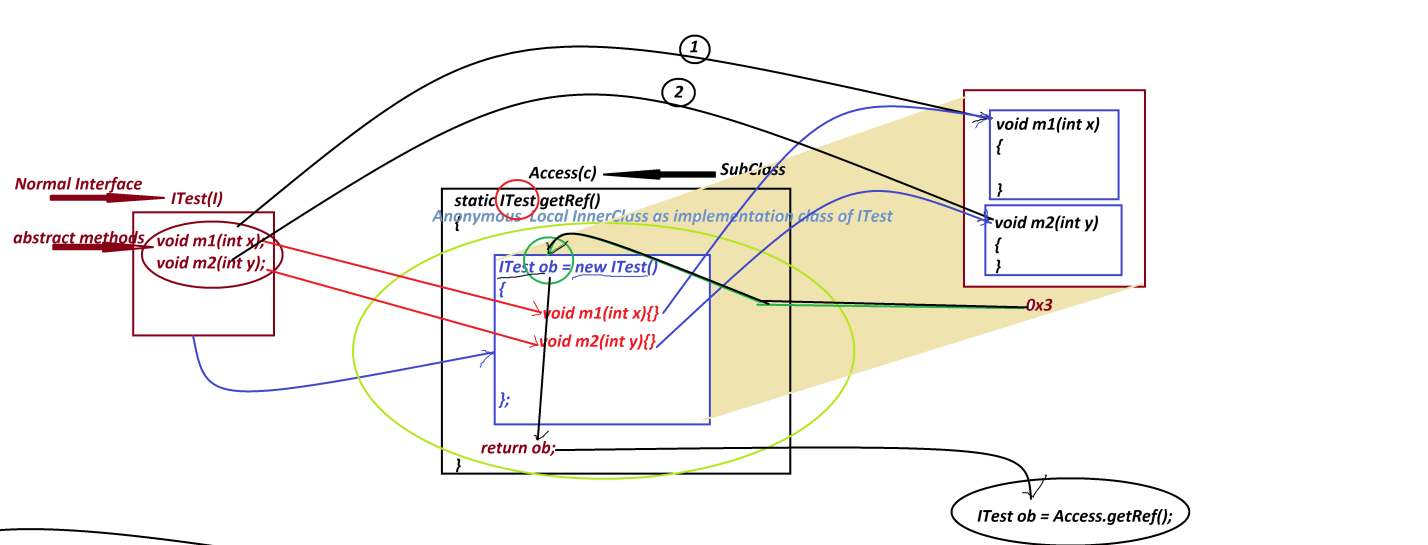
|  |
| --- |
| **package p2;**  **import p1.\*;**  **public class DemoModel1**  **{**  **public static void main(String[] args)**  **{**  **IClass ob = new IClass();//Implementation Object**  **ob.m1(11);**  **ob.m2(23);**  **}**  **}** |
| ***o/P:***  ***\*\*\*\*\*Implemented m1(x)\*\*\*\*\*\****  ***The value x:11***  ***\*\*\*\*\*Implemented m2(xy\*\*\*\*\*\****  ***The value y:23*** |

***\*imp***

***Model-2 : Interface without Implementation Class\_Name***

***(Anonymous InnerClass as Implementation)***

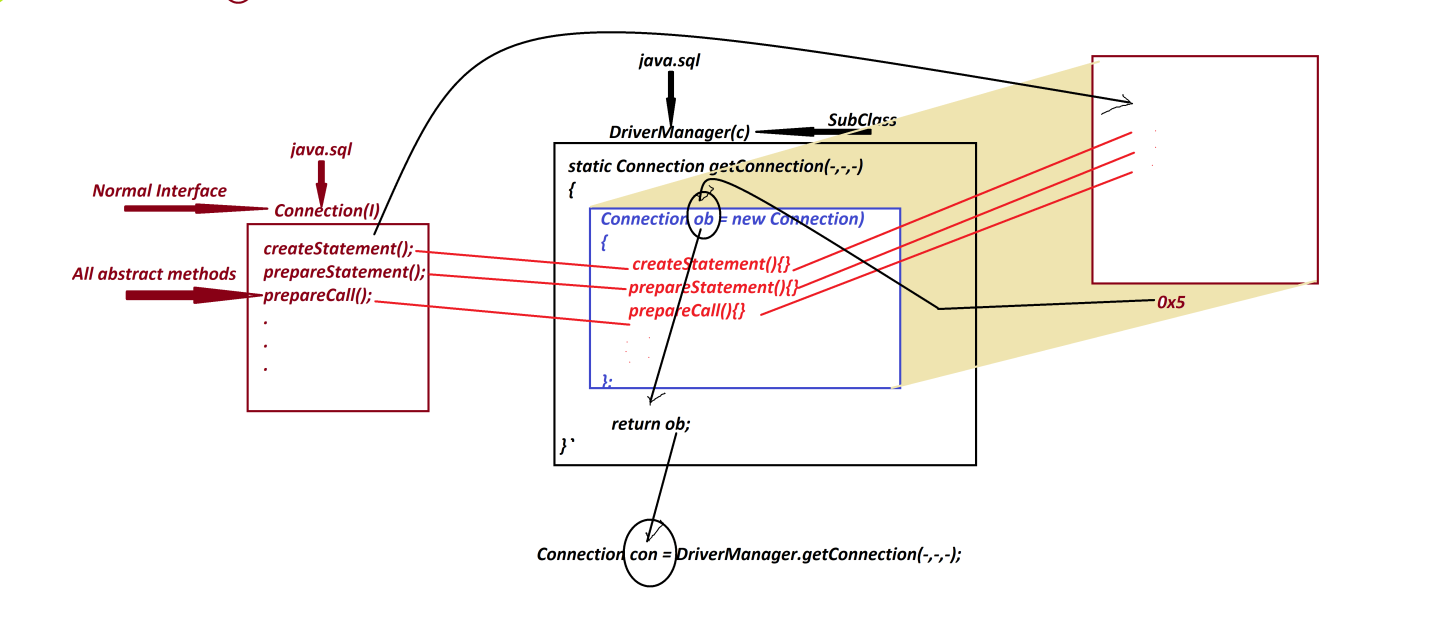
Diagram

******

***ProjectName : CoreJava\_Model\_2***

|  |
| --- |
| ***p1 : ITest.java***  **package p1;**  **public interface ITest**  **{**  **public abstract void m1(int x);**  **public abstract void m2(int y);**  **}**  ***p1 : Access.java***  **package p1;**  **public class Access**  **{**  **public static ITest getRef()**  **{**  **ITest ob = new ITest()**  **{**  **public void m1(int x)**  **{**  **System.out.println("\*\*\*\*\*Implemented m1(x)\*\*\*\*\*");**  **System.out.println("The value x:"+x);**  **}**  **public void m2(int y)**  **{**  **System.out.println("\*\*\*\*\*Implemented m2(y)\*\*\*\*\*");**  **System.out.println("The value y:"+y);**  **}**  **};**  **return ob;**  **}//OuterClass static method**  **}//OuterClass** |
| ***p2 : DemoModel2.java(MainClass)***  ***package p2;***  ***import p1.\*;***  ***public class DemoModel2***  ***{***  ***public static void main(String[] args)***  ***{***  ***ITest ob = Access.getRef();//Creating and Accessing Implementation Object***  ***ob.m1(11);***  ***ob.m2(12);***  ***}***  ***}*** |
| ***o/p:***  ***\*\*\*\*\*Implemented m1(x)\*\*\*\*\****  ***The value x:11***  ***\*\*\*\*\*Implemented m2(y)\*\*\*\*\****  ***The value y:12*** |

***Diagram representing generating 'Connection' implementation Object:***

******

*Date: 04/03/2025 (Day-6)*

***Note***

* we use **getConnection()** - method is from 'DriverManager' to create implementation Object for 'Connection' interface, because **getConnection()**- method internally holding 'Anonymous Local InnerClass as implementation class of Connection interface' and which generate Connection-Implementation Object.

------

**Method Signature of getConnection():**

**public static java.sql.Connection getConnection**

**(java.lang.String, java.lang.String, java.lang.String)**

**throws java.sql.SQLException;**

-----

Syntax

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

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

*DB-UName => system*

*DB-PWord => tiger*

**Connection con = DriverManager.getConnection**

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

-----

***\*imp***

**JDBC statements:**

* JDBC statements will specify the type of operation to be performed on DB Product.
* These JDBC statements are categorized into three types:

**1. Statement**

**2. PreparedStatement**

**3. CallableStatement**

**1.1 Statement**

* 'Statement' is an interface from java.sql package and which is used to execute normal queries without IN-Parameters.

(Normal queries means Create, Insert, Select, Update and delete)

* we use **createStatement()**-method from 'Connection' interface to create implementation object for 'Statement' interface, because this **createStatement()** -method internally holding 'Anonymous Local InnerClass as implementation class of Statement-Interface' and which generate 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' interface:

(a) **executeQuery()**

(b) **executeUpdate()**

**(a) executeQuery():**

* executeQuery()- method is used to execute select-queries Method
* **Signature of** **executeQuery()**:

**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 of*** **executeUpdate**:

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

* *Syntax*

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

**======**

***\*imp***

**Creating 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 Project-> Build Path -> Configure Build Path -> Libraries -> select 'Classpath' and click 'Add External JARs' -> Browse and select DB-Jar file from user defined folder-> Open-> Apply -> Apply and Close.

-----

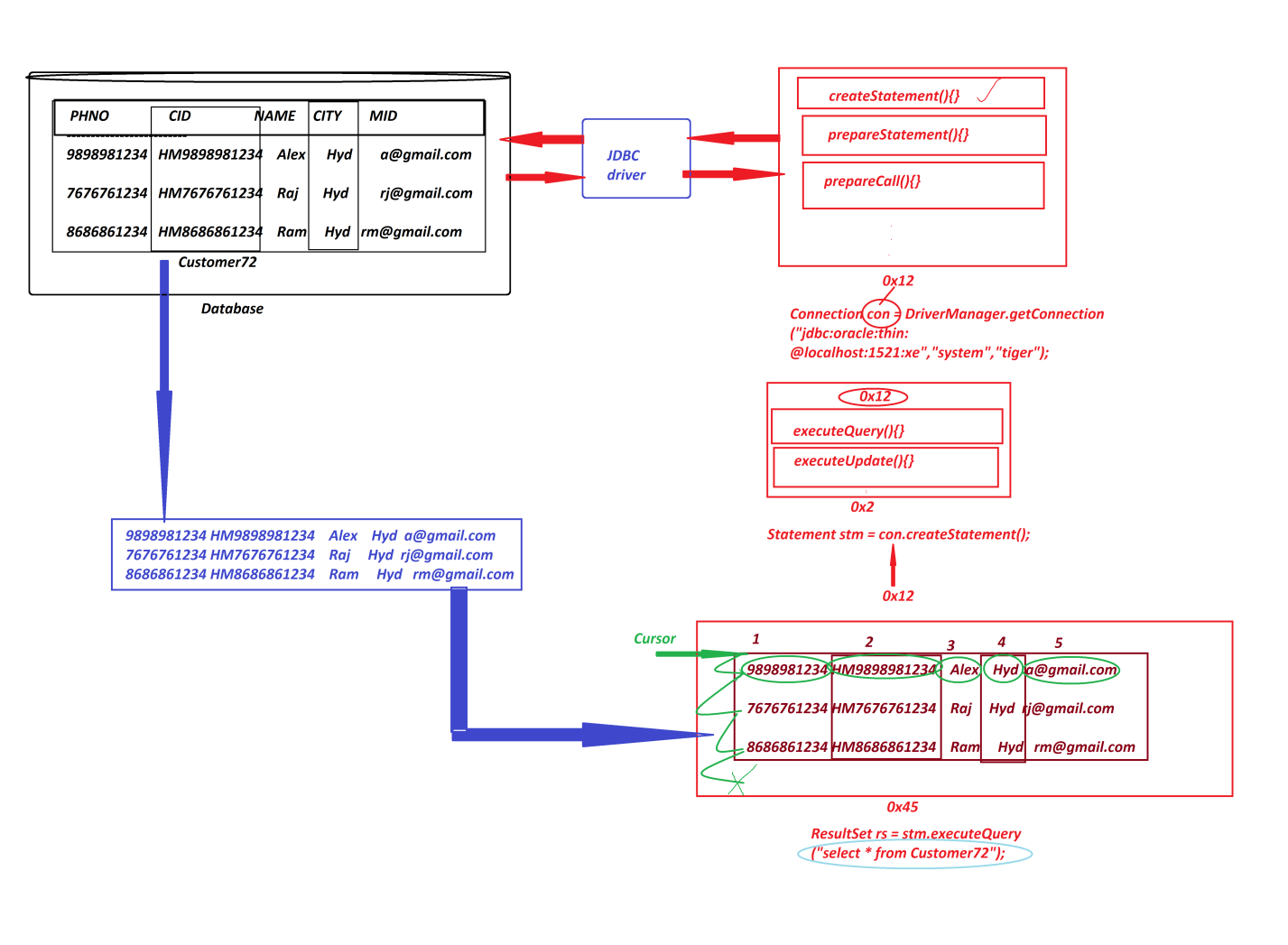
**step-4** : Create package in 'src'

-----

**step-5** : Create class(JDBC Program) in package and write JDBC-code to display all Customer details.

**======**

|  |
| --- |
| DBCon1.java -----------  package test;  import java.sql.\*;  public class DBCon1  {  public static void main(String[] args)  {  try  {  //step-1 : Loader driver  Class.forName("oracle.jdbc.driver.OracleDriver");  //step-2 : Creating Connection to Database Product  Connection con = DriverManager.getConnection  ("jdbc:oracle:thin:@localhost:1521:xe",  "system","tiger");  //step-3 : preparing JDBC-statement  Statement stm = con.createStatement();  //step-4 : Executing the query  ResultSet rs = stm.executeQuery("select \* from Customer72");  while(rs.next())  {  System.out.println(rs.getLong(1)+"\t"  +rs.getString(2)+"\t"+  rs.getString(3)+"\t"+  rs.getString(4)+"\t"+  rs.getString(5));  }//end of loop  //step-5 : Closing the connection from Database  con.close();  }//end of try  catch(Exception e)  {  e.printStackTrace();  }  }  } |

******

*Date: 05/03/2025 (Day-7)*

***Note***

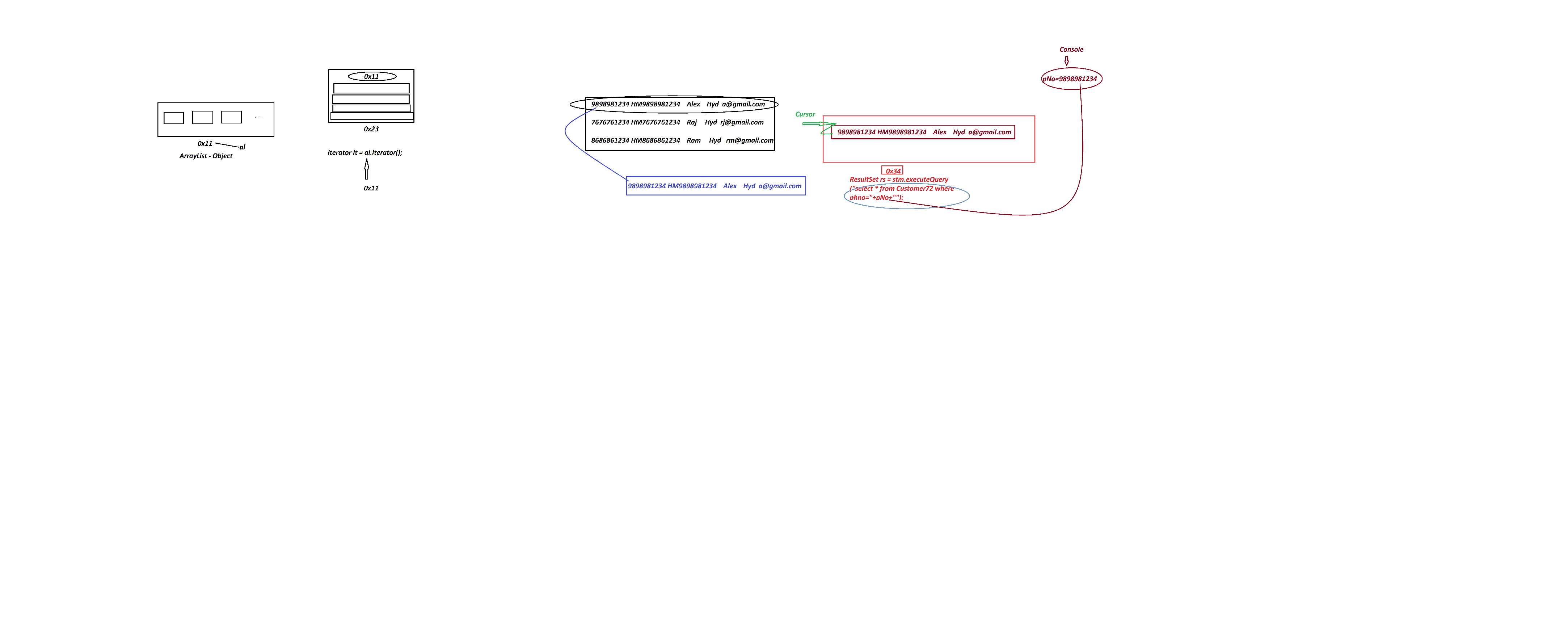
1. **'forName()'** method is from java.lang.Class and which is used to load the class at runtime or execution time, in this process the class is not identified by the Compiler at compilation stage.
2. **executeQuery()** -method will create implementation object for 'ResultSet-Interface' and the object will hold the result generated from select-queries,and the method also generate one cursor pointing before the first row.
3. we use **'next()'** method to move the cursor on ResultSet-object and which generate boolean result***.***

-----

Example

Q. Construct JDBC Application to display Customer details based on PhoneNo.

*Diagram*



***Program-* DBCon2.java**

|  |
| --- |
| package test;  import java.sql.\*;  import java.util.\*;;  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();           System.out.println("Enter the Cust-PhoneNo to display details:");           long pNo = s.nextLong();           ResultSet rs = stm.executeQuery           ("select \* from Customer72 where phno="+pNo+"");           if(rs.next()) {           System.out.println(rs.getLong(1)+"\t"           +rs.getString(2)+"\t"           +rs.getString(3)+"\t"           +rs.getString(4)+"\t"           +rs.getString(5));           }else {           System.out.println("Invalud Cust-PhNo....");           }           con.close();          }//end of try          catch(Exception e)          {           e.printStackTrace();          }  }  } |
| o/P:  **Enter the Cust-PhoneNo to display details: 9898981234**  **9898981234 HM9898981234 Alex Hyd a@gmail.com** |

Example

Q. Construct JDBC Application to read Customer details from Console and insert into Customer72

Table (Insert Operation)

#### Program : DBCon3.java

|  |
| --- |
| package test;  import java.sql.\*;  import java.util.\*;  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();          System.out.println("Enter the Cust-PhoneNO:");          long phNo = Long.parseLong(s.nextLine());          String custId = "HM"+phNo;          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();          int k = stm.executeUpdate          ("insert into Customer72 values("+phNo+",'"+custId+"','"+name+"','"+city+"','"+mId+"')");          if(k>0) {          System.out.println("Customer details added Successfully...");          }          con.close();         }catch(Exception e) {         e.printStackTrace();         }  }  } |
| o/p:  Enter the Cust-PhoneNO: 4545451234  Enter the Cust-Name: RTER  Enter the Cust-City: Hyd  Enter the Cust-MailId: r@gmail.com  Customer details added Successfully... |

----

***Assignment***

*DB Table : BookDetails72*

*(bcode,bname,bauthor,bprice,bqty)*

*primary key : bcode*

*program-1 : Construct JDBC Application to insert 5 book details*

Program-2 : Construct JDBC Application to display all book details

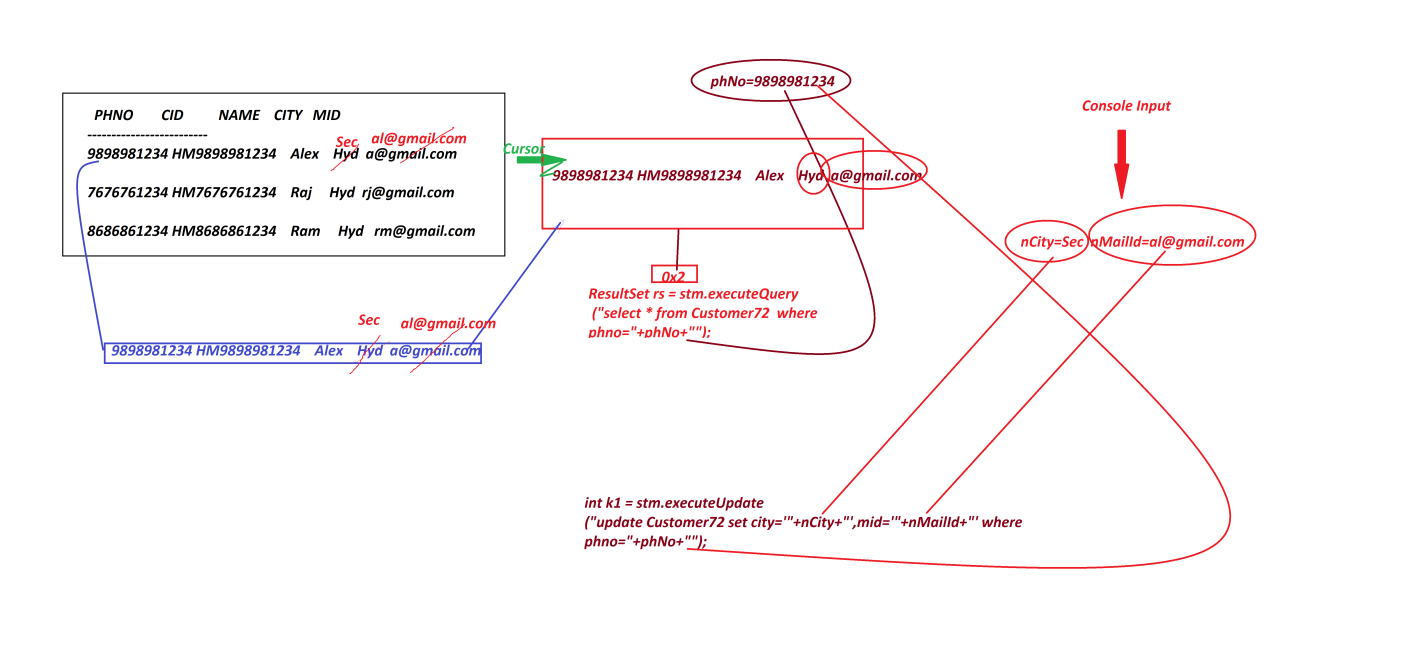
*program-3 : Construct JDBC Application to display book details based on bookCode*

[[**View Program**](https://github.com/lalitpatil891/Code-with-AdvanceJava/tree/main/PracticePrograms/BookTrack72_JDBC_app_1/src/test)]

*Date: 06/03/2025 (Day-8)*

Example

Construct JDBC Application to perform Update and Delete Operations on Customer table.

(based Customer PhoneNo) 

[[View Program](https://github.com/lalitpatil891/Code-with-AdvanceJava/blob/main/PracticePrograms/JDBC_app_5/src/test/DBcon5.java)]

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

Example

#### Construct JDBC Application to perform the following operations

1.create

2.insert

3.update

4.delete

##### Program : DBCon5.java

package test;

import java.sql.\*;

import java.util.\*;

public class DBCon5 {

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("Enter the query(create/insert/update/delete)");

String qry = s.nextLine();

int k = stm.executeUpdate(qry);

System.out.println("The value in k:"+k);

if(k>=0) {

System.out.println("Query executed Successfully...");

}

con.close();

}catch(SQLSyntaxErrorException sqe) {

System.out.println(sqe.getMessage());

System.out.println("Error Code:"+sqe.getErrorCode());

}catch(SQLIntegrityConstraintViolationException sie) {

System.out.println(sie.getMessage());

System.out.println("Error Code:"+sie.getErrorCode());

}catch(Exception e) {

e.printStackTrace();

}

}

}

o/p:(Create)

Enter the query(create/insert/update/delete)

create table emp72(id varchar2(10),name varchar2(15),desg varchar2(10),primary key(id))

The value in k:0

Query executed Successfully...

o/p:(Insert)

Enter the query(create/insert/update/delete)

insert into Emp72 values('A11','Alex','SE')

The value in k:1

Query executed Successfully...

o/p:(Update)

Enter the query(create/insert/update/delete)

update Emp72 set desg='ME' where id='A11'

The value in k:1

Query executed Successfully...

o/p:(Delete)

Enter the query(create/insert/update/delete)

delete from Emp72 where id='A21'

The value in k:1

Query executed Successfully...

o/p:

Enter the query(create/insert/update/delete)

create table Emp72(id varchar2(10),name varchar2(15),desg varchar2(10),primary key(id))

Table already avilable...

Error Code:955

955 - Create

001 – Insert

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

Assignment

*DB Table : Product72(code,name,price,qty)*

*Primary key : code*

*Construct JDBC Application to perform the following operations based on User Choice:*

*1.AddProduct*

*2.ViewAllProducts*

*3.ViewProductByCode*

*4.UpdateProductByCode(price-qty)*

*5.DeleteProductByCode*

*6.Exit*

*Note:*

*=>repeat the above choice(operations) until we perform exit-operation*

**[**[**View Program**](https://github.com/lalitpatil891/Code-with-AdvanceJava/blob/main/PracticePrograms/JDBC_Inventory_App/src/test/InventoryAppDBcon.java)**]**

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

*Date: 07/03/2025 (Day-9)*

***\*imp***

**1.2 PreparedStatement**

* 'PreparedStatement' is an interface from java.sql package and which is used to execute normal queries with IN-Parameters.
* we use parepareStatement() -method from 'Connection' interface to create implementation Object for 'PreparedStatement' interface, becuase the prepareStatement()-method internally holding 'Anonymous Local innerclass as implementation class of preparedStatement Interface and which generate PreparedStatement-Object
* 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:

**(a)** **executeQuery()**

**(b)** **executeUpdate()**

**(a) executeQuery()**

* *executeQuery()* - method is used to execute select-queries.
* Method Signature of ***executeQuery()*:**

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

* ***syntax***

**ResultSet rs = ps.executeQuery();**

**(b) executeUpdate()**

* **executeUpdate()** -method is used to execute NonSelect queries.
* Method signature of **executeUpdate()**
* **public abstract int executeUpdate() throws java.sql.SQLException;**
* ***syntax***

**int k = ps.executeUpdate();**

-----

***Note***

***executeQuery()*** *and* ***executeUpdate()*** *methods are with parameter in 'Statement' and without parameter in 'preparedStatement'.*

------

***Example :***

***(Demonstrating PreparedStatement)***

DBTable : BankCustomer72(accno,cid,cname,balance,acctype)

primary key : accno

*create table BankCustomer72(*

*accno number(15),*

*cid varchar2(15),*

*cname varchar2(15),*

*balance number(10,2),*

*acctype varchar2(15),*

*primary key(accno));*

-----

***Construct JDBC Application to perform the following operations based on Choice:***

***1.AddBankCustomer***

***2.ViewAllBankCustomers***

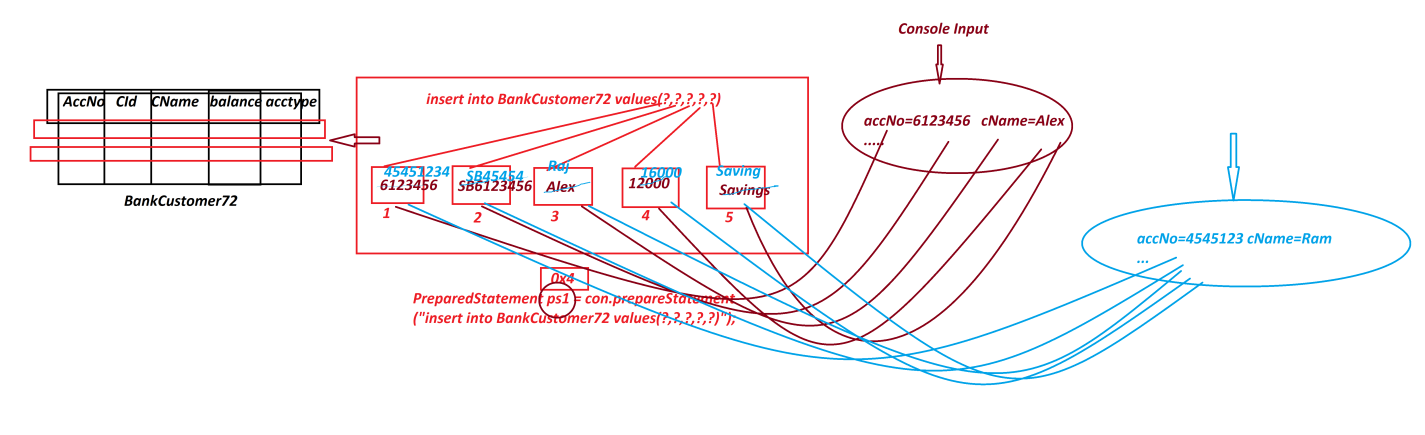
***3.Exit***

***Note: repeat the process until we perform exit operation***

***Program*** DBCon6.java

|  |
| --- |
| package test;  import java.sql.\*;  import java.util.\*;  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 ps1 = con.prepareStatement  ("insert into BankCustomer72 values(?,?,?,?,?)");  //Compilation process  PreparedStatement ps2 = con.prepareStatement  ("select \* from BankCustomer72");//Compilation Process  while(true) {  System.out.println("\*\*\*\*\*\*Operations Choice\*\*\*\*\*\*");  System.out.println("\t1.AddBankCustomer"  + "\n\t2.ViewAllBankCustomers"  + "\n\t3.Exit");  System.out.println("Enter Your Choice:");  int choice = Integer.parseInt(s.nextLine());  switch(choice) {  case 1:  //read data from console into Local variables  System.out.println("Enter the CustAccNo:");  long accNo = Long.parseLong(s.nextLine());  String cId = "SB"+accNo;  System.out.println("Enter the CustName:");  String cName = s.nextLine();  System.out.println("Enter the Cust-Balance:");  float balance = Float.parseFloat(s.nextLine());  System.out.println("Enter the Cust-AccType:");  String accType = s.nextLine();    //Load data to PreparedStatement Object using Setter methods  ps1.setLong(1, accNo);  ps1.setString(2, cId);  ps1.setString(3, cName);  ps1.setFloat(4, balance);  ps1.setString(5, accType);    int k = ps1.executeUpdate();//Execution Process  if(k>0) {  System.out.println("BankCustomer Added Successfully...");  }    break;  case 2:  ResultSet rs = ps2.executeQuery();//Execution process  while(rs.next()) {  System.out.println(rs.getLong(1)+"\t"  +rs.getString(2)+"\t"  +rs.getString(3)+"\t"  +rs.getFloat(4)+"\t"  +rs.getString(5));  }//end of loop  break;  case 3:  System.out.println("Operations Stopped...");  System.exit(0);  default:  System.out.println("Invalid Choice....");  }//end of switch  }//end of while  }catch(Exception e) {  e.printStackTrace();  }  }  } |
| *o/p:*  *\*\*\*\*\*\*Operations Choice\*\*\*\*\*\**  *1.AddBankCustomer*  *2.ViewAllBankCustomers*  *3.Exit*  *Enter Your Choice:*  *2*  *6123456 SB6123456 Alex 12000.0 Savings*  *454541234 SB454541234 Ram 16000.0 Savings*  *\*\*\*\*\*\*Operations Choice\*\*\*\*\*\**  *1.AddBankCustomer*  *2.ViewAllBankCustomers*  *3.Exit*  *Enter Your Choice:*  *3*  *Operations Stopped...* |

***Diagram***



*Date: 08/03/2025 (Day-10)*

***Example***

*Construct JDBC Application to perform the following operations on Choice based on AccNo*

*1.UpdateBankCustomer*

*2.DeleteBankCustomer*

***Program*** *DBCon7.java*

|  |
| --- |
| package test;  import java.util.\*;  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 ps1 = con.prepareStatement  ("select \* from BankCustomer72 where accno=?");  //Compilation process  PreparedStatement ps2 = con.prepareStatement  ("update BankCustomer72 set balance=? where accno=?");  //Compilation Process  PreparedStatement ps3 = con.prepareStatement  ("delete from BankCustomer72 where accno=?");  //Compilation Process  System.out.println("Enter the Cust-AccNo to perform Update/Delete operation:");  long accNo = s.nextLong();  ps1.setLong(1, accNo);  ResultSet rs = ps1.executeQuery();  if(rs.next()) {  System.out.println("\*\*\*\*\*\*\*Operation Choice\*\*\*\*\*");  System.out.println("\t1.UpdateBankCustomer"  + "\n\t2.DeleteBankCustomer");  System.out.println("Enter your Choice:");  int choice = s.nextInt();  switch(choice) {  case 1:  System.out.println("Existing balance:"+rs.getFloat(4));  System.out.println("Enter the new balance:");  float nBal = s.nextFloat();  ps2.setFloat(1, nBal);  ps2.setLong(2,accNo);  int k1 = ps2.executeUpdate();  if(k1>0) {  System.out.println("Customer Updated Successfully...");  }  break;  case 2:  ps3.setLong(1, accNo);  int k2 = ps3.executeUpdate();  if(k2>0) {  System.out.println("Customer deleted Successfully....");  }  break;  default:  System.out.println("Invalid Choice....");  }//end of switch  }else {  System.out.println("Invalid accNo....");  }  con.close();  }catch(Exception e) {  e.printStackTrace();  }  }  } |
| o/p:(Update)  Enter the Cust-AccNo to perform Update/Delete operation:  454541234  \*\*\*\*\*\*\*Operation Choice\*\*\*\*\*  1.UpdateBankCustomer  2.DeleteBankCustomer  Enter your Choice:  1  Existing balance:16000.0  Enter the new balance:  20000  Customer Updated Successfully...  o/p:(Delete)  Enter the Cust-AccNo to perform Update/Delete operation:  454541234  \*\*\*\*\*\*\*Operation Choice\*\*\*\*\*  1.UpdateBankCustomer  2.DeleteBankCustomer  Enter your Choice:  2  Customer deleted Successfully.... |
| ***Diagram*** |

**'ResultSet' in JDBC *\*imp***

* 'ResultSet' is an interface from java.sql package and which is instantiated to holdthe result generated from select-queries.
* 'ResultSet' Objects are categorized into two types:

*1.NonScrollable ResultSet Objects*

*2.Scrollable ResultSet Objects*

***1. NonScrollable ResultSet Objects:***

* In NonScrollable ResultSet Objects the cursor can be moved only in one direction, from top-of-table-data to bottom-of-table-data, which means only in forward direction.
* we use the following syntax to create NonScrollable ResultSet Object:

**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***

* In Scrollable ResultSet Objects the cursor can be moved in both directions,which means can be moved in foward and backward directions.
* we use the following syntax to create Scrollable ResultSet Object:

***syntax-1*** : Using 'Statement'

**Statement stm = con.createStatement(type, mode);**

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

***syntax-2*** : Using 'PreparedStatement'

**PreparedStatement ps =**

**con.prepareStatement(type, mode,"select-query-structure");**

**ResultSet rs = ps.executeQuery();**

*Date: 10/03/2025 (Day-11)*

**Define "type"?**

* "type" specifies the direction of Cursor on ResultSet Object.
* The following fields from ResultSet-Interface will specify the "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 to be performed on ResultSet Object.
* The following fields from ResultSet-Interface will specify the "mode":

**public static final int CONCUR\_READ\_ONLY;**

**public static final int CONCUR\_UPDATABLE;**

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

***Note***

* we use the following some important methods to control cursor on ResultSet Object:

**1. afterLast()**

**2. beforeFirst()**

**3. first()**

**4. last()**

**5. previous()**

**6. next()**

**7. absolute(int)**

**8. relative(int)**

**1. afterLast()**

* afterLast() - method will make the cursor point after the last row in ResultSet Object.

**2. beforeFirst()**

* beforeFirst() - method will make the cursor point before the first row in ResultSet Object.

**3. first()**

* first() - method will make the cursor point to the first row of ResultSet Object.

**4. last()**

* last() - method will make the cursor point to the last row of ResultSet Object.

**5. previous()**

* previous() - method is used to move the cursor in backward direction.

**6. next()**

* next() - method is used to move the cursor in forward direction.

**7. absolute(int)**

* absolute(int) - method is used to move the cursor to specified row number.

**8.relative(int)**

* relative(int) - method is used to take incre/decre value as parameter and move the cursor in forward or backward direction from current cursor position.

***Example***

|  |
| --- |
| ***Program: DBCon8.java***  package test;  import java.sql.\*;  public class DBCon8 {  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 Customer72");  System.out.println("------3rd row------");  rs1.absolute(3);  System.out.println(rs1.getLong(1)+"\t"  +rs1.getString(2)+"\t"  +rs1.getString(3)+"\t"  +rs1.getString(4)+"\t"  +rs1.getString(5));  System.out.println("------relative(-2)------");  rs1.relative(-2);  System.out.println(rs1.getLong(1)+"\t"  +rs1.getString(2)+"\t"  +rs1.getString(3)+"\t"  +rs1.getString(4)+"\t"  +rs1.getString(5));  System.out.println("------last row------");  rs1.last();  System.out.println(rs1.getLong(1)+"\t"  +rs1.getString(2)+"\t"  +rs1.getString(3)+"\t"  +rs1.getString(4)+"\t"  +rs1.getString(5));  System.out.println("------first row------");  rs1.first();  System.out.println(rs1.getLong(1)+"\t"  +rs1.getString(2)+"\t"  +rs1.getString(3)+"\t"  +rs1.getString(4)+"\t"  +rs1.getString(5));  System.out.println("\*\*\*\*\*PreparedStatement\*\*\*\*\*\*");  PreparedStatement ps = con.prepareStatement(  "select \* from BankCustomer72",  ResultSet.TYPE\_SCROLL\_INSENSITIVE,  ResultSet.CONCUR\_READ\_ONLY);  ResultSet rs2 = ps.executeQuery();  System.out.println("------reverse-----");  rs2.afterLast();  while(rs2.previous()) {  System.out.println(rs2.getLong(1)+"\t"  +rs2.getString(2)+"\t"  +rs2.getString(3)+"\t"  +rs2.getString(4)+"\t"  +rs2.getString(5));  }//end of loop  }catch(Exception e) {  e.printStackTrace();  }  }  } |
| o/p:  \*\*\*\*\*\*\*Statement\*\*\*\*\*\*  ------3rd row------  4545451234 HM4545451234 RTER Hyd r@gmail.com  ------relative(-2)------  9898981234 HM9898981234 Alex Sec al@gmail.com  ------last row------  4545451234 HM4545451234 RTER Hyd r@gmail.com  ------first row------  9898981234 HM9898981234 Alex Sec al@gmail.com  \*\*\*\*\*PreparedStatement\*\*\*\*\*\*  ------reverse-----  321321 SB321321 Raj 10000 Savings  6123456 SB6123456 Alex 12000 Savings |

***Assignment***

*DB Table: Employee72(eid,ename,edesg,bsal,hra,da,totsal)*

*Primary Key : eid*

*Construct JDBC Application to perform the following Operations based on Choice:*

*1.AddEmployee*

*2.ViewAllEmployees*

*3.ViewEmployeeByCode*

*4.UpdateEmployeeById(bSal)*

*5.DeleteEmployeeById*

*6.Exit*

*Calcutions:*

*hra = 91% of bSal*

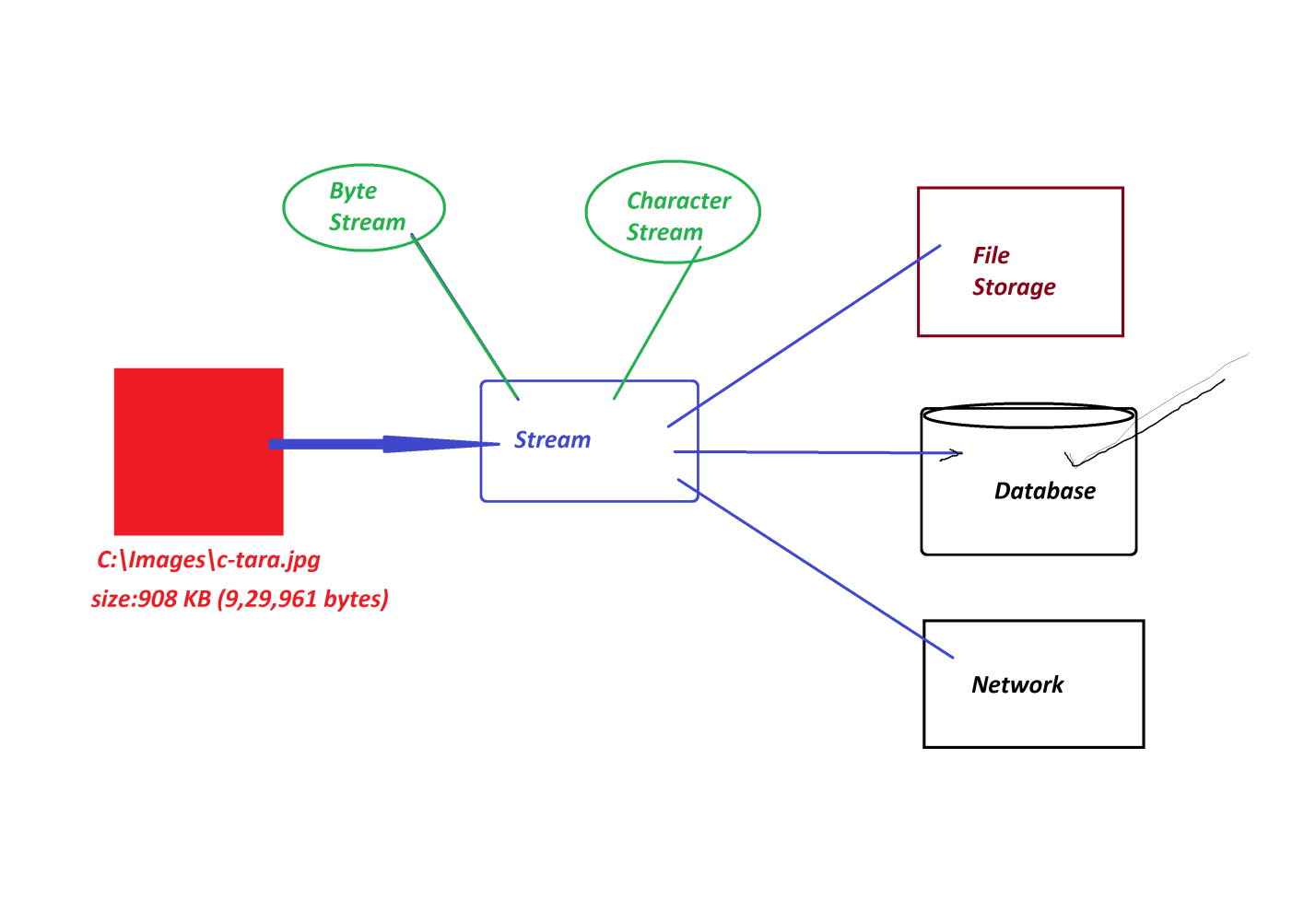
*da = 63% of bSal*

*totSal = bSal+hra+da*

*Exception: Min bSal must be 12000/-,else raise the exception*

**[**[**View Program**](https://github.com/lalitpatil891/Code-with-AdvanceJava/tree/main/PracticePrograms/JDBC_app_Employee72/src/test)**]**

*Date: 11/03/2025 (Day-12)*



***\*imp***

***Streams with Database product***

**Define stream? (Normal definition)**

* The continuous flow of data is known as stream.

***Types of streams***

* Java language support two types of streams:

**1.Byte Stream (Binary Stream)**

**2.Character Stream**

**1. Byte Stream(Binary Stream)**

* The continuous flow of data in the form of 8-bits is known as Byte Stream or Binary Stream.
* Through Byte Stream we can send all Multi-Media data formats,which means Text, Audio, Video, Image and Animation.

**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,and which is 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.

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

***\*imp***

**Storing Stream data to Database product**

**Step-1 :**

Create DB Table with name StreamTab72(id, name, mfile)

create table StreamTab72(id varchar2(10), name varchar2(15), mfile BLOB, primary key(id));

**Step-2 :** Construct JDBC Application to store Image to Database product

|  |
| --- |
| **Program : DBCon9.java**  package test;  import java.util.\*;  import java.io.\*;  import java.sql.\*;  public class DBCon9  {  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 StreamTab72 values(?,?,?)");  System.out.println("Enter the User-Id:");  String id = s.nextLine();  System.out.println("Enter the User-Name:");  String name = s.nextLine();  System.out.println("Enter the Location(fPath&fName) of User-Image(Source)");  String path = s.nextLine();  File f = new File(path);  if(f.exists()) {  FileInputStream fis = new FileInputStream(path);  ps.setString(1,id);  ps.setString(2,name);  ps.setBinaryStream(3, fis, f.length());  int k = ps.executeUpdate();  if(k>0) {  System.out.println("Image Stored Successfully...");  }  fis.close();  }else {  System.out.println("Invalid fPath or fName....");  }  con.close();  }catch(Exception e) {  e.printStackTrace();  }  }  } |
| ***o/p:***  Enter the User-Id:  A121  Enter the User-Name:  Alex  Enter the Location(fPath&fName) of User-Image(Source)  C:\Images\c-tara.jpg  Image Stored Successfully... |
| ***Diagram*** |

*Date: 12/03/2025 (Day-13)*

***Retrieving Stream(Image) from Database Product***

|  |
| --- |
| **Program : DBCon10.java**  package test;  import java.util.\*;  import java.io.\*;  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");  PreparedStatement ps = con.prepareStatement  ("select \* from StreamTab72 where id=?");  System.out.println("Enter the User-Id to retrive details:");  String id = s.nextLine();  ps.setString(1, id);  ResultSet rs = ps.executeQuery();  if(rs.next()) {  Blob b = rs.getBlob(3);  byte by[] = b.getBytes(1, (int)b.length());  System.out.println("User-Id:"+rs.getString(1));  System.out.println("User-Name:"+rs.getString(2));  System.out.println("Enter the lacation(fPath&fName-destination) to store Stream: ");  String path = s.nextLine();  FileOutputStream fos = new FileOutputStream(path);  fos.write(by);  System.out.println("Stream stored to specified location....");  fos.close();  }else {  System.out.println("Invalid User Id....");  }  }catch(Exception e) {  e.printStackTrace();  }  }  } |
| o/p:  Enter the User-Id to retrive details:  A121  User-Id:A121  User-Name:Alex  Enter the lacation(fPath&fName-destination) to store Stream:  E:\Images\XYZ.jpg  Stream stored to specified location.... |
| ***Diagram*** |

***FAQ***

**Define** **'FileInputStream'?**

* ‘FileInputStream' is a class from java.io package and which is instantiated to link(open) the file to read Byte-Stream data.
* ***Syntax***

**FileInputStream fis = new FileInputStream("fPath&fName");**

------

***FAQ***

**Define 'FileOutputStream'?**

* 'FileOutputStream' is a class from java.io package and which is instantiated to create a new file with 0KB and links(opens) the file to write Byte-stream data.
* ***Syntax***

**FileOutputStream fos = new FileOutputStream("fPath&fName");**

------

***FAQ***

**Define setBinaryStream() method?**

* setBinaryStream()-method belongs to PreparedStatement and which links stream to parameter-index-field of PreparedStatement Object.
* Through this method we must specify para-index-no,location-of-stream and length-of-stream.
* ***Syntax***

**ps.setBinaryStream(3, fis, f.length());**

------

***FAQ***

**define getBlob() method?**

* getBlob()-method is from ResultSet and which is used to instantiate Blob-Interface and this Blob-Object internally linked to Stream-column of ResultSet.
* ***Syntax***

**Blob b = rs.getBlob(3);**

**-------**

***FAQ*** ***\*imp***

**Define getBytes() method?**

* getBytes()-method is from Blob,and which is used to convert stream into byte-Array.
* ***Syntax***

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

**------**

***\*imp***

**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',because the prepareCall()-method internally holding 'Anonymous Local InnerClass as implementation class of CallableStatement' and generate CallableStatement Object.
* Method Signature of **prepareCall():**

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

* ***Syntax***

**CallableStatement cs = con.prepareCall("{call Proce\_name/Func\_name}");**

**-----**

*Date: 12/03/2025 (Day-13)*

***What is the difference***

***(i)function***

***(ii)member function***

***(iii)method***

**(i) function**

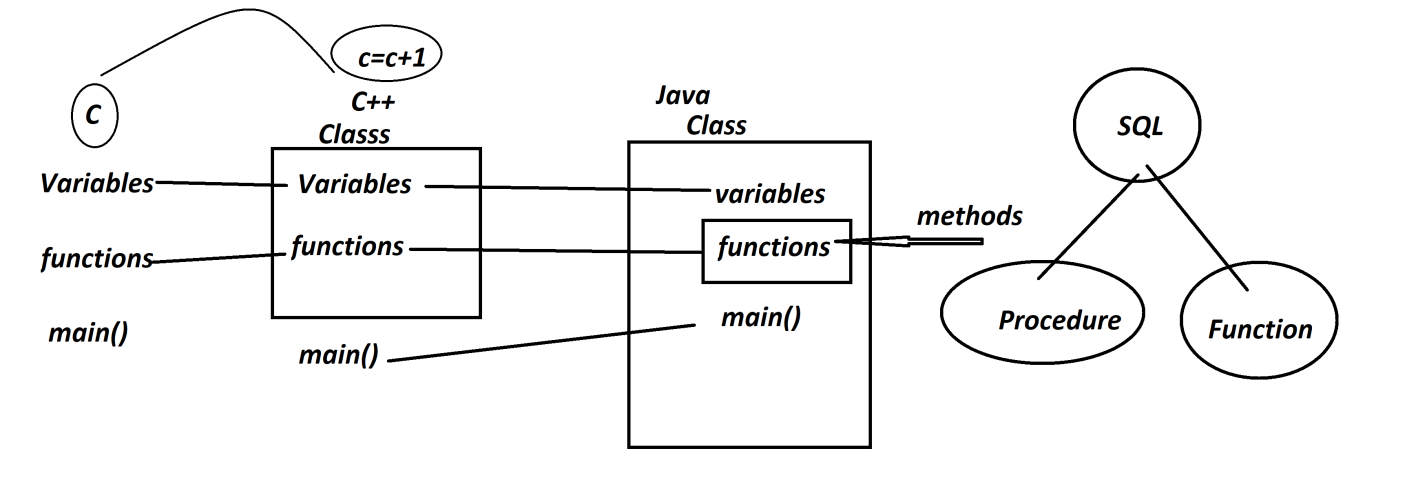
* The part of program which is executed outof main()-program is known as 'function' in C-Lang.

**(ii) member function**

* The functions which are declared as members of class in C++ are known as member functions.
* These member functions can be declared inside the class or Outside the class with class reference.

**(iii) method**

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



-----

***FAQ***

**Define 'Procedure'?**

* Procedure is a set-of-queries executed on Database product at-a-time, and after execution Procedure will not return any value.

(Procedure means NonReturn\_type)

*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:

***(a) IN-Parameter Procedure***

***(b) OUT-Parameter Procedure***

***(c) IN-OUT-Parameter Procedure***

**(a) IN-Parameter Procedure**

* The Procedures which take the data from Java-Program and sent to database product are known as IN-Parameter Procedures.

**(b)** **OUT-Parameter Procedure**

* The Procedures which take the data from database product and sent to Java-Program are known as OUT-Parameter Procedures.

**(c) IN-OUT-Parameter Procedure**

* The procedures which perform both operations are known as IN-OUT-Parameter Procedures.

***FAQ***

**Define 'Function'?**

* Function is a set-of-queries executed on Database product at-a-time,and after execution it will return the value.
* we use 'return' statement to return the value from Function.

*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 Application to demonstrate Procedure**

**Step-1 :** *Create the following tables*

EmpData72(eid,ename,edesg);

EmpAddress72(eid,hno,sname,city,state,pincode);

EmpContact72(eid,mid,phno);

EmpSalary72(eid,bsal,hra,da,totsal);

-----

create table EmpData72(eid varchar2(10),ename varchar2(15),edesg varchar2(10), primary key(eid));

-----

create table EmpAddress72(eid varchar2(10), hno varchar2(15), sname varchar2(15), city varchar2(15), state varchar2(15), pincode number(10), primary key(eid));

-----

create table EmpContact72(eid varchar2(10), mid varchar2(25), phno number(15), primary key(eid));

-----

create table EmpSalary72(eid varchar2(10), bsal number(10), hra number(10,2), da number(10,2), totsal number(10,2), primary key(eid));

-----

**Step-2 :** *Construct Procedure to perform Insert operation to Employee tables*

create or replace procedure InsertEmployee72

(id varchar2, en varchar2, ed varchar2, hn varchar2, sn varchar2, cty varchar2, st varchar2, pcode number, md varchar2, pno number, bs number, h number, d number, ts number) is

begin

insert into EmpData72 values(id, en, ed);

insert into EmpAddress72 values(id, hn, sn, cty, st, pcode);

insert into EmpContact72 values(id, md, pno);

insert into EmpSalary72 values(id, bs, h, d, ts);

end;

/

------

**Step-3 :** *Construct JDBC Application to execute Procedure*

|  |
| --- |
| ***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");  CallableStatement cs = con.prepareCall  ("{call InsertEmployee72(?,?,?,?,?,?,?,?,?,?,?,?,?,?)}");  System.out.println("Enter the Emp-Id:");  String eId = s.nextLine();  System.out.println("Enter the Emp-Name:");  String eName = s.nextLine();  System.out.println("Enter the Emp-Desg:");  String eDesg = s.nextLine();  System.out.println("Enter the Emp-HNo:");  String hNo = s.nextLine();  System.out.println("Enter the Emp-SName:");  String sName = s.nextLine();  System.out.println("Enter the Emp-City:");  String city = s.nextLine();  System.out.println("Enter the Emp-State:");  String state = s.nextLine();  System.out.println("Enter the Emp-PinCode:");  int pinCode = Integer.parseInt(s.nextLine());  System.out.println("Enter the Emp-MailId:");  String mId = s.nextLine();  System.out.println("Enter the Emp-PhNo:");  long phNo = s.nextLong();  System.out.println("Enter the Emp-bSal:");  int bSal = s.nextInt();  float hra = 0.93F\*bSal;  float da = 0.61F\*bSal;  float totSal = bSal+hra+da;  cs.setString(1, eId);  cs.setString(2, eName);  cs.setString(3, eDesg);  cs.setString(4, hNo);  cs.setString(5, sName);  cs.setString(6, city);  cs.setString(7, state);  cs.setInt(8, pinCode);  cs.setString(9, mId);  cs.setLong(10, phNo);  cs.setInt(11, bSal);  cs.setFloat(12, hra);  cs.setFloat(13, da);  cs.setFloat(14, totSal);  cs.execute();//Execute Procedure  System.out.println("Employee added Successsfully....");  con.close();  }catch(Exception e) {  e.printStackTrace();  }  }  }  ***o/p***  Enter the Emp-Id:  T121  Enter the Emp-Name:  Alex  Enter the Emp-Desg:  TE  Enter the Emp-HNo:  12-34/h  Enter the Emp-SName:  SRN  Enter the Emp-City:  Hyd  Enter the Emp-State:  TG  Enter the Emp-PinCode:  506112  Enter the Emp-MailId:  alex@gmail.com  Enter the Emp-PhNo:  9898981234  Enter the Emp-bSal:  45000  Employee added Successsfully.... |
| Diagram: |

*Date: 15/03/2025 (Day-15)*

**Construct Application to retrieve all details of Employee using Procedure:**

**step-1:** *Construct the Procedure to retrive Employee details based on Emp-Id*

create or replace procedure RetrieveEmployee72

(id varchar2, en OUT varchar2, ed OUT varchar2, hn OUT varchar2, sn OUT varchar2,

cty OUT varchar2, st OUT varchar2, pcode OUT number, md OUT varchar2, pno OUT number,

bs OUT number, h OUT number, d OUT number, ts OUT number) is

begin

select ename,edesg into en,ed from EmpData72 where eid=id;

select hno,sname,city,state,pincode into hn,sn,cty,st,pcode from EmpAddress72 where eid=id;

select mid,phno into md,pno from EmpContact72 where eid=id;

select bsal,hra,da,totsal into bs,h,d,ts from EmpSalary72 where eid=id;

end;

/

**Step-2 :** *Construct JDBC Application to execute Procedure*

|  |
| --- |
| Program : DBCon12.java  package test;  import java.util.\*;  import java.sql.\*;  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");  CallableStatement cs = con.prepareCall  ("{Call RetrieveEmployee72(?,?,?,?,?,?,?,?,?,?,?,?,?,?)}");  System.out.println("Enter the Emp-Id to retrieve details:");  String eId = s.nextLine();  cs.setString(1, eId);  cs.registerOutParameter(2, Types.VARCHAR);  cs.registerOutParameter(3, Types.VARCHAR);  cs.registerOutParameter(4, Types.VARCHAR);  cs.registerOutParameter(5, Types.VARCHAR);  cs.registerOutParameter(6, Types.VARCHAR);  cs.registerOutParameter(7, Types.VARCHAR);  cs.registerOutParameter(8, Types.INTEGER);  cs.registerOutParameter(9, Types.VARCHAR);  cs.registerOutParameter(10, Types.BIGINT);  cs.registerOutParameter(11, Types.INTEGER);  cs.registerOutParameter(12, Types.FLOAT);  cs.registerOutParameter(13, Types.FLOAT);  cs.registerOutParameter(14, Types.FLOAT);  cs.execute();  System.out.println("\*\*\*\*\*\*Details\*\*\*\*\*\*");  System.out.println("Emp-Id:"+eId);  System.out.println("Emp-Name:"+cs.getString(2));  System.out.println("Emp-Desg:"+cs.getString(3));  System.out.println("Emp-HNo:"+cs.getString(4));  System.out.println("Emp-SName:"+cs.getString(5));  System.out.println("Emp-City:"+cs.getString(6));  System.out.println("Emp-State:"+cs.getString(7));  System.out.println("Emp-PinCode:"+cs.getInt(8));  System.out.println("Emp-MailId:"+cs.getString(9));  System.out.println("Emp-PhoneNo:"+cs.getLong(10));  System.out.println("Emp-BSal:"+cs.getInt(11));  System.out.println("Emp-HRA:"+cs.getFloat(12));  System.out.println("Emp-DA:"+cs.getFloat(13));  System.out.println("Emp-TotSal:"+cs.getFloat(14));  con.close();  }catch(Exception e) {  e.printStackTrace();  }  }  } |
| o/p:  Enter the Emp-Id to retrieve details:  T121  \*\*\*\*\*\*Details\*\*\*\*\*\*  Emp-Id:T121  Emp-Name:Alex  Emp-Desg:TE  Emp-HNo:12-34/h  Emp-SName:SRN  Emp-City:Hyd  Emp-State:TG  Emp-PinCode:506112  Emp-MailId:alex@gmail.com  Emp-PhoneNo:9898981234  Emp-BSal:45000  Emp-HRA:41850.0  Emp-DA:27450.0  Emp-TotSal:114300.0 |

------

***Assignment:***

*DT Tables:*

*StudentData72(rollno,name,branch)*

*StudentAddress72(rollno,hno,sname,city,state,pincode)*

*StudentContact72(rollno,mid,phno)*

*StudentMarks72(rollno,t,h,e,m,s,so)*

*StudentResult72(rollno,totmarks,per,grade)*

*Application-1:*

*Step-1 : Construct Procedure to insert student deatails to DB Tables*

*Step-2 : Construct JDBC Application to execute Procedure*

*Application-2 :*

*Step-1 : Construct Procedure to retrieve student details based on rollNo*

*Step-2 : Construct JDBC Application to execute Procedure*

#pending

*------*

*Date: 17/03/2025 (Day-16)*

**Construct Application demonstrating Function in JDBC**

**step-1:** *Construct Function to retrieve Employee TotSal based in emp-Id*

create or replace Function RetrieveTotSal72

(id varchar2) return number as ts number;

begin

select totsal into ts from EmpSalary72 where eid=id;

return ts;

end;

/

**step-2:** *Construct JDBC Application to execute function.*

|  |
| --- |
| **Program : DBCon13.java**  package test;  import java.util.\*;  import java.sql.\*;  public class DBCon13 {  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 ?:=RetrieveTotSal72(?)}");  System.out.println("Enter the Emp-Id to retrieve TotSal:");  String eId = s.nextLine();  cs.setString(2, eId);  cs.registerOutParameter(1, Types.FLOAT);  cs.execute();  System.out.println("\*\*\*\*\*\*Details\*\*\*\*\*\*");  System.out.println("Emp-Id:"+eId);  System.out.println("TotSal:"+cs.getFloat(1));  con.close();  }catch(Exception e) {  e.printStackTrace();  }  }  } |
| o/p:  Enter the Emp-Id to retrieve TotSal:  T121  \*\*\*\*\*\*Details\*\*\*\*\*\*  Emp-Id: T121  TotSal:114300.0 |

-----------------------------------------------------------------------------------***Assignment***

Construct and Execute Function to retrieve Student Percentage based on RollNo. #pending

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

***FAQ***

**Define registerOutParameter()-method?**

* registerOutParameter()-method is from 'CallableStatement' and which specify the type of data recored(loaded) to the Parameter-Index-field of CallableStatement-Object.
* ***syntax***

**cs.registerOutParameter( 1, Types.FLOAT);**

***Note***

"Types" in JDBC is a class from java.sql package and which specify the SQL-TYPE used in registerOutParameter()-method.

------

***\*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 the statements in Tranasction are executed at-a-time or not-at-all,is known as Atomicity.

***C – Consistency***

* The process in which the selected state of resources remain same until the Transaction is complemented,is known as Consistency.

***I – Isolation***

* The process in which multiple users are executed independently is known as Isolation.

***D – Durability***

* The process in which recording the state of transaction and making it available for Customers,is known as Durability

-------

*Date: 18/03/2025 (Day-17)*

***FAQ***

**Define Transaction Management?**

* The process of controlling the Transaction from starting to ending is known as Transaction Management.
* We use the following methods in Transaction Management:

***(a) getAutoCommit()***

***(b) setAutoCommit()***

***(c) setSavepoint()***

***(d) releaseSavepoint()***

***(e) commit()***

***(f) rollback()***

---

**(a) getAutoCommit()**

* getAutoCommit() - method is used to know the status of Commit-Operation.
* ***Syntax***

**boolean b = con.getAutoCommit();**

---

**(b) setAutoCommit()**

* setAutoCommit() - method is used to stop the commit-operation.
* ***Syntax***

**con.setAutoCommit(false);**

---

**(c) setSavepoint()**

* setSavepoint() - method is used to take one savepoint to perform rollback operation.
* ***Syntax***

**Savepoint sp = con.setSavepoint();**

---

**(d) releaseSavepoint()**

* releaseSavepoint() - method is used to delete the savepoint.
* ***Syntax***

**con.releaseSavepoint(sp);**

---

**(e) commit()**

* commit() - method is used to save the data permanently to Database.
* ***Syntax***

**con.commit();**

---

**(f) rollback()**

* rollback() - method will reset the buffers and moves the control to savepoint, when the transaction is failed.
* ***Syntax***

**con.rollback();**

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

***Example***

*DB Table : Bank72(accNo,name,bal,accType)*

*Primary Key:accNo*

*create table Bank72(accno number(15),name varchar2(15),bal number(10,2),accType varchar2(15),*

*primary key(accno));*

*insert into Bank72 values(6123456,'Alex',12000,'Savings');*

*insert into Bank72 values(313131,'Ram',500,'Savings');*

*ACCNO NAME BAL ACCTYPE*

*---------- --------------- ---------- ---------------*

*6123456 Alex 12000 Savings*

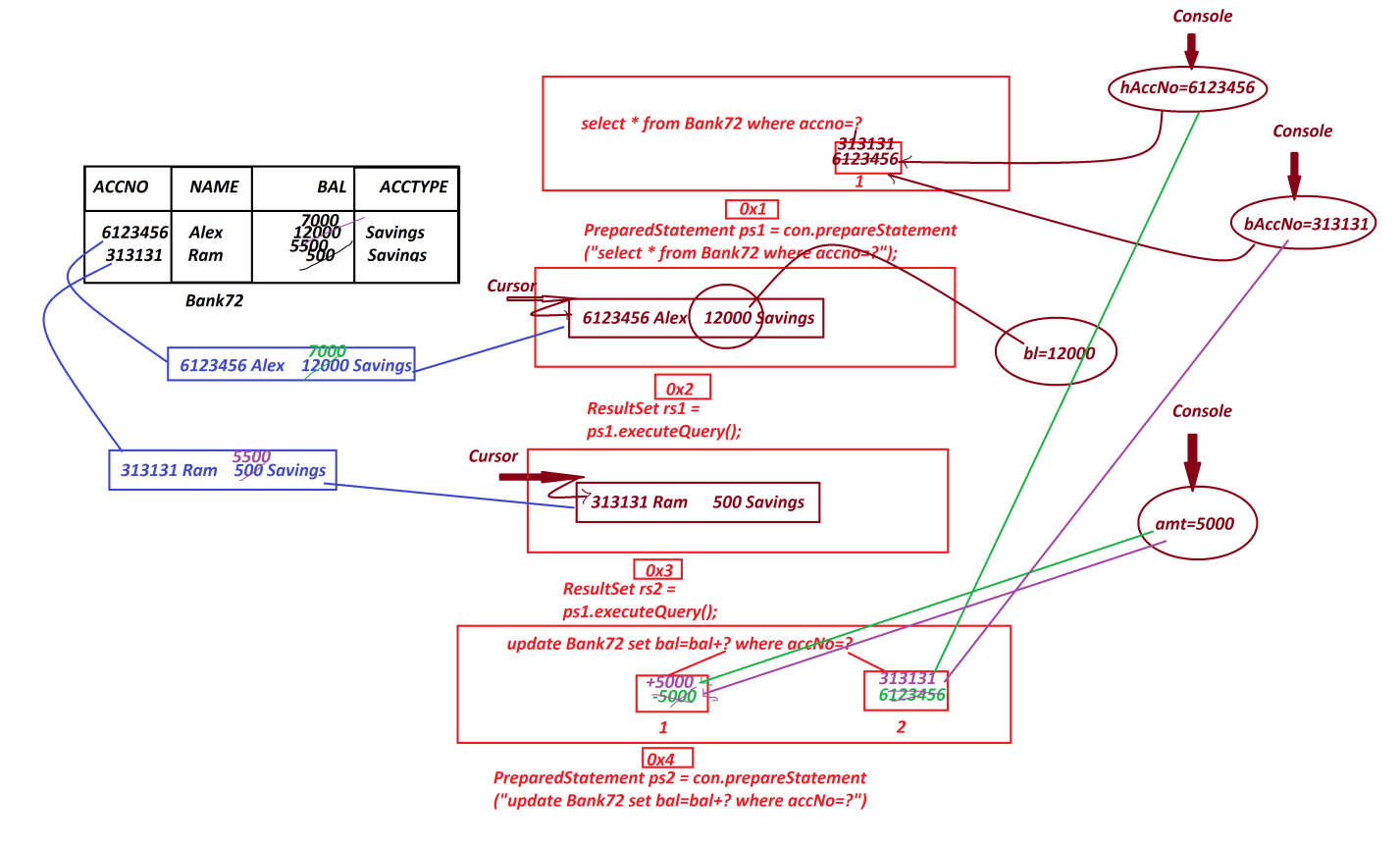
*313131 Ram 500 Savings*

*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*

***Layout***



|  |
| --- |
| **Program: DBCon14.java**  package test;  import java.util.\*;  import java.sql.\*;;  public class DBCon14 {  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 Bank72 where accno=?");  PreparedStatement ps2 = con.prepareStatement  ("update Bank72 set bal=bal+? where accNo=?");  System.out.println("Commit-Status : "+con.getAutoCommit());  con.setAutoCommit(false);//Auto-Commit-Operation-Stopped  System.out.println("Commit-Status : "+con.getAutoCommit());  System.out.println("Enter the Home-AccNo:");  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 Amount 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();//Updated in Buffer    //Statement-2 : Add amt:5000/- to accNo:313131  ps2.setFloat(1, +amt);  ps2.setLong(2, bAccNo);  int q = ps2.executeUpdate();//Updated in Buffer    if(p==1 && q==1) {  System.out.println("Transaction Successfull...");  con.commit();  }else {  System.out.println("Transaction failed....");  con.rollback();  }  }else {  System.out.println("InSufficient Fund...");  }  }else {  System.out.println("Invalid bAccNo...");  }  }else {  System.out.println("Invalid HomeAccNo...");  }  con.close();  }catch(Exception e) {  e.printStackTrace();  }  }  } |
| **o/p**  Commit-Status : true  Commit-Status : false  Enter the Home-AccNo:  6123456  Enter the benificieryAccNo:  313131  Enter the Amount to be Transferred:  5000  Transaction Successfull...  ------------------------------------------------------  ACCNO NAME BAL ACCTYPE  ---------- --------------- ---------- ---------------  6123456 Alex 7000 Savings  313131 Ram 5500 Savings |

*Date: 19/03/2025 (Day-18)*

##### Batch Processing in JDBC

* The process of collecting multiple queries as batch and executing on Database product at-a-time, is known as Batch Processing.
* Batch Processing Support only NonSelect queries, and which is also known as Batch Update processing.
* In real time, the Batch Processing is performed using 'Statement'.
* The following are some important methods related to barch processing:

***(a) addBatch()***

***(b) executeBatch()***

***(c) clearBatch()***

---

**(a) addBatch()**

* addBatch() - method is used to add NonSelect-query to a batch.
* *Method Signature*

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

* ***syntax***

**stm.addBatch("NonSelect-Query");**

**---**

**(b) executeBatch()**

* executeBatch()-method is used to execute all queries from the batch at-a-time.
* *Method Signature*

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

* ***syntax***

**int k[] = stm.executeBatch();**

---

**(c) clearBatch()**

* clearBatch()-method will delete all queries from the batch and destroys the batch.
* *Method Signature*

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

* ***syntax***

**stm.clearBatch();**

***Example***

|  |
| --- |
| **Program : DBCon15.java**  package test;  import java.util.\*;  import java.sql.\*;  public class DBCon15 {  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("\*\*\*\*\*\*Insert on Bank72\*\*\*\*\*\*");  System.out.println("Enter the AccNo:");  long accNo = Long.parseLong(s.nextLine());  System.out.println("Enter the Name:");  String name = s.nextLine();  System.out.println("Enter the Balance:");  float bal = Float.parseFloat(s.nextLine());  System.out.println("Enter the AccType:");  String accType = s.nextLine();  stm.addBatch  ("insert into Bank72 values("+accNo+",'"+name+"',"+bal+",'"+accType+"')");    System.out.println("\*\*\*\*\*delete on Customer72\*\*\*\*\*");  System.out.println("Enter the Cust-PhoneNo to delete the details:");  long pNo = Long.parseLong(s.nextLine());  stm.addBatch  ("delete from Customer72 where phno="+pNo+"");    int k[] = stm.executeBatch();  for(int i : k)  {  System.out.println("query executed : "+i);  }//end of loop  stm.clearBatch();  }catch(Exception e) {  e.printStackTrace();  }  }  } |
| o/p:  \*\*\*\*\*\*Insert on Bank72\*\*\*\*\*\*  Enter the AccNo:  434343123  Enter the Name:  RTE  Enter the Balance:  1200  Enter the AccType:  Savings  \*\*\*\*\*delete on Customer72\*\*\*\*\*  Enter the Cust-PhoneNo to delete the details:  9898981234  query executed : 1  query executed : 1 |

***FAQ***

**What is the difference b/w**

**(i) Batch Processing**

**(ii) Procedures**

* Batch Processing will execute only NonSelect-Queries,but using Procedures we can execute both 'Select and NonSelect Queries'
* when we use Batch Processing the execution load is on Server,but when we use Procedures then execution load is on Database.

-----

***FAQ***

**Define MetaData?**

* The data which is holding information about other data is known as MetaData.
* JDBC will provide the following MetaData components:

***1. DatabaseMetaData***

***2. ParameterMetaData***

***3. ResultSetMetaData***

**1. DatabaseMetaData**

* DatabaseMetaData is an interface from java.sql package and which is instantiated to hold information about Connection-Object.
* ***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.
* ***Synatax***

**ParameterMetaData pmd = ps.getParameterMetaData();**

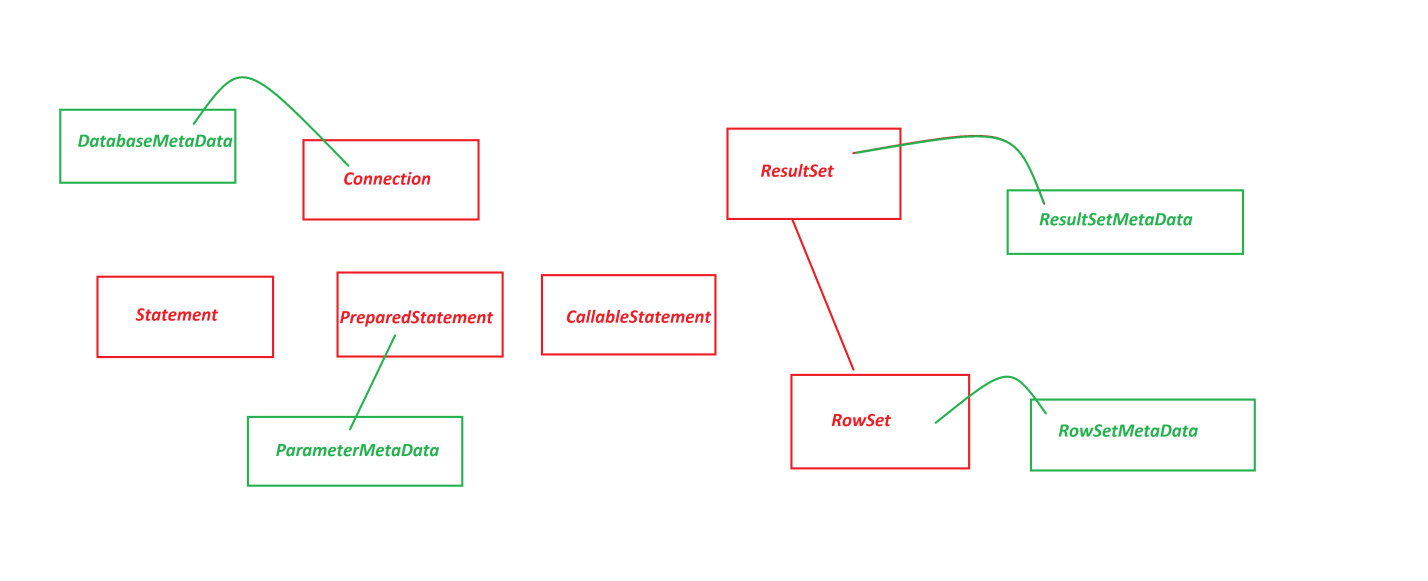
**3. ResultSetMetaData**

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

***Syntax***

**ResultSetMetaData rsmd = rs.getMetaData();**

Diagram



Date: 20/03/2025 (Day-19)

***Example***

|  |
| --- |
| Program : DBCon16.java  package test;  import java.sql.\*;  public class DBCon16 {  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("Driver Name:"+dmd.getDriverName());  System.out.println("Driver Version:"+dmd.getDriverMajorVersion());  System.out.println("\*\*\*\*\*ParameterMetaData\*\*\*\*\*\*");  PreparedStatement ps = con.prepareStatement  ("update Customer72 set city=? where phno=?");  ParameterMetaData pmd = ps.getParameterMetaData();  System.out.println("Para Count:"+pmd.getParameterCount());  System.out.println("\*\*\*\*\*ResultSetMetaData\*\*\*\*\*\*\*");  PreparedStatement ps2 = con.prepareStatement  ("select phno,city,mid from Customer72");  ResultSet rs = ps2.executeQuery();  while(rs.next()) {  System.out.println(rs.getLong(1)+"\t"  +rs.getString(2)+"\t"  +rs.getString(3));  }//end of loop  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:  \*\*\*\*\*\*DatabaseMetaData\*\*\*\*\*  Driver Name:Oracle JDBC driver  Driver Version:11  \*\*\*\*\*ParameterMetaData\*\*\*\*\*\*  Para Count:2  \*\*\*\*\*ResultSetMetaData\*\*\*\*\*\*\*  8686861234 Hyd rm@gmail.com  4545451234 Hyd r@gmail.com  Col-Count:3  2nd Col-Name:CITY |

**Summary of CoreJava Objects:**

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 JDBC Objects:**

1.Connection Object

2.Statement Object

3.PreparedStatement Object

4.CallableStatement Object

5.ResultSet Object

(i)Scrollable ResultSet Object

(ii)NonScrollable ResultSet Object

6.DatabaseMetaData Object

7.ParameterMetaData Object

8.ResultMetaData Object

9.RowSet Object

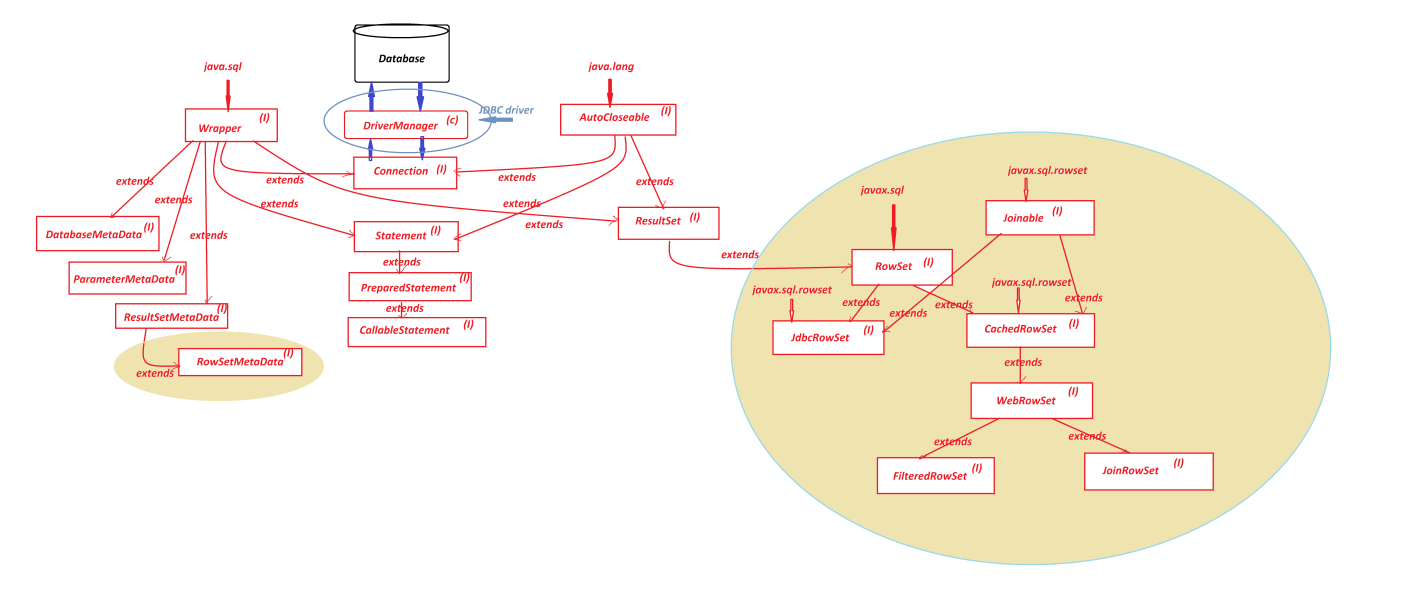
(a)JdbcRowSet Object

(b)CachedRowSet Object

10.RowSetMetaData Object

------

**Hierarchy of JDBC API**

******

Date: 21/03/2025 (Day-20)

***FAQ***

Define **'Wrapper' in JDBC?**

* 'Wrapper' is an interface from java.sql package and which specify the JDBC-Components are connected to database Product or not.
* 'Wrapper' extended to 'Connection' interface and having the following methods:

(i) isWrapperFor()

(ii) unwrap()

***FAQ***

Define **'AutoCloseable'?**

* 'AutoCloseable' is an interface from java.lang package and which supports auto-closing Operation.
* The Components which are used in try-with-resource statement must be extended or implemented from 'java.lang.AutoCloseable' interface.
* The JDBC-Components like Connection, Statement, PreparedStatement, CallableStatement and ResultSet will support auto-closing operation.

***Program : DBCon17.java***

|  |
| --- |
| package test;  import java.util.\*;  public class DBCon17 extends Exception  {  public DBCon17(String msg)  {  super(msg);  }  public static void main(String[] args)  {  Scanner s = new Scanner(System.in);  try(s;)//Java9  {  try  {  System.out.println("Enter the Marks of CoreJava:");  int cj = s.nextInt();//Exception for NonInteger value  if(cj<0 || cj>100)//Raise Exception  {  DBCon17 ob = new DBCon17("Invalid Marks....");  throw ob;  }  System.out.println("------Details-----");  System.out.println("Marks of CoreJava:"+cj);  }//end of try  catch(InputMismatchException ime)  {  System.out.println("Enter only Integer value....");  }  catch(DBCon17 d)  {  System.out.println(d.getMessage());  }  }//end of try with resource  }  } |
| ***Diagram*** |

Date: 22/03/2025 (Day-21)

***\*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)***

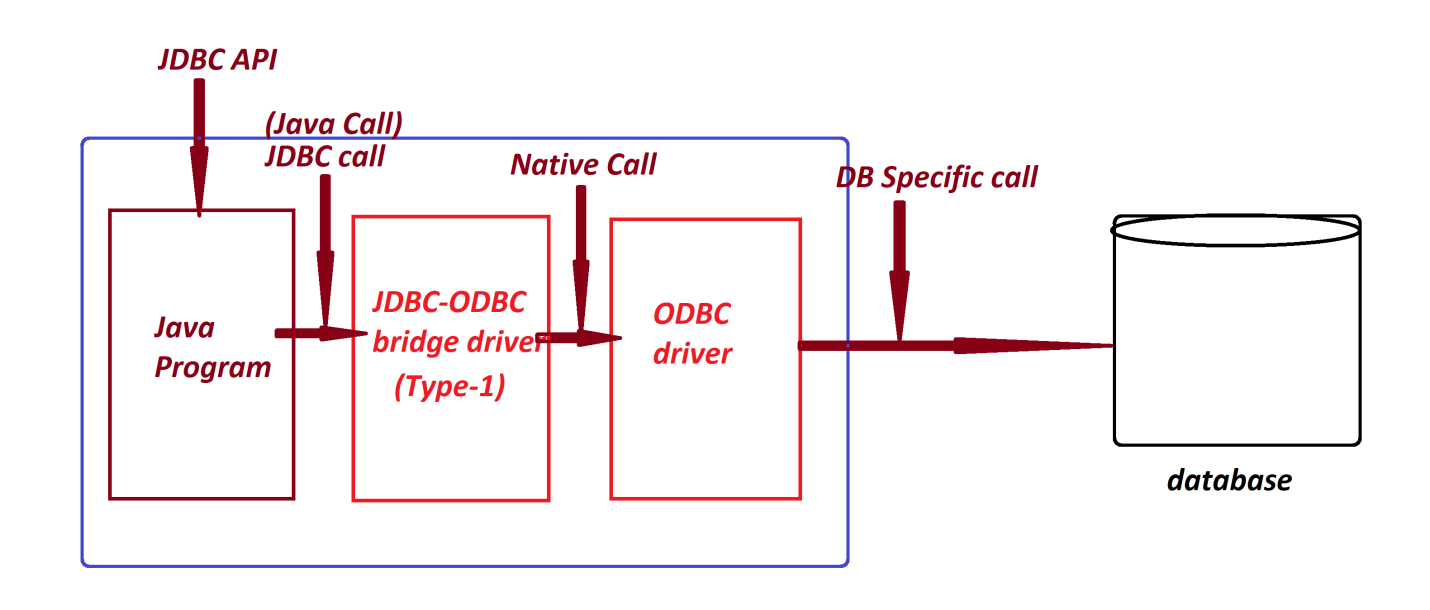
***3.Network protocol driver (Type-3 driver)***

***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?**

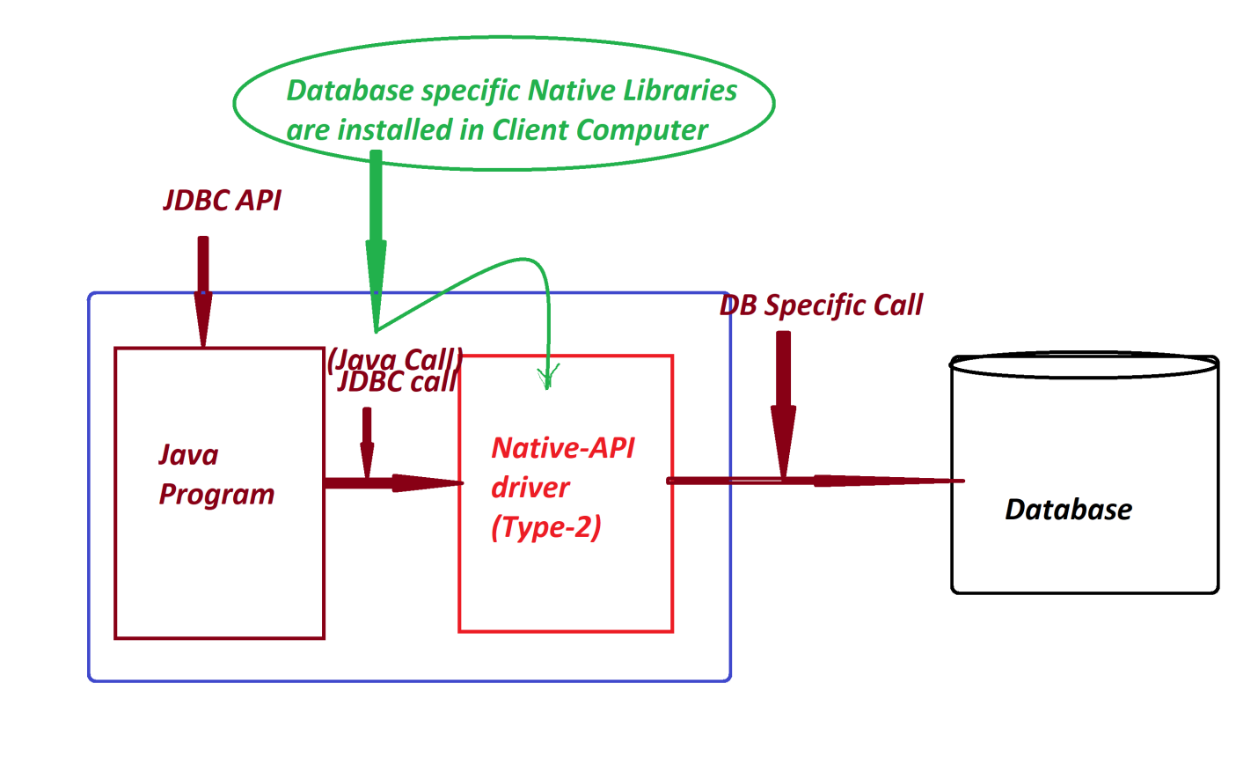
* ODBC stands for 'Open DataBase Connectivity', and this driver will support to establish connection to any type of database.
* This ODBC driver is PlatForm dependent driver, because which internally uses C/C++ codes.

---

**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***

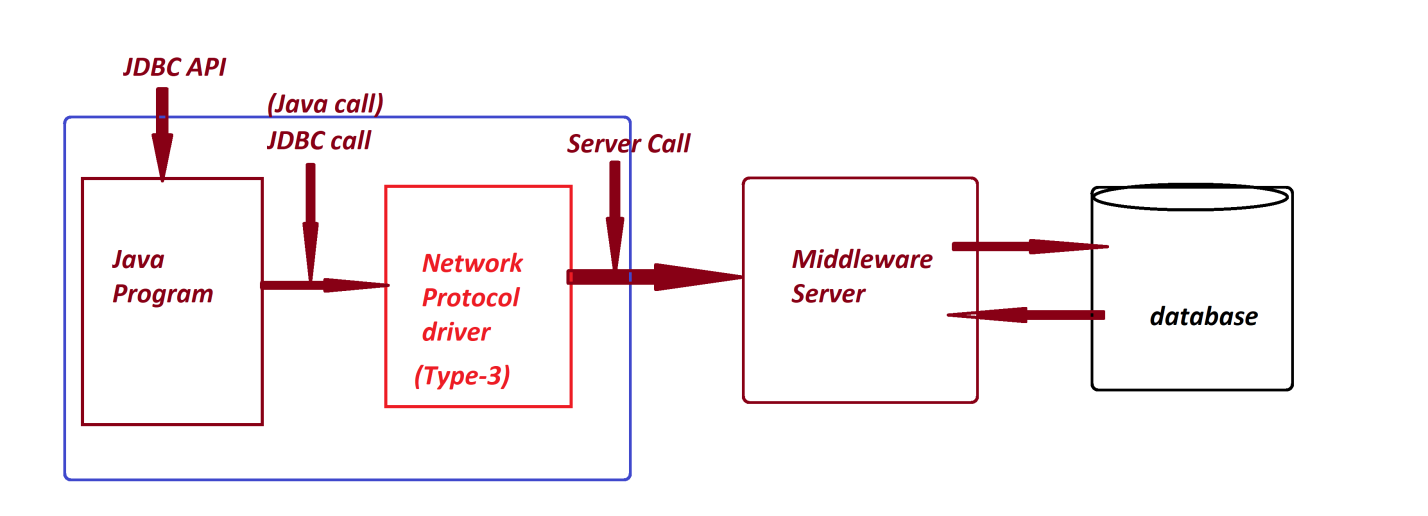
* When we construct application with Type-2 driver, then the application will become Database dependent and which is not preferable in realtime.

---

**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***

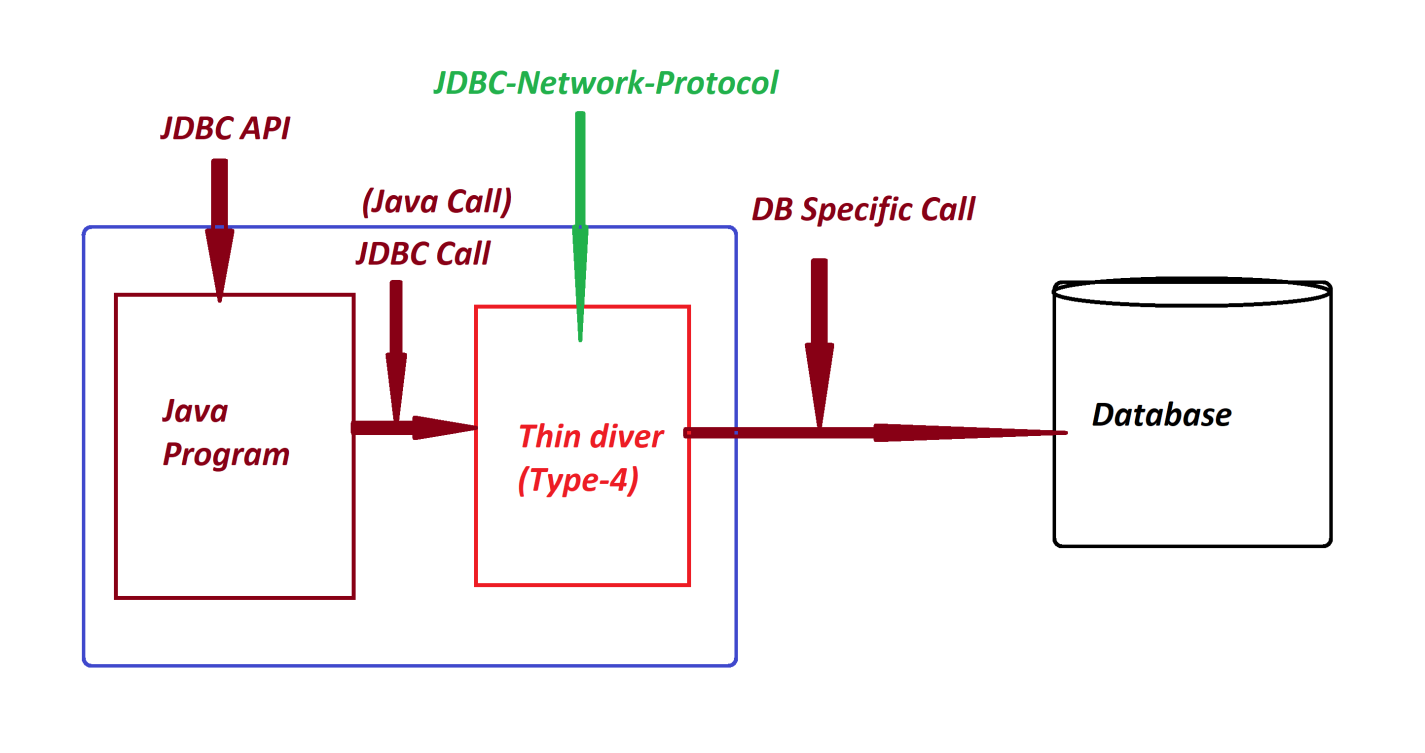
* When we want to use Type-3 driver, we have to make Network settings in Client Computer and the Network components are involved in execution process and degrades the performance of an application. (Execution time increases)

***\*imp***

**4. Thin driver (Type-4 driver)**

* Type-4 driver will take the support of Database(JDBC)-Network-protocol to establish connection 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***



***FAQ***

Define **Serialization process?**

* The process of converting Object-state into stream is known as Serialization process.
* To perform Serialization process the class must be implemented from 'java.io.Serializable' interface.

***FAQ***

*What is the advantage of* ***Serialization process?***

* Through Serialization process we can make Objects available in the form of Stream and can be moved on the network from one location to another location.

***FAQ***

Define **DeSerialization process?**

* The process of converting Stream into Object-state is known as DeSerialization process.

***\*imp***

**Types of Objects**

* Based on Serialization process the Objects are categorized into two types:

*(a) Serializable Objects*

*(b) NonSerializable Objects*

**(a) Serializable Objects**

* The Objects which are generated from implementation classes of Serializable interface are known as Serializable Objects.

***Example***

All CoreJava Objects

**(b) NonSerializable Objects**

* The Objects which are generated from Non-Implementation classes of Serializable interface are known as NonSerializable Objects.

***Example***

All JDBC Objects

---

***\*imp***

**Java PlatForms**

* Java PlatForms are categorized into the following:

***1. JavaSE***

***2. JavaEE***

***3. JavaME***

***4. JavaFX***

**1. JavaSE**

* JavaSE stands for **'Java Standard Edition**' and which provide environment to construct NonServer Applications or Stand-Alone-Applications.

***Example*** -> *CoreJava + JDBC*

**2. JavaEE**

* JavaEE satnds for **'Java Enterprise Edition**' and which provide environment to construct Server based Applications.

***Example*** -> Servlet + JSP + WebServices

***FAQ***

*Define Server based Applications?*

* The applications which are executed in server environment are known as Server Applications or Server based Applications.
* These Server based applications are categorized into two types:

*(a) Web Applications*

*(b) Enterprise Applications*

(a) Web Applications:

* The applications which are constructed using Servlet-JSP and available in 3-tier architecture are known as Web Applications.

(b) 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.
* These Enterprise Applications are available in n-tier architecture.

**3. JavaME**

* JavaME stands for 'Java Micro Edition' and which provide environment to construct applications related to Machine and Mobile.
* JavaME can also be called as Java Machine Edition or Java Mobile Edition.

**4. JavaFX**

* JavaFX introduced by Java8 version and which provide environment to develop rich UI Application.

Date: 24/03/2025 (Day-22)

***\*imp***

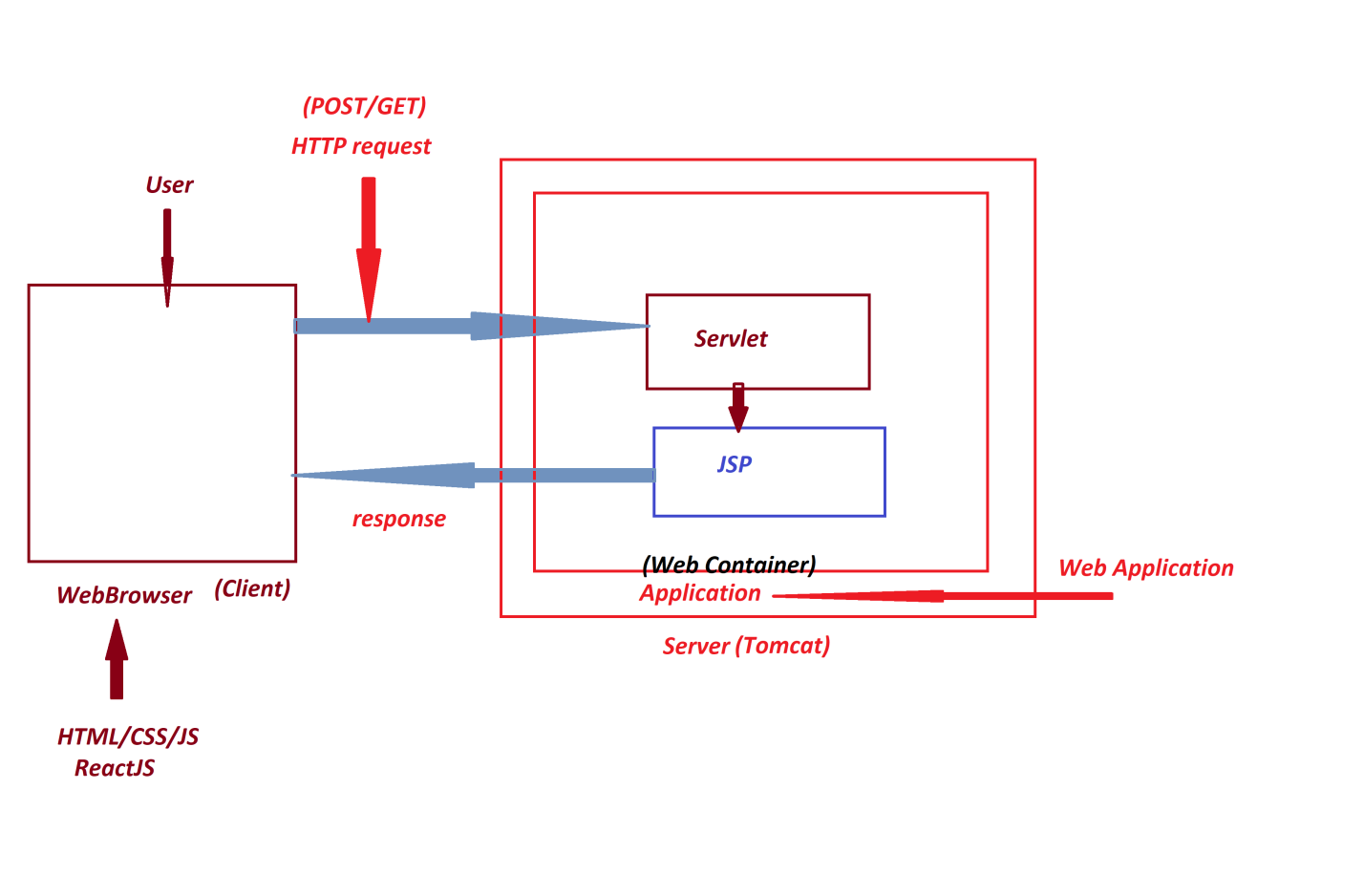
**Servlet Programming**

***FAQ***

define **Servlet?**

* The platform independent Java-Program, which is executed in server environment is known as Servlet or Server-program.
* Servlet-Program will specify the functionality of Servers.
* Servlet-Program will accept the request from the user through Web Browser and provide the response.

***Diagram***



***FAQ***

define **Client?**

* Client means Web Browser, from where the request is raised to the Server.

***FAQ***

define **Server?**

* Server means service provider, which means accepting request and giving response using Server-Program-Support.

(service - means accept request and provide response)

**Types of Servers**

* Servers are categorized into two types:

*(a) Web Servers*

*(b) Application Servers.*

**(a) Web Servers**

* Web Servers will provide 'Web Container' to execute Web Applications.
* Web Servers are preferable to execute static content data.
* Web Servers will accept request from HTTP protocol.

***Example*** -> Tomcat

**(b) Appplication Servers**

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

(EJB - Enterprise Java Bean)

* Application Servers are preferable to execute dynamic content data.
* Application will accept request from from HTTP, RMI and RPC protocols.

(RMI - Remote Method Invocation)

(RPC - Remote Procedure Call)

***Example*** -> WebSphere

WebLogic

Date: 25/03/2025 (Day-23)

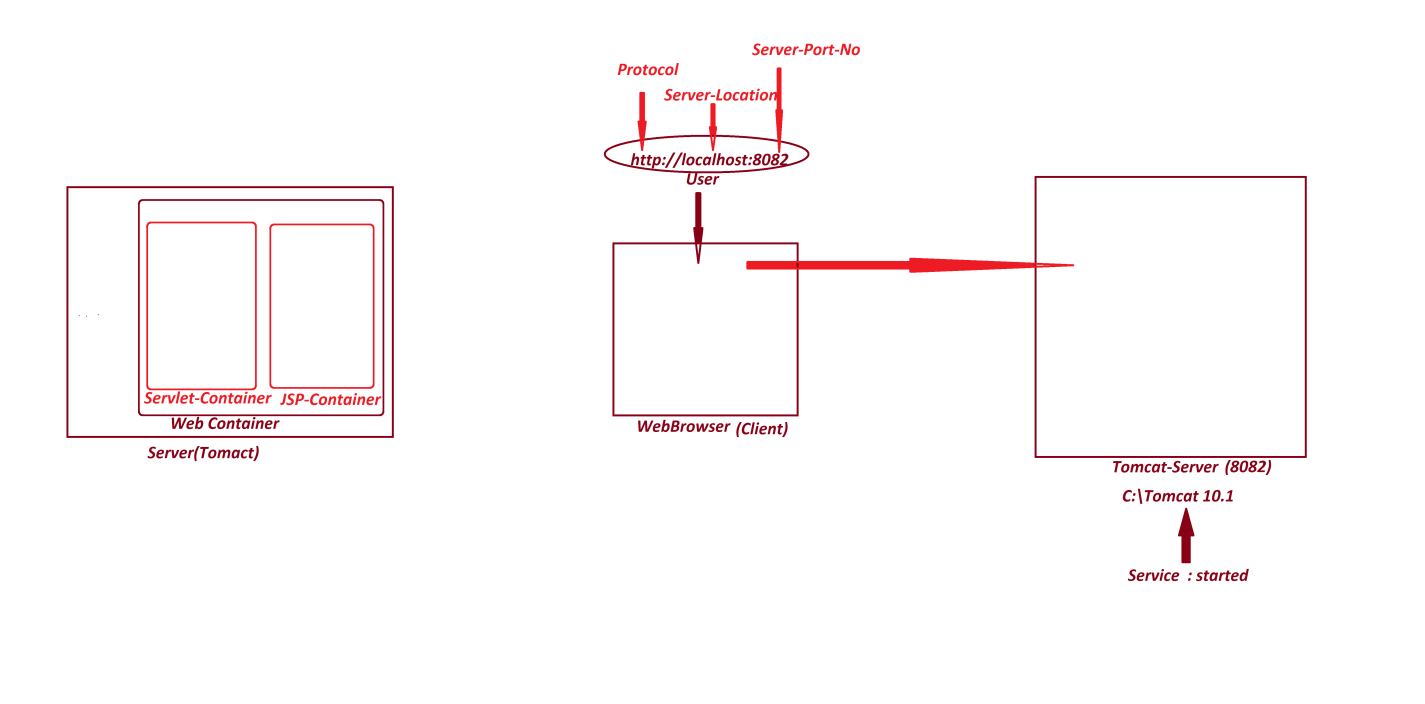
***FAQ***

define **'Web Container'?**

* The executable component of WebServers and application servers is known as Web Container.
* Web Container internally divided into the following SubContainers:

**(i) Servlet Container** <------- Catelina

**(ii) JSP Container** <------- Jasper



***\*imp***

**Installing Tomcat Server**

**step-1 :** Download Tomcat10-version from 'Apache Organiozation'

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

**step-2 :** Install Tomcat Server While installing process,

Select the type of Install : Full

(Click Next)

Server Shutdown port : 8089(User defined PORT NO)

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

User Name : system

Password : lalit

(Click Next)

Select Java Virtual Machine Path (JDK11 and above)

(Click Next)

***Note***

Tomcat Server available in:

C:\Tomcat 10.1

**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,using the following URL:

<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

***Diagram***

***\*imp***

**Servlet-API**

* 'jakarta.servlet' package is known as Servlet-API from Tomcat10-version onwards.

(*Upto Tomcat9-version 'javax.servlet' is Servlet-API)*

* 'jakarta.servlet.Servlet' is an interface, root of Servlet-API.
* The following are some important methods of 'Servlet' interface:

***1. init()***

***2. service()***

***3. destroy()***

***4. getServletInfo()***

***5. getServletConfig()***

Date: 26/03/2025 (Day-24)

**1. init()**

* init() - method is used to perform initialization process, which means making the required programming components ready for service() -method.
* *Method Signature of init():*

public abstract void init(jakarta.servlet.ServletConfig)

throws jakarta.servlet.ServletException;

---

**2. service()**

* service()-method will perform request handling process, which means accepting the request and providing the response.
* *Method Signature of service()*

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 opened part of initialization and request handling process.
* *Method Signature destroy()*

public abstract void destroy();

---

**4. getServletInfo()**

* getServletInfo()-method is used to display the servlet-Information.
* *Method Signature of getServletInfo()*

public abstract java.lang.String getServletInfo();

---

**5. getServletConfig()**

* getServletConfig()-method is used to display the Servlet Configuration details.
* *Method Signature of getServletConfig()*

public abstract jakarta.servlet.ServletConfig getServletConfig();

***Note***

* The following three methods are executed automatically in same order and which are known as **Servlet-Life-Cycle** methods:

***(i) init()***

***(ii) service()***

***(iii) destroy()***

* 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 of 'Servlet-Interface'.

***Diagram***

---

***\*imp***

***Creating and executing Web Application using IDE Eclipse:***

**Step-1 :** Open IDE Eclispe,while Opening select 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' file to Project through 'Build path'

RightClick on 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

Click on Servers->Click on 'Click this link to Create new Server'->

Select the Server with version and click 'Next'->

Browse and select 'Tomcat Installation directory'->Click 'Finish'

**Step-5 :** Construct HTML file to read data to Servlet Program

RightClick on webapp->New->HTML file->name the file and 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"><br>  <input type="submit" value="Display">  </form>  </body>  </html> |

**Step-6 :** Construct 'web.xml' mapping file

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 Servlet-Program in package

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 String getServletInfo()  {  return "Servlet displaying User details....";  }  @Override  public ServletConfig getServletConfig()  {  return this.getServletConfig();  }  } |

**Step-9 :** Execute the Application(Project)

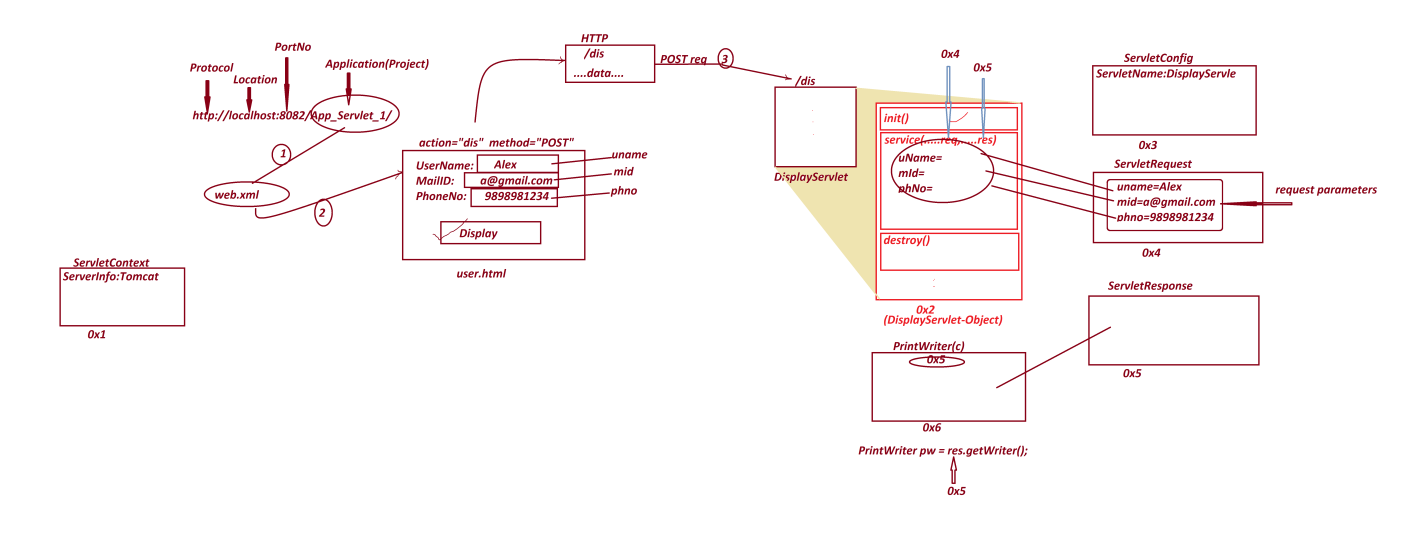
RightClick on Project->Run AS->Run On Server->Select the server and Click 'Finish'

http://localhost:8082/App\_Servlet\_1/

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

Date: 27/03/2025 (Day-25)

***Execution Flow of above application***



**ServletContext**

* 'ServletContext' is an interface from 'jakarta.servlet' package and which is instantiated automatically when WebApplication deployed into server.
* This ServletContext-Object is recorded with Server information.

***Note***

* The Servlet-Program identified based on URL-Pattern and loaded for execution, and which is instantiated automatically.
* After Servlet-Program instantiation, the execution controls will find the following Servlet-Life-Cycle methods and executed automatically:

**(i) init()**

**(ii) service()**

**(iii) destroy()**

**ServletConfig**

* 'ServletConfig' is an interface from 'jakarta.servlet' package and which is instantiated automatically before Life-Cycle-methods execution.
* This ServletConfig-Object is loaded with Servlet\_name.

**ServletRequest**

* 'ServletRequest' is an interface from 'jakarta.servlet' package and which is instantiated automatically while service() -method execution.
* This ServletRequest- Object is loaded with HTML form data.

**ServletResponse**

* 'ServletResponse' is an interface from 'jakarta.servlet' package and which is instantiated while service()-method execution.
* ServletResponse- Object is added with data which we are sending as response.

***FAQ***

define **getParameter()** method?

* getParameter()- method is from 'ServletRequest' and which is used to read parameter-value from request-Object to Servlet-Program.
* Method Signature of **getParameter()**

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

***Syntax***

**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 PrinterWriter-Object will hold the response of ServletResponse-Object
* Method Signature of getWriter()

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

***Syntax***

**PrintWriter pw = res.getWriter();**

***FAQ***

define **setContentType()** method?

* setContentType()- method is from ServletResponse and which specify the type of data added to the response-object.
* Method Signature of setContentType()

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

***Syntax***

**res.setContentType("text/html");**

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

***Assignment***

*Construct Servlet Applications to read and display Book details.*

*BookCode*

*BookName*

*BookAuthor*

*BookPrice*

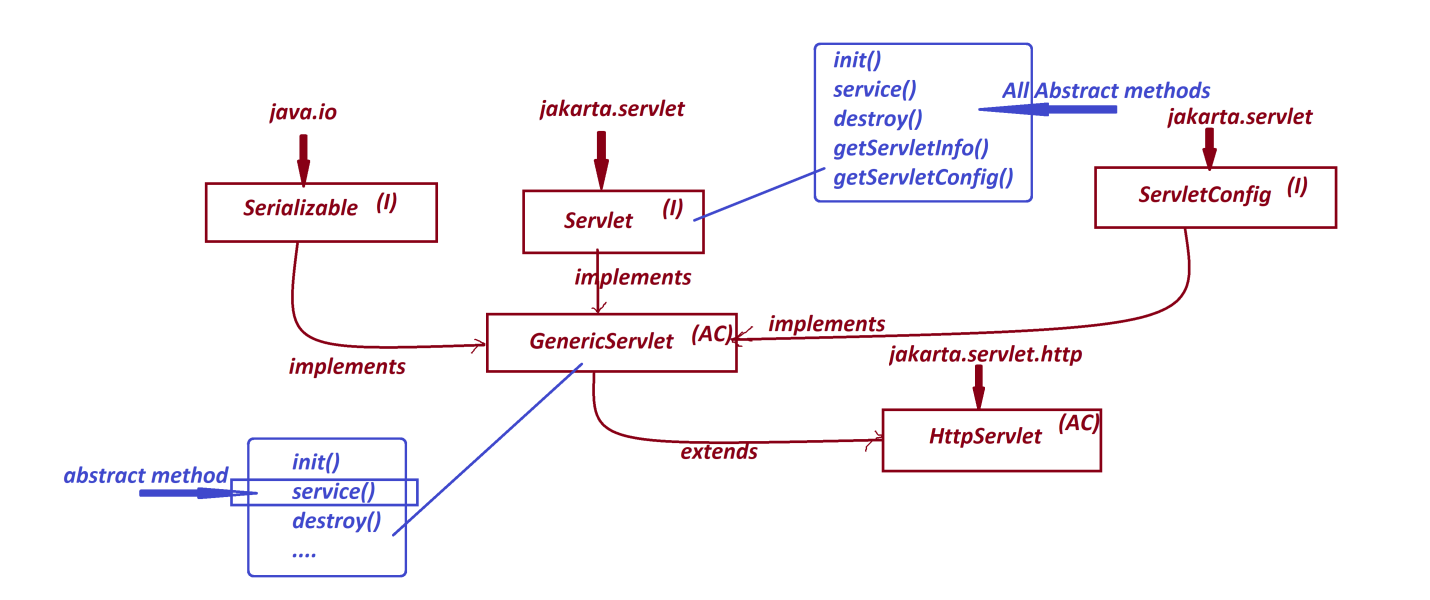
*BookQty*

**[**[**View Program**](https://github.com/lalitpatil891/Code-with-AdvanceJava/tree/main/servlet_programs/App_servlet_book_details/src/main)**]**

śDate: 28/03/2025 (Day-26)

***\*imp***

**Hierarchy of Servlet-API**



* In the process of Constructing Servlet-Program, we can use any one of the following design model.

***Model-1 : Using 'Servlet' interface***

***Model-2 : Using 'GenericServlet' AbstractClass***

***Model-3 : Using 'HttpServlet' AbstractClass***

*---*

***Model-1: Using 'Servlet' interface***

* In Model-1, the Servlet-Program must be implemented from 'Servlet' interface and which must implement all abstract methods of Servlet-Interface.

***Model-2: Using 'GenericServlet' AbstractClass***

* In Model-2, the Servlet-Program must be extended from 'GenericServlet' AbstarctClass, and in this process only service()-method is abstract method

***Model-3: Using 'HttpServlet' AbstractClass***

* In Model-3, the Servlet-Program must be extended from 'HttpServlet' AbstractClass, and in this process all methods are NonAbstract methods

*(i) init()*

*(ii) service()/doPost()/doGet()*

*(iii) destroy()*

***Note***

***service()*** *- method will accept both POST and GET requests.*

***doPost()*** *- method will accept only POST request*

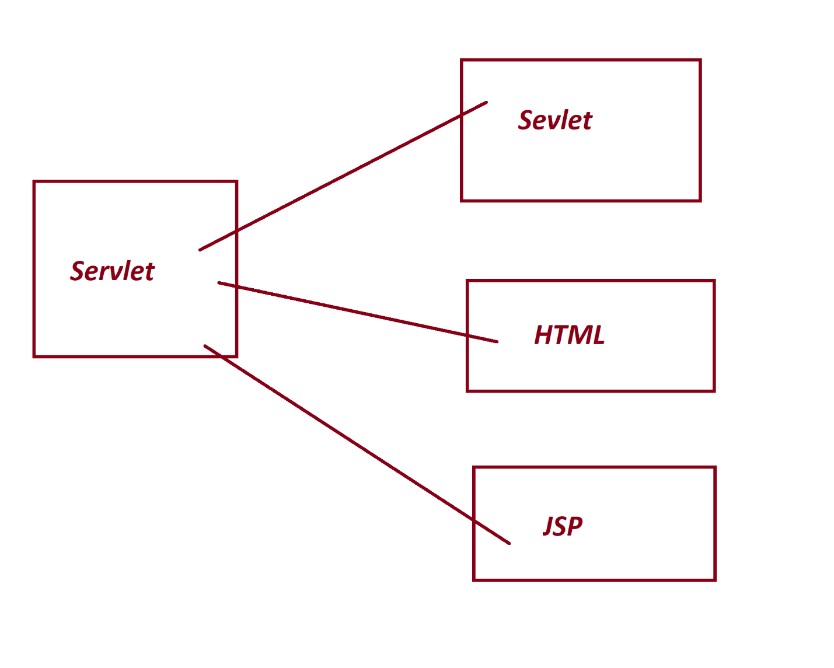
***doGet()*** *- method will accept only GET request*

------

***\*imp***

**'RequestDispatcher' in Servlet Programming**

* 'RequestDispatcher' is an interface from jakarta.servlet package and which is used to perform Servlet-Communications like Servlet-Servlet Communication, Servlet-HTML Communication and Servlet-JSP Communication.



* These Servlet Communications are categorized into two types:

(a) forward Communication process

(b) include Communication process

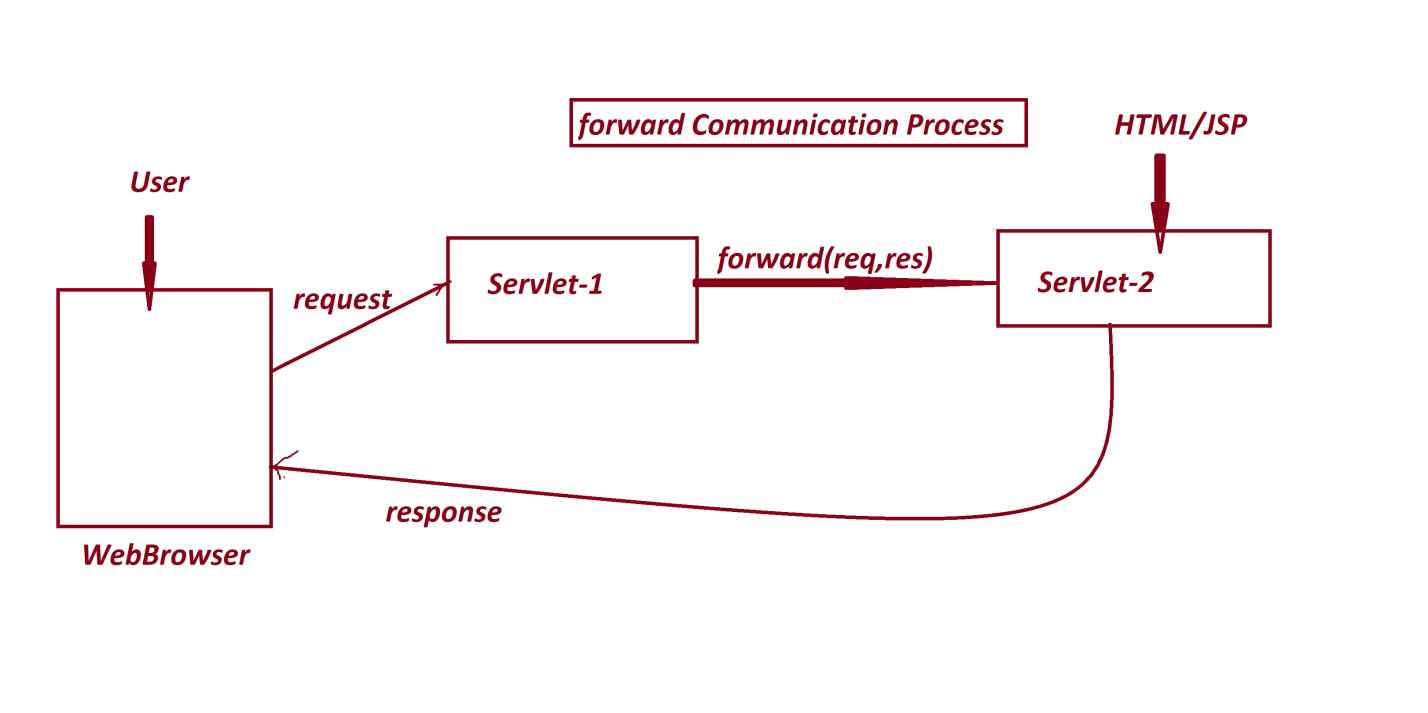
**(a)forward Communication process**

* In forward Communication Process, Servlet-1 will take the request and forwards the request to Servlet-2, In this process Servlet-2 will give the response.
* Servlet-2 can be replaced with HTML file/JSP file.
* We use forward()-method from '*RequestDispatcher*' to perform forward Communication process.
* *Method Signature of* ***forward()***

public abstract void forward(jakarta.servlet.ServletRequest,

jakarta.servlet.ServletResponse) throws jakarta.servlet.ServletException, java.io.IOException;

***Diagram***



**(b) include Communication process**

* In include Communication process the Servlet-1 will take the request and generate the response, in this process the response is included with the response on Servlet-2.
* Servlet-2 can be replaced with HTML file/JSP file.
* *Method Signature of* ***include()***

public abstract void include(jakarta.servlet.ServletRequest,

jakarta.servlet.ServletResponse) throws jakarta.servlet.ServletException, java.io.IOException;

***Diagram***

------

***Note***

* we use getRequestDispatcher()-method to create implementation Object for 'RequestDispatcher' interface.
* This requestDispatcher()-method is available from 'ServletRequest'
* Method Signature of **getRequestDispatcher()**

**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);

***-------***

Date: 29/03/2025 (Day-27)

***\*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 Java Code, which means in JSP Programs we can write HTML code and Java Code.
* JSP programs are saved with **(.jsp)** as an extention.

***Example*** -> View.jsp

* we use the following JSP tags to write Java Code in JSP programs:

***1. Scripting Tags***

***2. Directive Tags***

***3. Action Tags***

*===*

1. **Scripting Tags**

* The tags which are used to write normal code are known as Scripting Tags.
* Types

***(a) Scriptlet Tag***

***(b) Expression Tag***

***(c) Declarative Tag***

---

**(a) Scriptlet Tag**

* Scriptlet Tag is used to write normal Java Code in JSP programs.
* ***syntax***

**<%**

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

**%>**

---

**(b) Expression Tag**

* Expression Tag is used to assign the value to variable or which is used to display the data directly to WebBrowser.
* ***syntax***

**<%=**

**-----Value/Expression----**

**%>**

---

**(c) Declarative Tag**

* Declarative tag is used to declare Variables and methods in JSP programs.
* ***syntax***

**<%!**

**---Variables/Methods-----**

**%>**

---

===

**2. Directive Tags**

* Directive tags will specify the directions(specifications) in translation process.
* ***Types***

***(a) @page***

***(b) @include***

***(c) @taglib***

---

**(a) @page**

* '@page' tag will provide specifications about current JSP page.
* ***syntax***

**<%@page**

**language="java"**

**setContentType="text/html"**

**import="java.util.\*"**

**...%>**

**(b)** **@include**

* '@include' tag will specify the file which is included with the response.

***syntax***

**<%@include file="file-name" %>**

**(c) @taglib**

* '@taglib' tag is used to link the external libraries.

**Syntax**

**<%@taglib uri="lib...." prefix="nm"...%>**

---

**3. Action Tags**

* Actions tags are used to peform some actions while execution process.
* ***Types***

(a) <jsp:forward>

(b) <jsp:include>

(c) <jsp:param>

(d) <jsp:useBean>

(e) <jsp:setProperties>

(f) <jsp:getProperties>

***\*imp***

**Bean Class in Servlet Programming**

* The class which is declared 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 declared in 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 and which are intermediate storages b/w Servlet-prorams and Database product

Date: 01/04/2025 (Day-28)

**\*imp**

**DAO in JDBC**

* DAO stands for 'Data Access Object' and which is separate layer in MVC (Model View Controller) to hold database related codes (Persistent Logics).
* This DAO Layer includes DB-Connection, DB-Create, DB-Insert, DB-Retrieve, DB-Update and DB-Delete.
* In the process of establishing Communication b/w Servlet-Program and DB-Product, the DB-Jar file must be copied into "lib" folder of 'WEB-INF'.

***Layout***

***Ex-Application***

*Servlet Application to perform the following two operations on Database Product*

*1. AddBookDetails*

*2. ViewBookDetailsByCode*

DB Table: BookDetails72(code,name,author,price,qty)

primary key : code

create table BookDetails72(

code varchar2(10),

name varchar2(15),

author varchar2(15),

price number(10,2),

qty number(10),

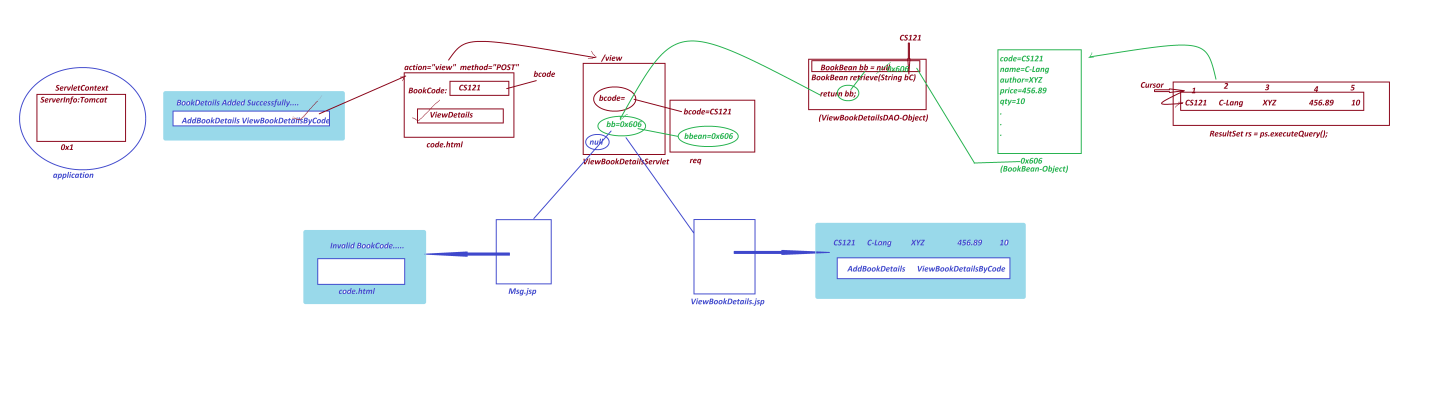
primary key(code));

|  |
| --- |
| **-------------**  **BookBean.java**  **-------------**  package test;  import java.io.\*;  @SuppressWarnings("serial")  public class BookBean implements Serializable  {  private String code,name,author;  private float price;  private int qty;  public BookBean() {}  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 String getAuthor() {  return author;  }  public void setAuthor(String author) {  this.author = author;  }  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;  }  }  **-----------**  **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;  }  } |
| **-------------**  **home.html**  **-------------**  <!DOCTYPE html>  <html>  <head>  <meta charset="ISO-8859-1">  <title>Insert title here</title>  </head>  <body>  <a href="BookDetails.html">AddBookDetails</a>  <a href="view">ViewBookDetailsByCode</a>  </body>  </html> |
| **-------------------**  **BookDetails.html**  **-------------------**  <!DOCTYPE html>  <html>  <head>  <meta charset="ISO-8859-1">  <title>Insert title here</title>  </head>  <body>  <form action="add" method="post">  BookCode:<input type="text" name="bcode"><br>  BookName:<input type="text" name="bname"><br>  BookAuthor:<input type="text" name="bauthor"><br>  BookPrice:<input type="text" name="bprice"><br>  BookQty:<input type="text" name="bqty"><br>  <input type="submit" value="AddBookDetails">  </form>  </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 |

Date: 02/04/2025 (Day-29)

|  |
| --- |
| **----------------------**  **AddBookDetailsDAO.java**  **----------------------**  package test;  import java.sql.\*;  public class AddBookDetailsDAO  {  public int k = 0;  public int insert(BookBean bb)  {  try {  Connection con = DBConnection.getCon(); //Accessing Connection Object  PreparedStatement ps = con.prepareStatement  ("insert into BookDetails72 values(?,?,?,?,?)");  ps.setString(1, bb.getCode());  ps.setString(2, bb.getName());  ps.setString(3, bb.getAuthor());  ps.setFloat(4, bb.getPrice());  ps.setInt(5, bb.getQty());  k = ps.executeUpdate();  }catch(Exception e) {  e.printStackTrace();  }  return k;  }  }  **--------------------------------**  **AddBookDetailsServlet.java**  **--------------------------------**  package test;  import java.io.\*;  import jakarta.servlet.\*;  import jakarta.servlet.annotation.\*;  @WebServlet("/add")  public class AddBookDetailsServlet extends GenericServlet  {  @Override  public void service(ServletRequest req,ServletResponse res)throws  ServletException,IOException  {  BookBean bb = new BookBean();//Bean Object  bb.setCode(req.getParameter("bcode"));  bb.setName(req.getParameter("bname"));  bb.setAuthor(req.getParameter("bauthor"));  bb.setPrice(Float.parseFloat(req.getParameter("bprice")));  bb.setQty(Integer.parseInt(req.getParameter("bqty")));  int k = new AddBookDetailsDAO().insert(bb);  if(k>0) {  req.setAttribute("msg", "BookDetails Added Successfully...<br>");  RequestDispatcher rd =  req.getRequestDispatcher("AddBookDetails.jsp");  rd.forward(req, res);  }  }  }  **------------------**  **AddBookDetails.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);  %>  <a href="BookDetails.html">AddBookDetails</a>  <a href="code.html">ViewBookDetailsByCode</a>  </body>  </html> |

Date: 03/04/2025 (Day-30)

******

**------------**

**code.html**

**------------**

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

<form action="view" method="post">

Enter the BookCode:<input type="text" name="bcode"><br>

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

</form>

</body>

</html>

**------------------------------**

**ViewBookDetailsDAO.java**

**------------------------------**

package test;

import java.sql.\*;

public class ViewBookDetailsDAO

{

public BookBean bb = null;

public BookBean retrieve(String bC)

{

try {

Connection con = DBConnection.getCon();//Accessing Connection Object

PreparedStatement ps = con.prepareStatement

("select \* from BookDetails72 where code=?");

ps.setString(1, bC);

ResultSet rs = ps.executeQuery();

if(rs.next()) {

bb = new BookBean();

bb.setCode(rs.getString(1));

bb.setName(rs.getString(2));

bb.setAuthor(rs.getString(3));

bb.setPrice(rs.getFloat(4));

bb.setQty(rs.getInt(5));

}

}catch(Exception e) {

e.printStackTrace();

}

return bb;

}

}

**----------------------------------**

**ViewBookDetailsServlet.java**

**----------------------------------**

package test;

import java.io.\*;

import jakarta.servlet.\*;

import jakarta.servlet.annotation.\*;

@SuppressWarnings("serial")

@WebServlet("/view")

public class ViewBookDetailsServlet extends GenericServlet

{

@Override

public void service(ServletRequest req,ServletResponse res)throws

ServletException,IOException

{

String bcode = req.getParameter("bcode");

BookBean bb = new ViewBookDetailsDAO().retrieve(bcode);

if(bb==null) {

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

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

}else {

req.setAttribute("bbean", bb);

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

}

}

}

**------------------------**

**ViewBookDetails.jsp**

**------------------------**

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

pageEncoding="ISO-8859-1" import="test.BookBean"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

<%

BookBean bb = (BookBean)request.getAttribute("bbean");

out.println(bb.getCode()+"&nbsp&nbsp"

+bb.getName()+"&nbsp&nbsp"

+bb.getAuthor()+"&nbsp&nbsp"

+bb.getPrice()+"&nbsp&nbsp"

+bb.getQty()+"<br>");

%>

<a href="BookDetails.html">AddBookDetails</a>

<a href="code.html">ViewBookDetailsByCode</a>

</body>

</html>

**---------**

**Msg.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="code.html" %>

</body>

</html>

-------

**Assignment-1**

*Update above application with the following option*

*=>ViewAllBookDetails*

**Assignment-2**

*ProjectName : CustomerApp*

*DBTable : Customer72(id,name,city,mid,phno)*

*primary key : id*

*Generate Id : NIT+PhNo*

*Operations:*

*1.AddCustomer*

*2.ViewCustomerById*

*3.ViewAllCustomers*

Date: 04/04/2025 (Day-31)

***\*imp***

**'ServletContext' in Servlet Programming**

* 'ServletContext' is an interface from jakarta.servlet package and which is instantiated automatically when Web-Application deployed into Server.
* This ServletContext-Object is loaded with Server information.
* we use getServletContext()-method to access ServletContext Object reference into Servlet Program.
* ***Syntax***

**ServletContext sct = req.getServletContext();**

* We use 'application' an implicit object name to access ServletContext into JSP-Program.
* we use <context-param> tag in web.xml to intialize parameter-values in ServletContext Object.
* ***syntax***

<web-app>

<context-param>

<param-name> name </param-name>

<param-value> value </param-value>

</context-param>

...

</web-app>

* we use **getInitParameter()**-method to access the parameter-value from ServletContext Object.
* ***syntax***

**ServletContext sct = req.getServletContext();**

**String var = sct.getInitParameter("x");**

**String var = application.getInitParameter("x");**

---

***Example***

input.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>

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

</form>

</body>

</html>

web.xml

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

<web-app>

<context-param>

<param-name>x</param-name>

<param-value>1000</param-value>

</context-param>

<welcome-file-list>

<welcome-file>input.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

{

public void service(ServletRequest req,ServletResponse res)throws

ServletException,IOException

{

req.getRequestDispatcher("Display.jsp").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 uName = request.getParameter("uname");

out.println("Welcome User : "+uName+"<br>");

out.println("\*\*\*\*\*\*\*ServletContext\*\*\*\*\*\*<br>");

out.println("ServerInfo:"+application.getServerInfo()+"<br>");

out.println("Context-Value:"+application.getInitParameter("x")+"<br>");

%>

</body>

</html>

Date: 05/04/2025 (Day-32)

***\*imp***

**'ServletConfig' in Servlet Programming**

* 'ServletConfig' is an interface from jakarta.servlet package and which is instantiatedautomatically when Servlet-Program loaded for execution.
* This ServletConfig-Object is loaded with Servlet-name.
* we use getServletConfig()-method to access ServletConfig Object reference into Servlet program.
* ***Syntax***

ServletConfic scf = super.getServletConfig();

* we use <init-param> subtag of <servlet> tag in web.xml to initialize para-values to ServletConfig Object.

***syntax***

<web-app>

...

<servlet>

<servlet-name> name </servlet-name>

<servlet-class> class </servlet-class>

<init-param>

<param-name> name </param-name>

<param-value> value </param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name> name </servlet-name>

<url-pattern> /url </url-pattern>

</servlet-mapping>

...

</web-app>

* we use getInitParameter()-method to access initialized para-values from ServletConfig
* ***syntax***

String var = scf.getInitParameter("para-name");

***Example***

**-------------**

**input.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>

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

</form>

</body>

</html>

**----------**

**web.xml**

**----------**

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

<web-app>

<servlet>

<servlet-name>DisplayServlet</servlet-name>

<servlet-class>test.DisplayServlet</servlet-class>

<init-param>

<param-name>p</param-name>

<param-value>3000</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>DisplayServlet</servlet-name>

<url-pattern>/dis</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>input.html</welcome-file>

</welcome-file-list>

</web-app>

**-----------------------**

**DisplayServlet.java**

**-----------------------**

package test;

import java.io.\*;

import jakarta.servlet.\*;

@SuppressWarnings("serial")

public class DisplayServlet extends GenericServlet {

public void service ( ServletRequest req, ServletResponse res)throws

ServletException, IOException {

ServletConfig scf = super.getServletConfig();

req.setAttribute("scf", scf);

req.getRequestDispatcher("Display.jsp").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 uName = request.getParameter("uname");

out.println("Welcome User : "+uName+"<br>");

out.println("\*\*\*\*\*\*\*ServletConfig\*\*\*\*\*\*\*<br>");

ServletConfig scf = (ServletConfig)request.getAttribute("scf");

out.println("Servlet-Name:"+scf.getServletName()+"<br>");

out.println("Config-Value:"+scf.getInitParameter("p")+"<br>");

out.println("\*\*\*\*\*\*JSP-config\*\*\*\*\*\*\*\*\*<br>");

out.println("Servlet-Name:"+config.getServletName()+"<br>");

%>

</body>

</html>

***Output***

Welcome User: Alex

\*\*\*\*\*\* ServletConfig \*\*\*\*\*\*\*

Servlet-Name: DisplayServlet

Config-Value:3000

\*\*\*\*\*\* JSP-config \*\*\*\*\*\*\*\*\*

Servlet-Name:jsp

----

What is the diff b/w

1. *getParameter()*
2. *getParameterNames()*
3. *getInitParameter()*
4. *getInitParameterNames()*

**(i) getParameter()**

* getParameter()-method is used to get single parameter value from request object.
* *Method Signature*

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

* ***syntax***

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

**(ii) getParameterNames()**

* getParameterNames() -method is used get all parameter names from request Object.
* Method Signature

public abstract java.util.Enumeration<java.lang.String>

getParameterNames();

* ***syntax***

Enumeratiom<String> e = req.getParameterNames();

**(iii) getInitParameter()**

* getInitParameter() -method is used get single initialized para-value from ServletContext and ServletConfig Objects.
* Method Signature

**public abstract java.lang.String getInitParameter(java.lang.String);**

* **syntax**

**String p = sct.getInitParameter("p");**

**(iv) getInitParameterNames()**

* getInitParameterNames()-method is used to get all initialized parameter names from Context and Config Objects
* Method Signature

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

* ***Syntax***

**Enumeration<String> e = sct.getInitParameterNames();**

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

***Assignment***

*Construct Application to demonstrate the following methods:*

*::getParameterNames() - use this method to collect multiple parameter names from request*

*::getInitParameterNames() - use this method to collect multiple initialization para-names from Context Object*

Date: 08/04/2025 (Day-33)

\*imp

'ServletConfig' in Servlet Programming:

=>'ServletConfig' is an interface from jakarta.servlet package and which is instantiated

automatically when Servlet-Program loaded for execution.

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

=>we use getServletConfig()-method to access ServletConfig Object reference into Servlet

program.

syntax:

ServletConfic scf = super.getServletConfig();

=>we use <init-param> subtag of <servlet> tag in web.xml to initialize para-values to

ServletConfig Object.

syntax:

<wb-app>

...

<servlet>

<servlet-name> name </servlet-name>

<servlet-class> class </servlet-class>

<init-param>

<param-name> name </param-name>

<param-value> value </param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name> name </servlet-name>

<url-pattern> /url </url-pattern>

</servlet-mapping>

...

</web-app>

=>we use getInitParameter()-method to access initialized para-values from ServletConfif

Object

syntax:

String var = scf.getInitParameter("para-name");

Ex:

input.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>

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

</form>

</body>

</html>

web.xml

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

<web-app>

<servlet>

<servlet-name>DisplayServlet</servlet-name>

<servlet-class>test.DisplayServlet</servlet-class>

<init-param>

<param-name>p</param-name>

<param-value>3000</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>DisplayServlet</servlet-name>

<url-pattern>/dis</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>input.html</welcome-file>

</welcome-file-list>

</web-app>

DisplayServlet.java

package test;

import java.io.\*;

import jakarta.servlet.\*;

@SuppressWarnings("serial")

public class DisplayServlet extends GenericServlet{

public void service(ServletRequest req,ServletResponse res)throws

ServletException,IOException{

ServletConfig scf = super.getServletConfig();

req.setAttribute("scf", scf);

req.getRequestDispatcher("Display.jsp").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 uName = request.getParameter("uname");

out.println("Welcome User : "+uName+"<br>");

out.println("\*\*\*\*\*\*\*ServletConfig\*\*\*\*\*\*\*<br>");

ServletConfig scf = (ServletConfig)request.getAttribute("scf");

out.println("Servlet-Name:"+scf.getServletName()+"<br>");

out.println("Config-Value:"+scf.getInitParameter("p")+"<br>");

out.println("\*\*\*\*\*\*JSP-config\*\*\*\*\*\*\*\*\*<br>");

out.println("Servlet-Name:"+config.getServletName()+"<br>");

%>

</body>

</html>

o/p:

Welcome User : Alex

\*\*\*\*\*\*\*ServletConfig\*\*\*\*\*\*\*

Servlet-Name:DisplayServlet

Config-Value:3000

\*\*\*\*\*\*JSP-config\*\*\*\*\*\*\*\*\*

Servlet-Name:jsp

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

faq:

wt is the diff b/w

(i)getParameter()

(ii)getParameterNames()

(iii)getInitParameter()

(iv)getInitParameterNames()

(i)getParameter():

=>getParameter()-method is used to get single parameter value from request object.

Method Signature:

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

syntax:

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

(ii)getParameterNames():

=>getParameterNames()-method is used get all parameter names from request Object

Method Signature:

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

syntax:

Enumeratiom<String> e = req.getParameterNames();

(iii)getInitParameter():

=>getInitParameter()-method is used get single initialized para-value from

ServletContext and ServletConfig Objects.

Method Signature:

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

syntax:

String p = sct.getInitParameter("p");

(iv)getInitParameterNames():

=>getInitParameterNames()-method is used to get all initialized parameter names from

Context and Config Objects

Method Signature:

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

syntax:

Enumeration<String> e = sct.getInitParameterNames();

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

***Assignment***

Construct Application to demonstrate the following methods:

::getParameterNames() - use this method to collect multiple parameter names from request

::getInitParameterNames() - use this method to collect multiple initialization para-names from Context Object

----

Date: 09/04/2025 (Day-34)

Summary

(i) ServletContext is instantiated when WebApplication deployed into Server and ServletContext Object is loaded with Server Information.

(ii) ServletConfig is instantiated when Servlet-Program loaded for execution and ServletConfig Object is loaded with ServletName.

(iii) Only one ServletContext Object is created for total WebApplication.

(iv) Every Servlet Program will have its own ServletConfig object created.

(v) The information in ServletContext object can be shared by all the Servlets and JSPs of WebApplication.

(vi)The information in ServletConfig object is available to individual Servlet Program.

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

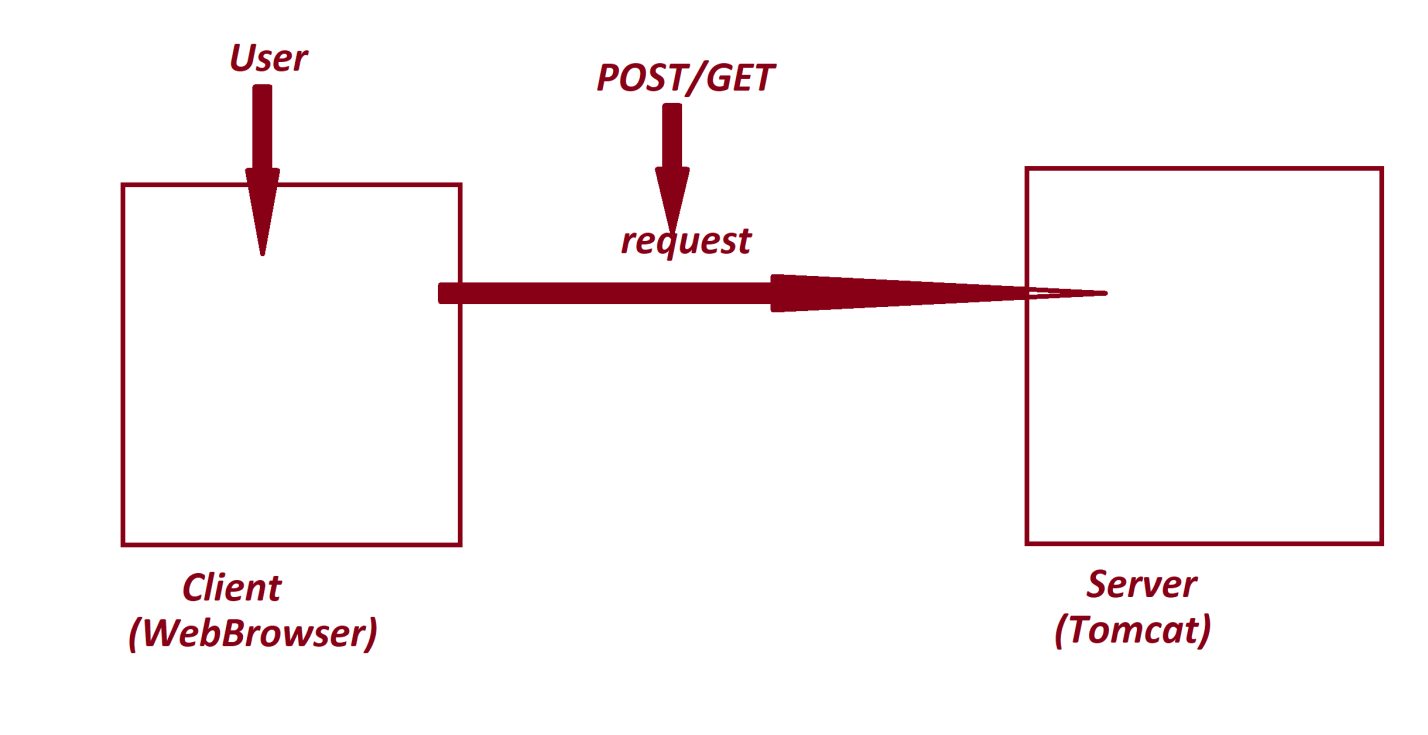
***FAQ***

define **'request'?**

* An act of asking formally for something is known as 'request'.
* The query which is generated by the user through WebBrowser is known as request.
* The request generated from WebBrowser is categorized into two types:

**1. POST request**

**2. GET request**



**1. POST request**

* The request which is raised to send some data to server is known as POST request.
* Through POST request we can send UnLimited data.
* Through POST request we can send all types of data,which means we can send Text, Audio, Video, Image and Animation data.
* The data in POST request is secure,because the data is binded to the body part of HTTP protocol.
* We use the following syntax to raise POST request:

**<form action="url" method="POST">**

**...**

**</form>**

* we use doPost()-method from 'HttpServlet' to accept only 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 limited data,which means we can send upto 10KB.
* Through GET request we can send only Text data.
* The data in GET requset is not secure,because the data is added to the query-string and displayed in AddressBar of WebBrowser.
* We use the following syntaxes to raise GET request:

***Syntax-1***

decaring method="GET" in <form> tag

<form action="url" method="GET">

...

</form>

***Syntax-2***

submit <form> tag without specifying "method"

<form action="url">

...

</form>

***Syntax-3***

* The request which is raised through hyperlinks is GET request

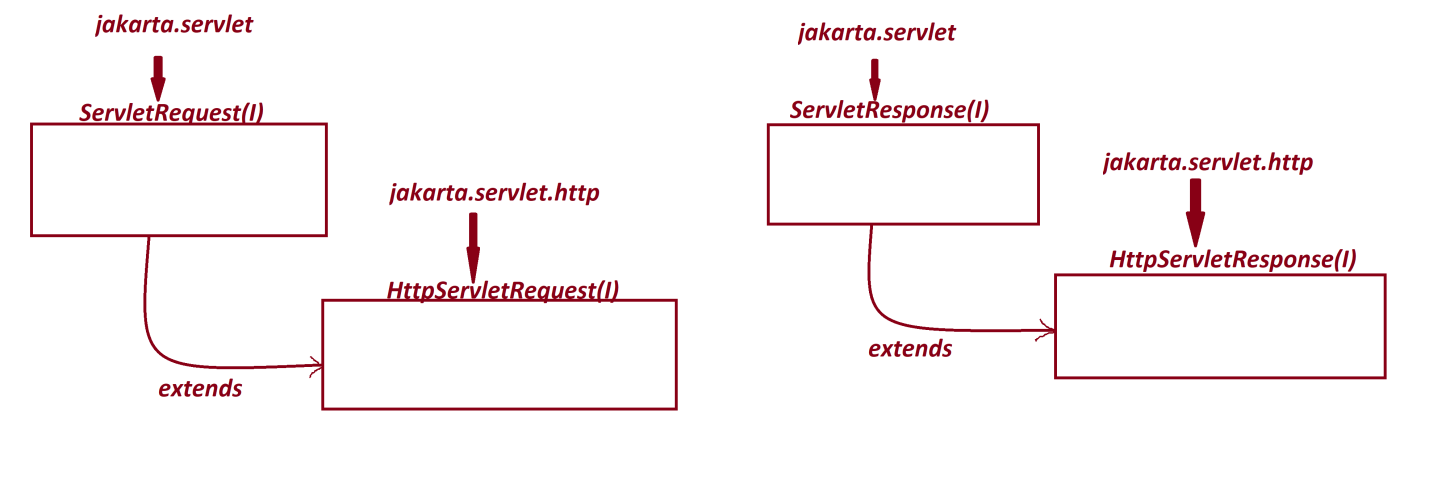
***Syntax-4***

* The request which is raised through WebBrowser by declaring servlet-url-pattern in AddressBar,is GET request
* we use doGet()-method from 'HttpServlet' to accept only 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.

Define **Session Management?**

* The process of recording the state of used and controlling the user in Session, is known as Session Management.
* We use the following Session tracking techniques in Servlet Programming to track the user in session:

*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 for 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 java.util.Enumeration<java.lang.String> getAttributeNames();

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

public abstract void invalidate();

public abstract jakarta.servlet.ServletContext getServletContext();

* we use getSession() - method from HttpServletRequest to instantiate HttpSession
* Method Signature of **getSession()**

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

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

**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 information from one Servlet to another Servlet in Session Tracking process.
* ***Syntax***

**Servlet-url-pattern?para=value&para=value&para=value&...**

***Note***

"?" - symbol represent separator b/w servlet-url-pattern and parameters.

"&" - symbol represent separator b/w parameters.

**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 Fields are not displayed to the end-user on WebBrowser.
* Through Hidden Form Fields we can send some information from one Servlet-program to another Servlet-Program in Session Tracking process.

***Syntax***

<form action="url" method="POST/GET">

<input type="hidden" name="nm" value="val">

....

</form>

---