[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

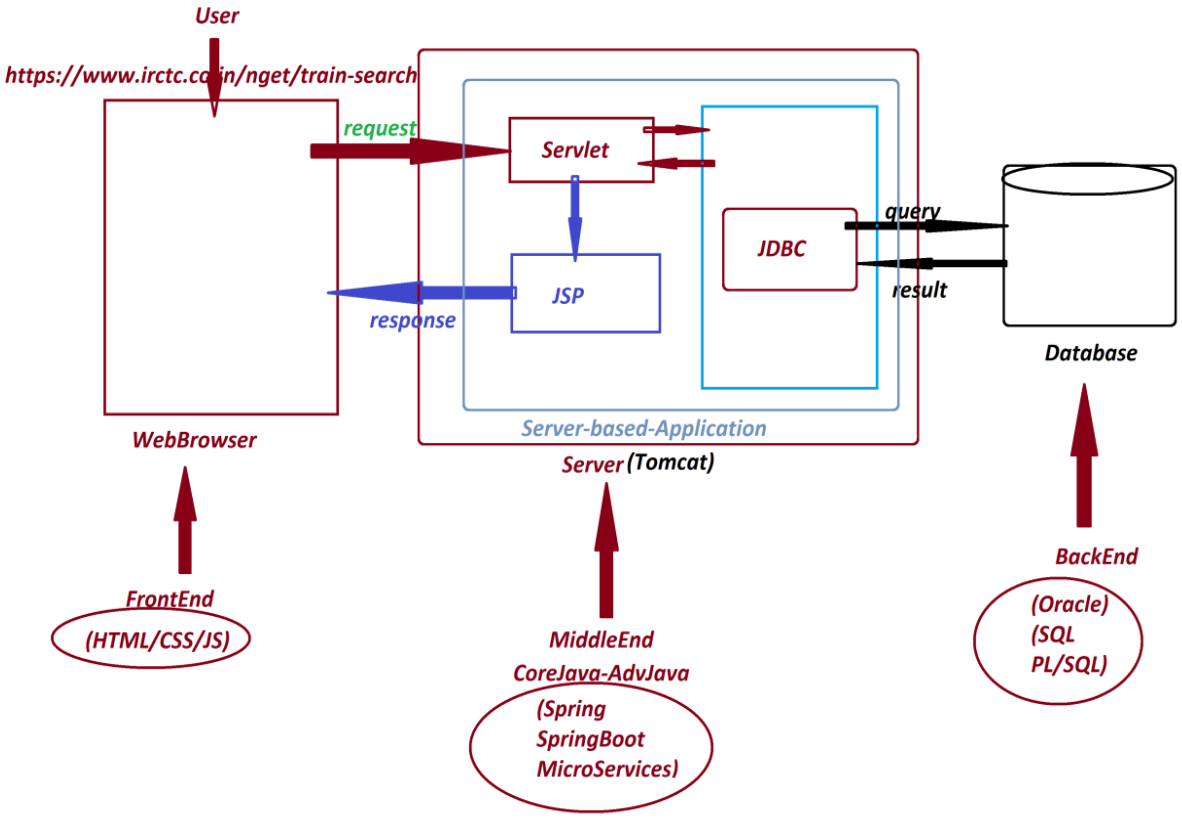
Define Server based Applications?

- ⇨ 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)

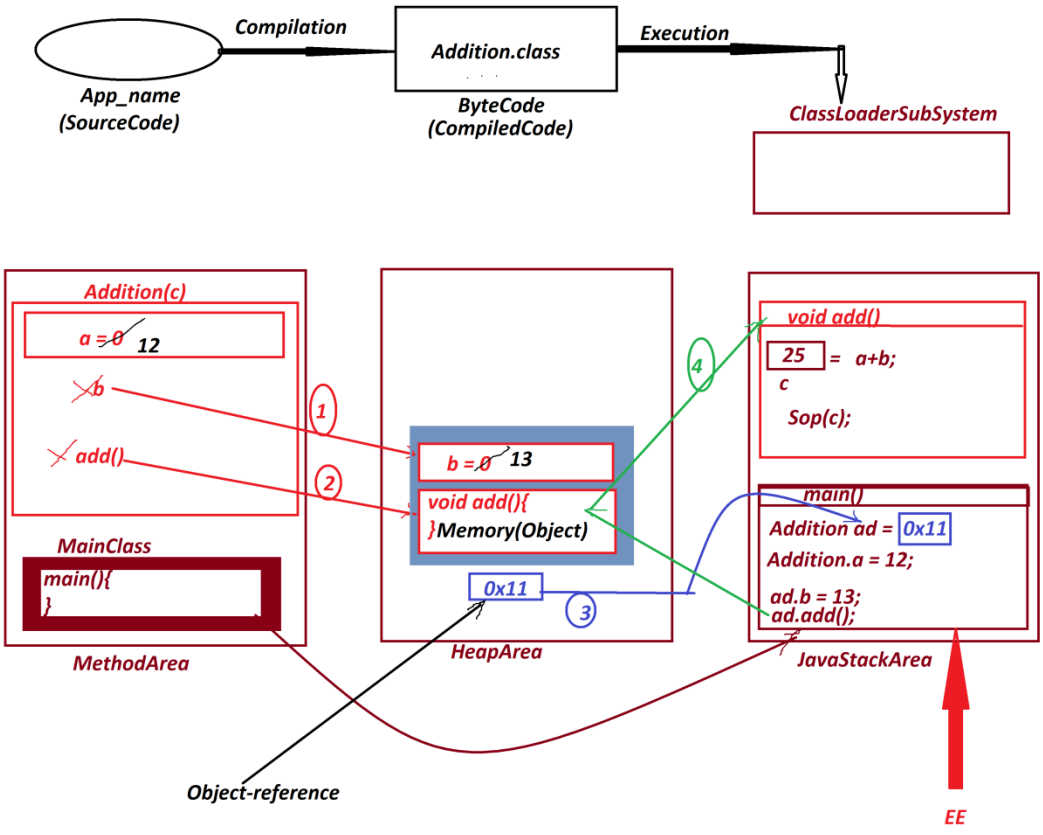
Example

```
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.

(ii) Object reference:

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

(iii) Object reference Variable:

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

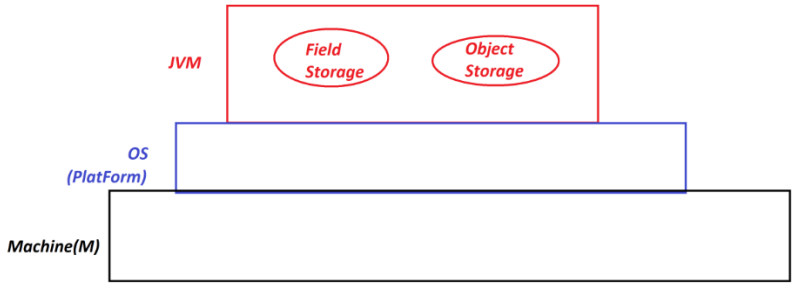
****imp***

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:

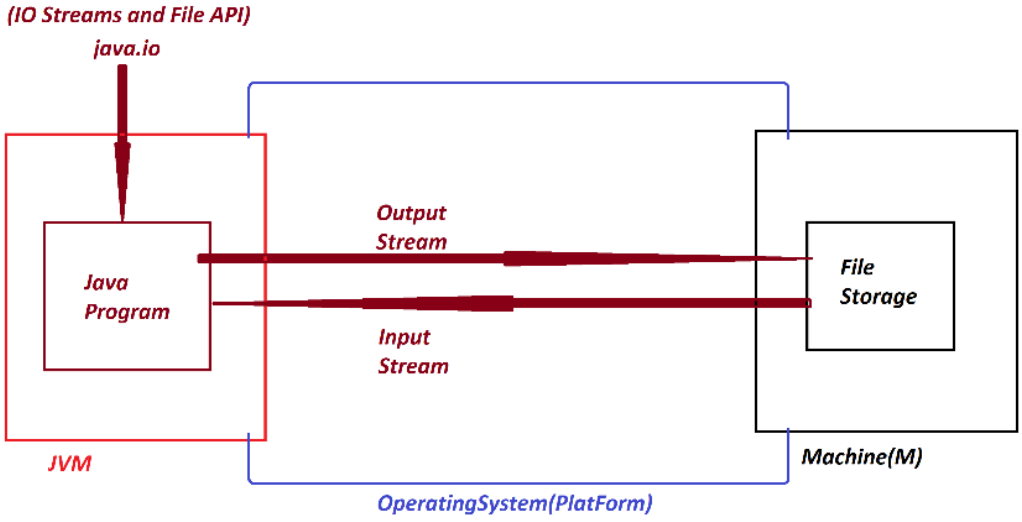
- ⇨ The Field and Object Storages which are generated part of JVM while Application execution will be destroyed automatically when JVM Shutdowns.
- ⇨ when we want to have permanent storage for Applications,then we have to take the support of any one of the following:
    - File Storage
    - Database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



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

3. File Storage:

- ⇨ The smallest permanent storage of ComputerSystem which is 'controlled and managed' by the OperatingSystem is known as FileStorage.
- ⇨ 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

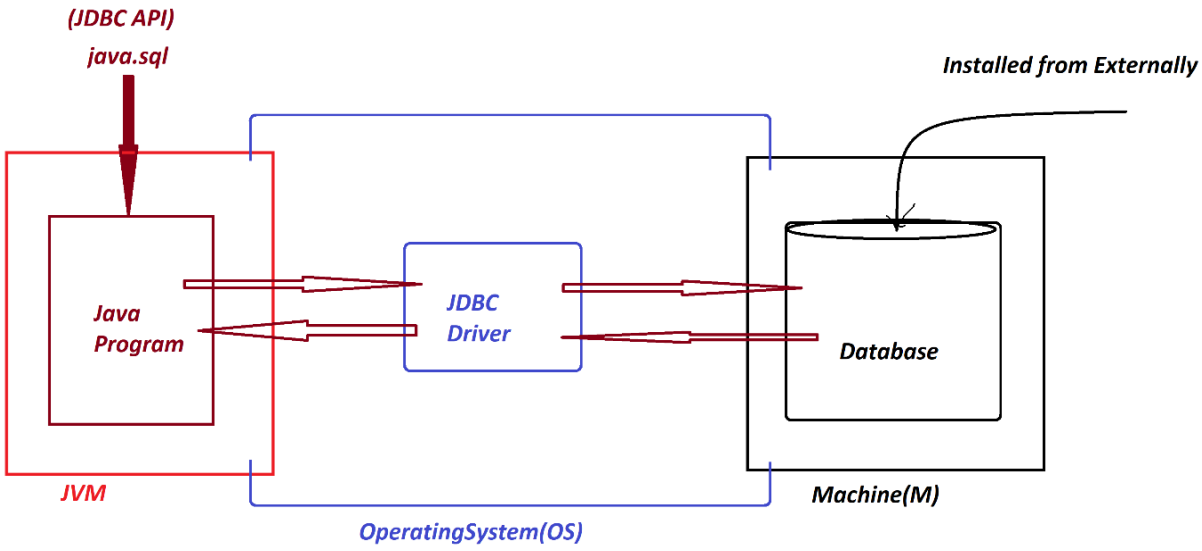
- ⇨ Because of disadvantages in File-Storage, the File-Storage cannot be taken as major backend for Java-Applications.
- ⇨ To Overcome all disadvantages of File-Storage, we use Database Storage.

=====

***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*

****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

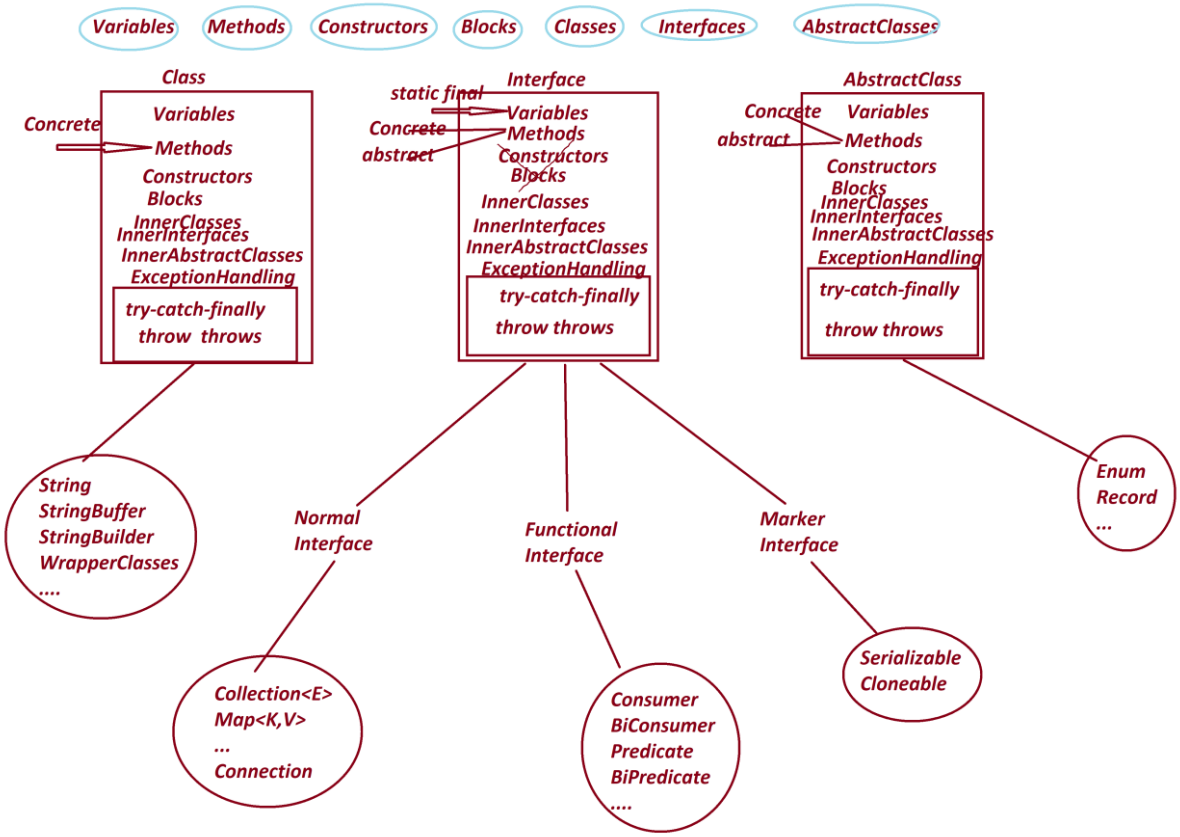
step-5 : Closing the connection from Database

****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:

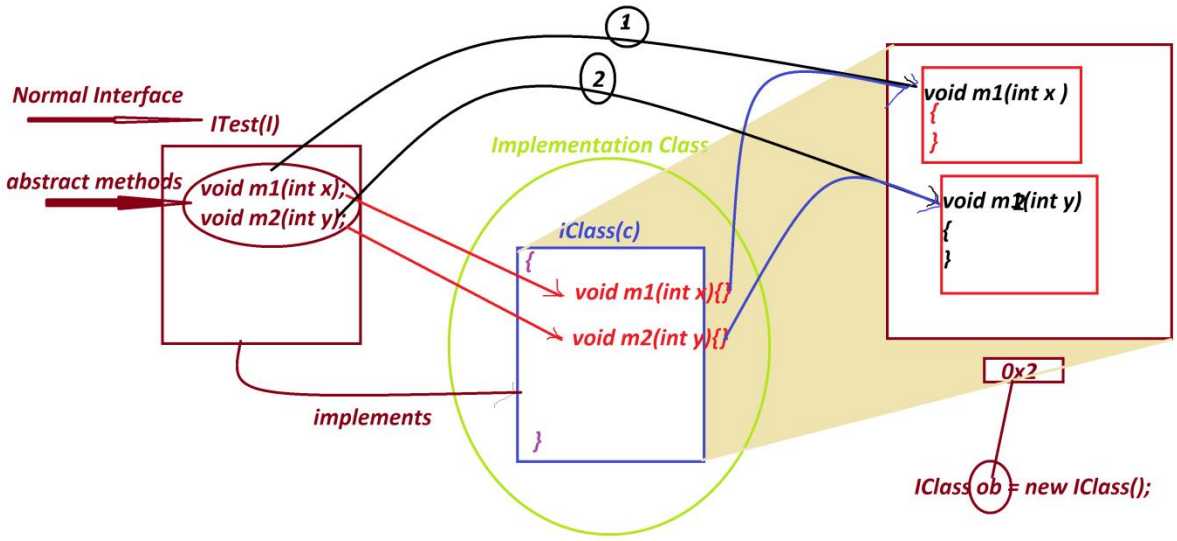
Model-1 : Interface with Implementation class with name.

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*********

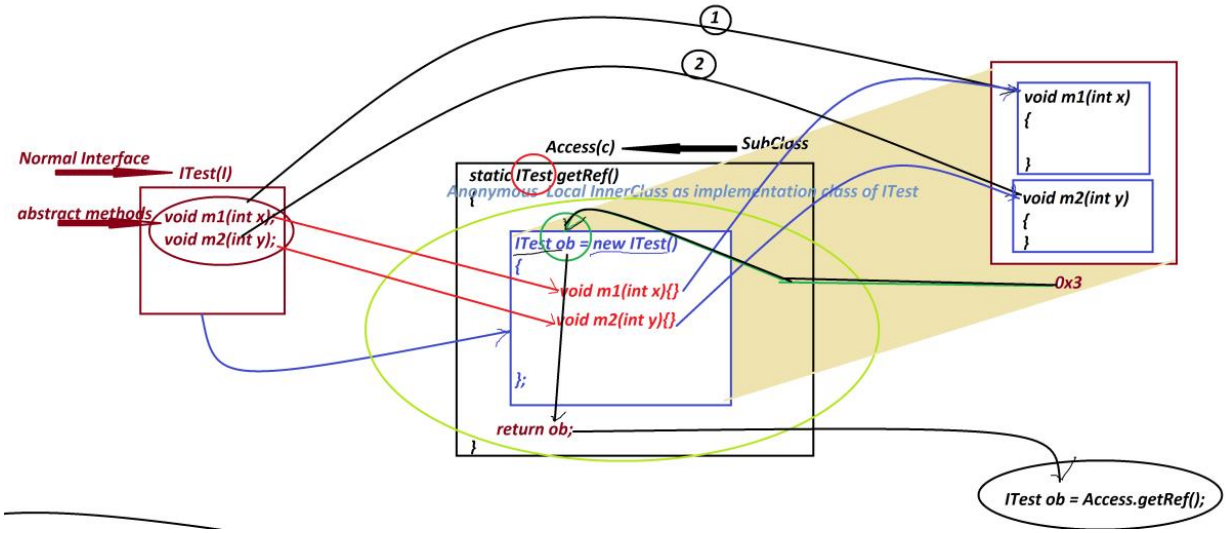
The value y:23

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:

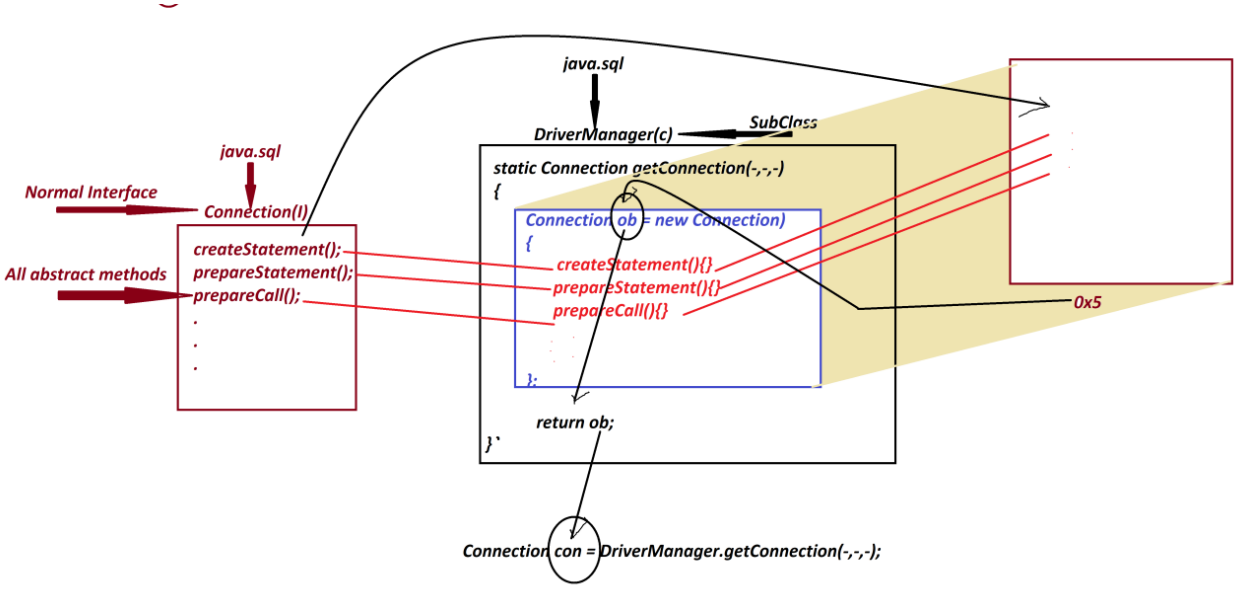
*****Implemented m1(x)*****

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. 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.

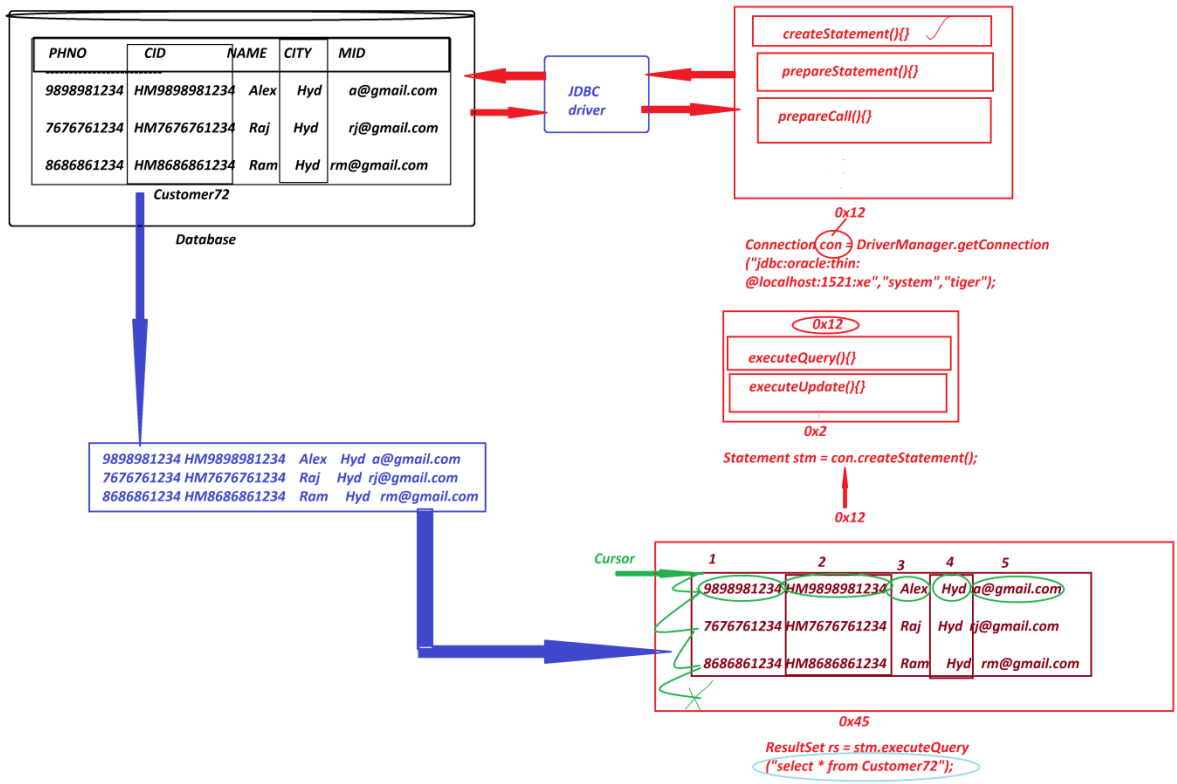
---

step-4 : Create package in 'src'

---

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

======


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
Customer72
Database
JDBC driver
createStatement(){}
prepareStatement(){}
prepareCall(){}
0x12
Connection con = DriverManager.getConnection ("jdbc:oracle:thin: @localhost:1521:xe","system","tiger");
0x12
executeQuery(){}
executeUpdate(){}
0x2
Statement stm = con.createStatement();
0x12
9898981234 HM9898981234 Alex Hyd a@gmail.com
7676761234 HM7676761234 Raj Hyd rj@gmail.com
8686861234 HM8686861234 Ram Hyd rm@gmail.com
Cursor
1 2 3 4 5
0x45
ResultSet rs = stm.executeQuery ("select * from Customer72");

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

---