Friday, June 17, 2022 9:43 AM

Inference = to infer = deduction

a =b
b= c
Therefore we infer that a=c

MyInterface obj = () -> {} //Inference that RHS is an object of subclass of LHS

So compilers that programmer is writing RHS that is the implementation of type given in LHS !!!

//thisis direct
class Abc implements MyInterface // Abc is a MyInterface , Abc is a subclass of MyInterface
{
}

()->{sysout} //Compiler cannot infer as too little info

```
main()
{
        anotherMethod( ()->{} ) ;  //compiler can infer that this is the object of subclass of MyInterface
                                   // from the definition of anotherMethod

        St.forEach( () ->{} );
        //compiler can infer that this is the object of subclass of Consumer
                                   // from the definition of forEach
}

anotherMethod( MyInterface obj )
{
        Obj.methodOfMyInterface
}

forEach( Consumer obj)
{

}
```

Reflection ---- class Class
Metadata + object creation + method invocation

Exercise ---- we will ask the user to enter a class name
We will get a class Class object using forName

We will create the object of that class using no parameter constructor
We will call a method of the class ----method name entered by user

EXPLORE ----------
We called different constructors and created objects

We saw the values of properties for different objects property.get(obj)
EXPLORE --- to set the value of properties using property.set(obj , value_to_be_set)
EXPLORE ---- to get or set static properties
We saw how to call a method without parameters and return type method.invoke(obj2)
EXPLORE ----- call methods with parameters , call method with return value , call static methods
We saw how to see all fields , all methods , all constructors using class Class
EXPLORE ---- find metadata about field qualifiers , scope, method return types , method qualifiers
Which interfaces are implemented, which class is extended, which exceptions are thrown by method

---

PACKAGING classes in JAVA ------ JAR files

When we want to DELIVER our code -------- We pack it in JAR files !!!!
Acronym JAR = Java Archive

Windows Archive file = .zip
Linuz archive ---- .tar , ,rar , .gzip

---

1. Export the current project into a JAR file in some folder
2. In another project ---- use the above jar in Build Path --->Add external jars
3. In another project use any classes of the above jar by calling them .

This is set in the ECLIPSE using BUILD PATH
We set the CLASSPATH = this is the variable that will tell the JVM , where to find the **classes**
We can give multiple JAR files from where the JVM and javac will find the classes

---

Stand Alone application ---- We run only one MAIN at a time , **One Process** , One JVM forms the **application**
|
|
|
Distributed Application -------- We run an **application** that includes **2 or more processes** / JVMS that communicate with each other.

2 processes form one application = 2-tier application
3 processes form one application = 3-tier application
4 or more processes that form one application = n-tier application

---

Tier = Layers = are given specific responsibilities
Basically 3 responsibilities
1. Presentation Layer (determines HOW to show the data to the USER)
2. Business Logic Layer (determine HOW to process the data coming from USER or Going back to USER)
3. Data Store Layer (DB , FILES) (to store and retrieve the data from NON VOLATILE storage)

2 -tier

Presentation + BL	Data Store

3-tier

Presentation	BL	Data Store

n-tier

Presentation	BL1	BL2	BL3…….	Data Store

TO **Support** Distributed Application Development java provides JEE standards !!!!!

J = Java
E = Enterprise (Distributed application)
E = edition

COMPONENT BASED ARCHITECTURES !!!
Components are Integrated at each layer !!
Socket + bulb } INTEGRATE two components to get the functionality
Advantages of Component architecture

1. End User can replace only component --- cheaper
2. Socket makers can focus on making sockets and bulb makers focus on making bulbs ---
   Each one not doing everything ---- expertise improves
3. Different combinations of socket and plug can be used

When we want to integrate 2 components ----- both components MUST follow standards !!!!

Same applies for software components !!!

1. Components can be written by different vendors and made available in JARs
2. We can mix and match
3. The specific component vendor is writing efficient code due to focus
4. As I can use readymade components my TTD (time to deliver) reduces

Software components must be integrated !!! FOR integration standards are important !!!

standards are defined by INTERFACES !!!!!!

These standards interfaces are PROVIDED by JEE
One component VENDOR will implement those interfaces, and other component vendor may call the methods of those interfaces

---

JEE standards for integrating Database Drivers and Database Client in Java = JDBC

JDBC = Java Data Base Connectivity

1. We need MySQL Server ---- Readymade download install
2. We need the MySQL Driver Component that interacts with the DB ---- readymade
3. Create a Java Client --- We write
   1. create a new Java project --- JdbcClient
   2. right click ---build path -----libraries ---- add external jars ---- select the mysql-connector-…..jar ---- apply close
   3. write a class study.Client
      Main

Mysql connector =

DRIVER component = Translator

We fire query through Java <---------->DRIVER<---------->MySQL

DB Connectivity -----

1. Load the DRIVER class such that its **class Class** is created !!!
2. create a JDBC URL
3. give credentials (uname ,pwd) and connect

URL = uniform resource locator

Mysql ---IP + port

Protocol : dbvendor://IP:portname/database

jdbc:mysql://localhost:3306/ietmar22

HW -----

Write another class study.hw.Client

Main

connection

show menu

1. Insert
2. Update ---- update product set name=varname , cost=varname where id=varname
3. Delete ---- delete from product where id = varname
4. Quit

Fire different queries

Take values from user !!!
