UNIT-II

CHAPTER -4

PROGRAMMING FUNDAMENTALS

Computer Programming:

Computer programming is a process of designing, writing, testing, debugging and maintaining the source code of computer programs written in a particular language.

Computer Programmer: -

A programmer is someone who writes computer program.

Computer Language :

A **computer language** is a system of communication with a computer.

Programming Language :

A language is the main medium of communicating between the Computer systems and the most common are the programming languages.

Types of Programming Language: -

Basically, there are two main categories of computer languages, namely

1. Low Level Language and
2. High Level Language.

1. **Low Level Language :** Low level languages are the machine codes in which the instructions are given in machine language in the form of 0 and 1 to a Computer system. It is mainly designed to operate and handle all the hardware and instructions set architecture of a Computer. The main function of the Low level language is to operate, manage and manipulate the hardware and system components.
   Low level language is also divided into two parts are Machine language and Assembly language.
   Machine Language is one of the low-level programming languages which is the first generation language developed for communicating with a Computer. It is written in machine code which represents 0 and 1 binary digits inside the Computer string which makes it easy to understand and perform the operations.
   Assembly Language is the second generation programming language that has almost similar structure and set of commands as Machine language. Instead of using numbers like in Machine languages here we use words or names in English forms and also symbols.
2. **High Level Language :** The high level languages are the most used and also more considered programming languages that helps a programmer to read, write and maintain. It is also the third generation language that is used and also running till now by many programmers.

The High Level Language is further divides into two categories i.e POP and OOP.

Difference between POP and OOP : -

POP	OOP
(i) POP stands for Procedure Oriented Programming language.	(i) OOPS stands for Object Oriented Programming language.
(ii) Emphasis is on doing things (algorithms)	(ii) Emphasis is on data rather than procedures.
(iii) Large programs are divided into smaller programs known as functions.	(iii) Programs are divided into what are known as objects.
(iv) Most of the functions share global data.	(iv) Data is hidden and cannot be accessed by external functions.
(v) Data move openly around the system from function to function.	(v) Objects may communicate with each other through functions.
(vi) Employs top-down approach in program design.	(vi) Follows bottom-up approach in program design.
(vii) Example : C, BASIC,PASCAL etc,	(vii) Example : C++, Java, .NET, Python etc.

Features of OOPs

These concepts are called as the features of this approach (OOP).

- Object
- Class
- Method
- Message Passing
- Inheritance
- Encapsulation
- Abstraction
- Polymorphism
- Binding

(i) Object:-

- Objects are the basic run-time entities in an Object – Oriented System.
- An instance of a class is called an object.
- They may represent a person, a place, a bank account, a table of data or any item that the program has to handle.
- For example, if 'customer' and 'account' object requesting for the bank balance. Each object contains data and code to manipulate the data.

(ii) Class:-

- The basic unit of an Object Oriented Programming is the class.
- Each class defines a particular set of characteristics and behavior attributed to a group of objects.
- The characteristics and behaviors are called as the members of the class.
- Thus a class is a collection of objects of similar type.
- For example a mango, apple and orange are members of the class fruit.

(iii) Method: -

- The ability of an object is called its methods.
- Tommy, being a Dog, has the ability to bark. So bark() is one of Tommy's methods.
- He may have other methods as well, like smell(), eat() or walk().

(iv) Message Passing: -

- The process by which an object sends data to another objects or asks the other object to invoke a method is called Message passing.
- In programming language we call it interfacing.
- For example the object called Hari (the trainer of Tommy) may tell Tommy to smell by passing a 'smell' message which invokes Tommy's smell () method.

(v) Inheritance: -

- Inheritance is the process by which objects of one class acquire the properties of objects of another class.
- It supports the concept of hierarchical classification.
- For example, the bird 'robin' is a part of the class 'flying bird' which is again a part of the class 'bird'.
- In OOP, the concept of hierarchical provides the idea of reusability.

(vi) Encapsulation: -

- The wrapping up of data and functions into a single unit (called class) is known as encapsulation.
- Data encapsulation is the most striking feature of a class.
- The data is not accessible to the outside world and only those functions which are wrapped in the class can access it.

(vii) Abstraction: -

- Abstraction refers to the act of representing essential features without including the background details or explanations.
- For example, a class Car would be made up of objects like engine, gearbox, steering and many more components.
- To build the car class, one does not need to know how the different components work internally, but only how to interface with them, so as to make the car function properly.

(viii) Polymorphism: -

- The word "Polymorphism" was derived from Greek word "Poly" means many and "Morph" means form.
- The ability to take more than one form is called polymorphism.
- Generally, there are three types of polymorphism found in different languages which support OOP.
- They are Overriding, Overloading and Parametric polymorphism.

(ix) Binding: -

- Binding is a mechanism creating link between method call and method actual implementation.
- Binding is of 2 types.
  i. Static Binding
  ii. Dynamic Binding

i. Static binding : When type of the object is determined at compiled time(by the compiler), it is known as static binding or Early binding.

ii. Dynamic binding: When type of the object is determined at run-time, it is known as dynamic binding or late binding.

Basic concept of Access Specifier / Modifier for class member:

- Access specifier /modifier control the accessibility of members in the same class, other class, same package, other package.
- All the access specifier /modifier are keywords, thereby should be written in lower case letters only.

There are 4 types of access specifier /modifiers in java:

1. public
2. protected
3. default
4. private

1. Public Access Specifier:

- When a class member is declared as public it means this member can be accessed any where i.e. in any class of any package.
- To declare the class and its members as public, use public key word.
- It has the widest scope among all other modifiers.

Example :

```
package mypack1;
 public class A
{
public void msg()
        {
                System.out.println ("Hello");
        }
}
```

If B is defined in another package and we want to access members of A within B class, write following code:

```
package mypack2;
import mypack1.A;
public class B
{
public static void main (String args[])
```

```
{
    A obj = new A ();
   obj.msg ();
}
}
```

Output: Hello

Note :

PACKAGE :

- Packages are group of related interfaces and classes that provides a convenient mechanism for managing large set of interface and classes.
- To use a package, you need to import it into your program by using the import statement. Example: **import java.lang.*;**
- **java.lang** is a default package which is already available to you.
- Some examples of packages are: **java. applet, java.util, java.io** etc.

2. Protected Access Specifier:

- Members of a class declared with protected access specifier can be accessed within the same class, and it sub class within same package or in other package.
- To declare the members as protected, use protected keyword.
- The **protected access modifier** is accessible within package and outside the package but through inheritance only.
- The protected access modifier can be applied on the data member, method and constructor.
- It can't be applied on the class.
- It provides more accessibility than the default modifier.

Example :

```
package my pack;
public class XYZ
{
protected void msg()
        {
                System.out.println ("Hello");
        }
 }
```

If you want to access protected members outside the package, you will have to type extend this class using **extends** keyword.

If ABC class is defined in another package and we want to access members of XYZ class within ABC class, write the following code:

```
package mypack1;
import mypack.XYZ;
class ABC extend XYZ
{
```

```
  public static void main (String args[])
  {
     ABC obj = new ABC();
     obj.msg ();
  }
}
```

Output : Hello

3. Default Access Specifier:

- The default means, we don't specify any specifier at all, then such a class, method or field will be accessible with in the class and other classes with in same package.
- It is known as **package level access specifier** because the classes of the same package only are allowed to access the members of the class.
- It provides more accessibility than private.
- But, it is more restrictive than protected, and public.

Example :

```
 package mypack ;
 class ABC
{
 void msg()
        {
        System.out.println ("Hello");
        }
}
```

If XYZ is a class within the same package and we want to access the members of class ABC within the XYZ class, Write the following code:

```
 package mypack :
class XYZ
{
 public static void main(String args[])
{
  ABC obj = new ABC (); //Compile Time Error
 obj.msg (); //Compile Time Error
}
}
```

In the above example, the scope of class ABC and its method msg() is default so it cannot be accessed from outside the package.

4. Private Access Specifier:

- Member of a class declared with private access specifier, accessed only within the class in which there is declared.
- These members are not available for subclass and other package.
- To declare the members as private, use private keyword.
- A class cannot be declared as private.

Example :

```
class A
{
private int data=40;
private void msg()
        {
                System.out.println ("Hello java");
        }
}
public class Simple
{
public static void main(String args[])
{
    A obj=new A ();
    System.out.println (obj.data);//Compile Time Error
    obj.msg ();//Compile Time Error
}
}
```

Accessibility of Different Access Specifier : -

Let's understand the access modifiers in Java by a simple table.

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

Basic concept of inheritance : -

Inheritance can be defined as the process, where one class acquires the properties of another. It supports the concepts hierarchical classification.

Super class and Sub class: -

Definition:

Inheritance is a mechanism that allows you to build new classes from existing classes, the old class is known as **base class or super class or parent class** and new class is known as **derived class or sub class or child class.**

Super class:

A super class is a class that has been extended to another class.

Syntax:

```
<access_specifier> class <class_name>
{
        --------
}
```

e.g.:

```
public class parent
{
        --------
}
```

Sub class:

A Sub class is a class that derives from another class. A sub class is defined by using the keyword **extends** along with super_class_name.

Syntax:

```
< access_specifier > class <Sub-class_name> extends <super_class_name>
{
        --------
}
```

e.g.:

```
public class child extends parent
{
        --------
}
```

Advantages of inheritance:

- Reusability
- Saves time and effort
- Reliability

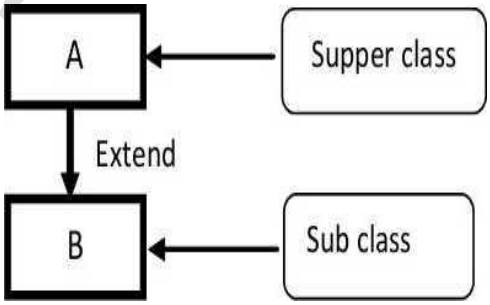
Forms / Types of Inheritance : -

There are five types of inheritance:

a) Single Inheritance
b) Multiple Inheritance
c) Multilevel Inheritance
d) Hierarchical Inheritance
e) Hybrid Inheritance

(a) Single Inheritance : -

When a sub class inherits from only one super class or base class ,it is known as single inheritance.



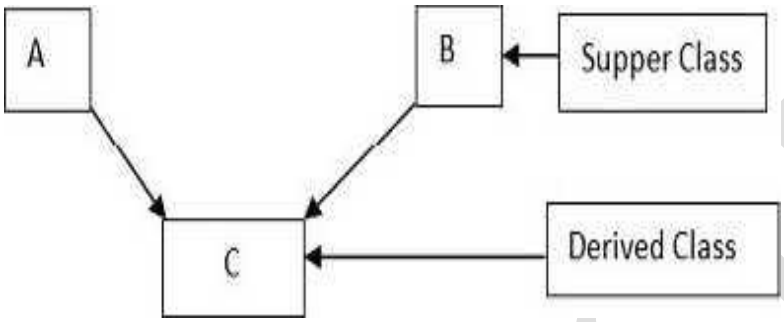
Here, class B inherits the properties and methods of class A.

(b) Multiple Inheritance : -

When a sub class inherits from more than one super classes or base classes, it is known as multiple inheritances.

Note : :- Java does not support Multiple Inheritance with classes.

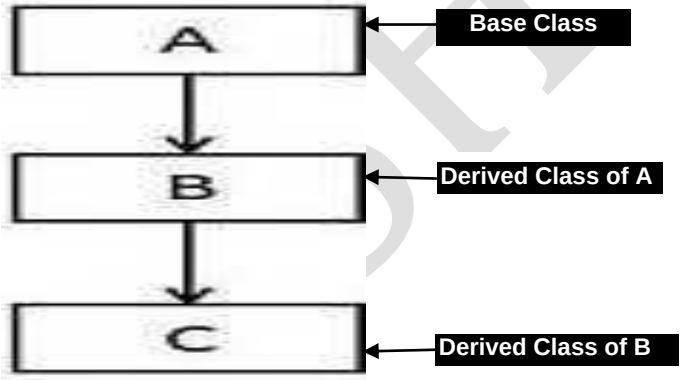
In java we can achieve multiple inheritances only through interface with the help of implements keyword.



Here, class C inherits the properties and methods of class A and class B.

(c) Multilevel Inheritance :-

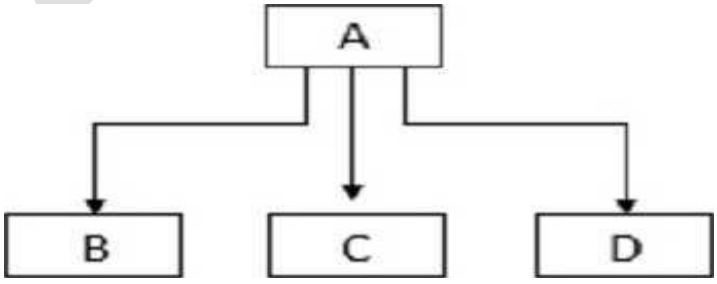
When a sub class inherits from a super class, which itself inherited from another super class, then such inheritance is known as multilevel inheritance.



Here class B inherits the properties and methods of class A and class C inherits the properties and methods of class B as well as inherits the properties and methods of class A.

(d) Hierarchical Inheritance : -

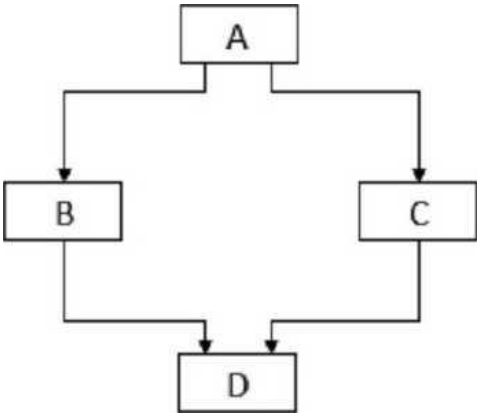
When more than one sub classes are inherited from one super class or base class, it is known as hierarchical inheritance.



Here, class B, C and class D inherits the properties and methods of class A.

(e) Hybrid Inheritance : -

When a sub class inherits from multiple base classes and all of its base classes inherit from a single base class, this form of inheritance is known as hybrid inheritance i.e. it is a combination of multilevel and multiple inheritance.



Here, class B and C inherits the properties and methods of class A. Class D inherits the properties and methods of class B and class C as well as inherits the properties and methods of class A.

Access Control of Class Members: -

Member Type	Inside Own Class	Inside Sub Classes		Inside Non- Sub Classes	
		In Same Package	In Other Package	In Same Package	In Other Packages
private	Access	Not Access	Not Access	Not Access	Not Access
public	Access	Access	Access	Access	Access
protected	Access	Access	Access	Access	Not Access
default	Access	Access	Not Access	Access	Not Access

Example 1 :

To show single inheritance or class inheritance.

```
class A
{
int x;
int y ;
int get(int p, int q)
{
x = p;
y = q;
return 0;
}
```

```
void show()
{
        System. out. println ( ” value of x = ” +x);
        System.out. println ( ” value of y = ” +y);
}
}
class B extends A
{
public static void main(String args [ ])
{
        b=new B ();
        b.get (5. 6);
        b.show();
}
}
```

Using Final Keyword : -

The final keyword provides three functionality to your program.

a) If a data member is declared with the help of final keyword, its value cannot be change.
b) If a class is declared as final, it cannot be inherited by other classes.
c) If a method is declared as final, it cannot be overridden.

Commonly used JAVA Library : -

A library is reusable software component that saves developer's time, by providing access to the code that performs a programming task.
Some commonly used Java libraries are String library, Math library, Utilities library, I/O library, etc.
Libraries are made available in program through import statement.
e.g. **import java . lang. * ;**
The **java.lang** library is imported in a Java program by default.

String : - Group of characters is known as strings.
In the Java programming language, strings are objects.

String Class : -
The String class is the class, whose instances or objects can hold unchanging strings. It means once the objects are initialized they cannot be modified or changed.
Strings in Java program are created by declaring object of string type class and initializing it with a string literal.

e.g.

```
String name= "My name is Shreyash";
```

The String objects can also be created by using new keyword.

e.g.

```
String name = new String ("What is your name?");
```

Method of String : -

String class has many methods. Some common methods are as follows:

i. charAt (int Index) : - It returns the character at the specified index.

e.g.

```
String str = " Asish ";
System.out.println (str .charAt(2));
```

Output : i

ii. concat (String Str) :- It concatenates the specified string to the end of current string object.

e.g.

```
String sl = "Asish" ;
sl = sl. concat (" Kumar ");
System.out.println (s1);
```

Output : Asish Kumar

iii. int length( ) : - It returns the length of current or this string.

e.g.

```
String str = " Asish ";
System.out.printIn (str . length());
```

Output : 5

iv. toLowerCase ( ) : - It converts all the characters of a given string to lowercase.

e.g.

```
String str = " ASISH" ;
System.out.println (str.toLowe rCase()) ;
```

Output: asish

v. **toUpperCase ( ) : -**It converts all the characters of a given string to uppercase.

e.g.

```
String str = " Asish " ;
System.out.printIn (str .toUpperCase());
```

Output : ASISH

vi. trim( ) : - It removes the spaces from the beginning and end of the current or this string.

e.g.

```
String ob="    Hello   ";
String object = ob.trim();
System.out.println (object);
```

Output : Hello

vii. Substring (int startindex) : - It returns a substring from the current string, which is starting from the start index character to the end of invoking string object.

e.g.

```
String str = "0123456789 " ;
System.out.println( str.substring(4));
```

Output : 456789

viii. String substring(int startindex, int endindex) : - It returns a substring from the current string, which is starting from the startindex character and ends at the endindex character.

e.g.

```
String str = " 0123456789 " ;
System.out.println(str.substring( 4,7));
```

Output : **456**

Math Class : -

To perform some mathematical operation in a program, Java provides a class that contain some mathematical methods is known as Math class.

The general syntax of using a method is:

Math.method_name (arguments)

Methods of Math Class :

Some commonly used mathematical methods are as follows:

i. pow() : - It is used to find the value of a number raised to another number.

Syntax:

double pow(double a, double b);

Example :

Math.pow(3, 2);

Output : **9**

ii. round () : - It is used to find the value of a number as rounded.

Syntax:

The round has two syntax versions:

public static long round (double a);

public static int round (float a);

Example :

Math.round (4.5);

Math.round(- 4 . 5);

Output : **5**

-4

Accessing My SQL Database using ODBC, JDBC : -

A database connection is the means by which a database server such as Oracle, My SQL, Microsoft SQL server, etc and client software designed in Java, C, C++ etc., communicate with each other. The client uses a database connection to send commands to and receives reply from the server.

Database connectivity with ODBC/JDBC : -

Database connectivity refers to connection and communication between application software and the database system. To connect the database with an application, a framework is used to send and execute SQL statements from or within the application code.

ODBC (Open Database Connectivity) : -

- It is the most popular programming interface for accessing relational database.
- It is Microsoft's API (Application Programming Interface) , which cannot be directly used with Java.
- It uses C interface (i.e platform independent and in secure).
- That is why Java has defined its own API (JDBCAPI) chat uses JDBC drive written in Java language.

JDBC (Java Database Connectivity) : -

- It is an API (Application Programming Interface), which consists of a set of Java classes, interfaces and exceptions.
- It allows the programmer to connect Java application to the database and provide methods for querying and updating data into the database.
- JDBC is a relational database orientation, which mainly performs following tasks:
    a) Establishing the connection with a database.
    b) Send the defined SQL statements to the database server.
    c) Process the statements to get the result.

Difference between ODBC and JDBC

ODBC	JDBC
(i) ODBC Stands for Open Database Connectivity.	(i) JDBC Stands for java database connectivity.
(ii) We can choose ODBC only windows platform.	(ii) We can Use JDBC in any platform.
(iii) We can use ODBC for any language like C,C++,Java etc.	(iii) We can use JDBC only for Java languages.
(iv) Mostly ODBC Driver developed in native languages like C,C++.	(iv) JDBC drivers are implemented in Java.
(v) ODBC is procedural.	(v) JDBC is object oriented.

JDBC Driver :

JDBC driver is a software component that enables Java application to interact with the database. There are four types of JDBC drivers.

1. JDBC – ODBC Bridge :

- It converts JDBC call methods to ODBC call methods to connect to the database.
- The ODBC drive needs to be installed on the client machine.
- It is also known as **Type 1 driver**.
- JDK 1.2 is a good example of this kind of driver.

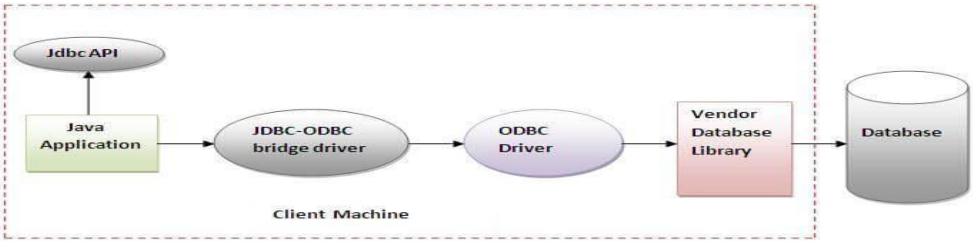


Figure- JDBC-ODBC Bridge Driver

Advantages of JDBC-ODBC Bridge:

- It is very easy to use.
- It can be easily connected to any database.

Disadvantages of JDBC-ODBC Bridge:

- Its performance degraded because it converts JDBC method into the ODBC function calls.

2. Native- API Driver (Partially Java Driver):

- This driver converts JDBC method calls into native calls of the database API.
- It is also known as **Type 2 driver.**
- The Oracle OCI (Oracle Call Interface) driver is an example of a Type 2 driver.

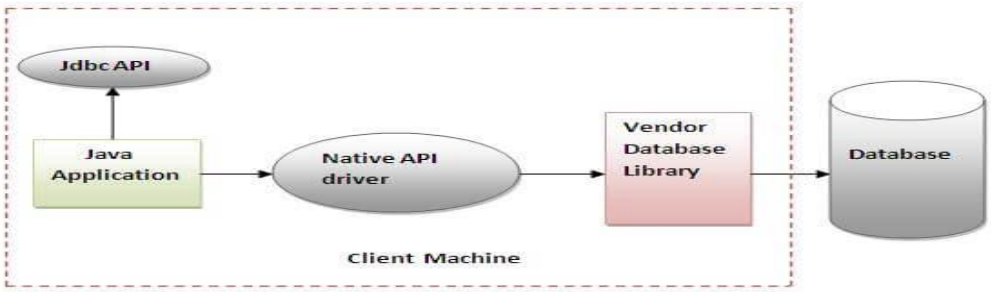


Figure- Native API Driver

Advantages of Native- API Driver:

- Its performance upgraded than JDBC-ODBC bridge driver.

Disadvantages of Native- API Driver:

- The vendor client library needs to be installed on client machine.
- The native driver needs to be installed on the each client machine.

3.Network Protocol Driver (Fully Java Driver) :

- The network protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol.
- Here, all libraries are installed on a server.
- To connect to database, these libraries accessed from the server. So, network is required.
- This driver is also known as **Type 3 driver** or middleware driver.

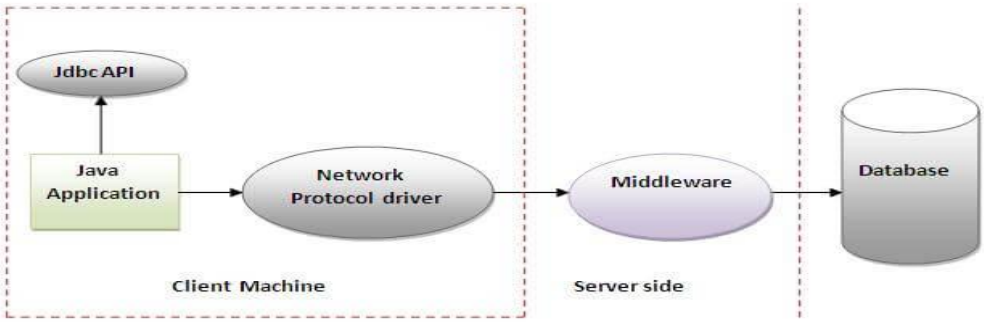


Figure- Network Protocol Driver

Advantages of Network Protocol Driver:

- No client side library is required.

Disadvantages of Network Protocol Driver

- It requires database specific coding to be done in the middle tier.
- Network support is required on client machine.

4. Thin Driver (Fully Java Driver) :

- The thin driver converts JDBC calls directly into the vendor specific database protocol.
- It is pure Java driver.
- It is also known as **Type 4 driver**.
- MySQL's Connector/J driver is an example of Type4 driver.

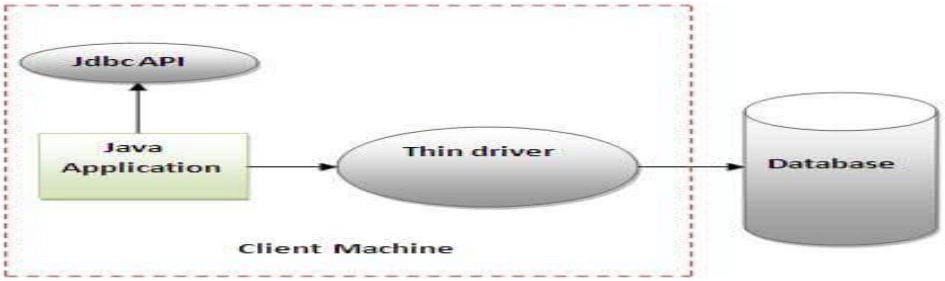


Figure- Thin Driver

Advantages of thin Driver:

- There is no software required at client side or server side.
- Better performance than all other drivers.

Disadvantages of thin Driver:

- Drivers depend on the database.

Classes used for Database connectivity: -

There are four native classes, required for database connectivity.
The classes are as follows:

1. Drive Manager Class :

- The Drive Manager class acts as an interface between user and drivers.
- It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver.
- The Drive Manager class maintains a list of Driver classes that have registered themselves by calling the method DriverManager.registerDriver ().

2. Connection Class :

- A Connection is the session between java client application and a specific database.
- The Connection interface provide many methods for transaction management like commit(), rollback() etc.

3. Statement Class :

- The Statement class provides methods to execute queries with the database.
- The statement interface is a factory of Result Set i.e. it provides factory method to get the object of Result Set.
- The simple example of Statement class to insert, update and delete the record.

4. Result set class :

- The object of Result Set maintains a cursor pointing to a row of a table.
- Initially, cursor points to before the first row.
- The simple example of Result Set interface to retrieve the data.

Connecting MYSQL Database from Java :

To connect two different types of applications (i.e. Java an object oriented language and MySQL an RDBMS (Relational Database Management System)), we need a software called MySQL Connector/j.

MySQL Connector/j is a thin Java driver that converts JDBC calls into the vendor specific database protocol.

Steps for connecting MySQL from Java, we need to do the following steps:

Step 1 : Start Net Beans IDE.

Step 2: Click on Tools menu -> Libraries

Step 3: In the Ant Library Manager dialog, From the Libraries: select MySQL JDBC Driver under class libraries.

Step 4: Click at Add JAR/Folder button and then select the downloaded and extracted driver file and click Add JAR/Folder button.

Step 5: Select the desired file and click on OK button. After that install the appropriate driver's, now establish a database connection using JDBC.

MySQL Database Connectivity in Java :

Step 1:

Import the Packages: You need to include all the packages that contain the JDBC classes needed for database programming. Most often, using **import java.sql.*** will suffice.

Step 2:

Register the JDBC Driver : Here you have to initialize a driver so that you can open a communication channel with the database.

Step 3:

Open a Connection:

Here, you can use the **getConnection()** method to create a Connection object, which represents a physical connection with the database.

Step 4:

Execute a Query: This actually requires to use an object of type Statement for building and submitting an SQL statement to the database.

Step 5:

Extract Data from ResultSet: It is suggested that you use the appropriate **getInt(), getlong(),getString(), getFloat(),getDate() etc** method to retrieve the data from the result set.

Step 6:

Clean Up: Here, it is essential to explicitly close all database resources versus relying on the JVM's garbage collection.