*3 - Minor pillars of oops :

(i) Modularity : Modularity refers to breaking down software into different parts. A rigid/strong object is broken into small pieces.

(a) re-usebility.

(b) parallel development of different modules making development faster.

(c) Module is easy to debug update and modify.

(d) Maintainability.

(ii) Persistance : Storing data for future use.

(iii) Concurrency : Performing several tasks ~~multiple~~ simultaneously or in parallel.

* 4 - Important pillars of oops :

1) Abstraction : Abstraction is a process of showing only essential/necessary features of an entity/object to the outside world and hide the other irrelevant information

2) Encapsulation: Encapsulation means wrapping up data and member functions (method) together into a single unit. i.e class Encapsulation automatically achieve the concept of data hiding providing security. to hide data by making the variable as private and expose the property to access the private data which would be public

3) Inheritance: The ability of creating a new class from an existing class. Inheritance is when an object acquires the property of another object. Inheritance allows a class to acquire the properties and behaviour of another class It helps to reuse, customize and enhance the existing code. So it helps to write a code accurately and reduce the development time.

4) Polymorphism: Poly means many & morphs means forms.. A subclass can define its own unique behaviour. and still share the same functionality.

MATRIKAS

Date: / /

(a) Static Polymorphism: (compile Time)

*Compile time polymorphism: It is also known as static polymorphism. This type of polymorphism is derived by function overloading. There are several methods present in a class having the same name but different types/order/number of parameters.

At compile time, java knows which method to invoke by checking the method signatures. So this is called compile time polymorphism. or static binding.

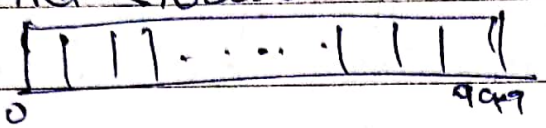
(b) Dynamic polymorphism: (Run time) If a subclass [child] extends to a super class [parent], so we can have same property of it. we can also have methods with same name & same parameter (not in same class). This is called overriding.

Collection Framework

Date: / /

individual variable cannot be stored much. int a, int b, int c, so array is used.

'Array' = int student [] = new Student [1000]



← Here we can store many values.

Disadvantage = (1) Wastage of memory. (fixed in size)

(2) Can hold homogeneous type object of one class. This can be solved by creating object of array.

(3) Not implemented on standard DS.

* Collection: Collection group of objects. It is a container object; Homogenous & Heterogenous.

Why? → We can send used to store multiple object as single group and send all object from one class to another class with return type & argument.

Date: / /

* Public Static Void main (String args[])

b command line arguments

public → access specifier, can be accessed out side the class
• main method must be public

static → As it is static so, no need to make its object, out side class

Void → No return value,

String → It is a class as 'S' is capital letter

args[] → Its Object name of the class String of type array. args can also be changed, and its a variable, information is stored in this args array of type String.

main → 'main' is name of method

psvm → JVM always executes from here

instead of 'public static' we can write 'static public'

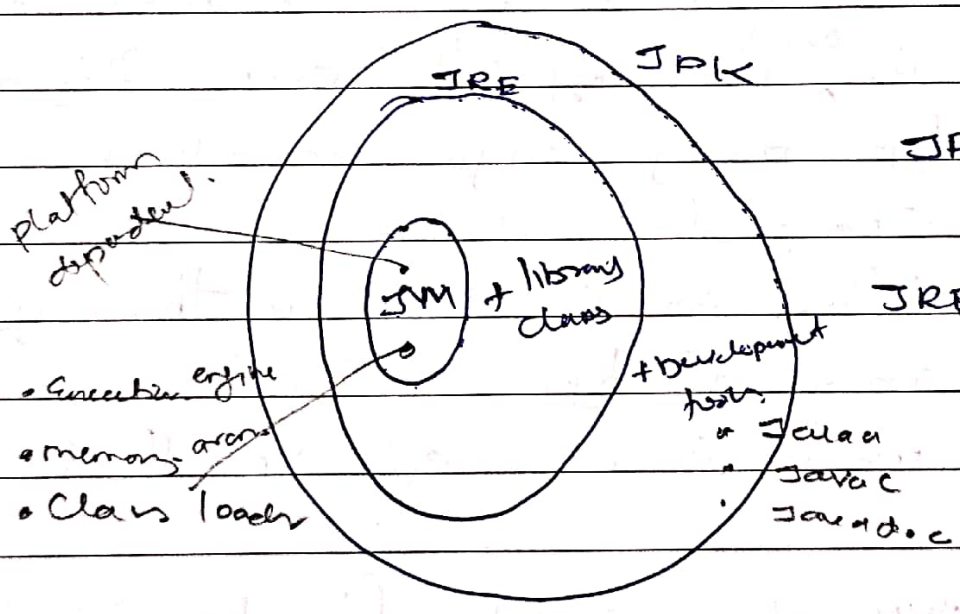
Date: / /

* JDK, JRE & JVM

JDK: To develope & run ~~programs~~ application JDK provides environment
(Java Development Kit)

JRE: Provides environment for JPK
(Java Runtime Environment)

JVM: It is an interpreter : executes the program line by line.
(Java Virtual Machine)



JPK = JRE + Develop tools.

JRE = JVM + Library Class.

* Is JRE platform dependent or independent?

→ JRE is also platform dependent

* Ultimate base Class in java class hierarchy?

→ 'object' class in 'java.lang' package

MATRIKAS

when are the

Date: / /

* Reference type in Java,
→ ① Strong Reference → default type of Reference / → When NULL, collects garbage
not eligible for garbage collection
② Weak Reference → should be explicitly specified & can be used for garbage collection. [java.lang.ref.WeakReference] → when no memory it is back (losing)
③ Soft Reference
④ Phantom Reference → eligible for garbage collection, before putting to garbage it is put into queue called Reference queue - after calling finalize method on these

* What is Widing & Nowing?
* Automatically done smaller → Longer data type is 'Widening'
Manually done by using type casting is called 'Nowing'.

* Print without Semicolon → if, switch, while

* Can code gets executed, without main method? How?
→ Yes, by using static block, it is a group of statements. Not gets executed only once when the class is loaded into the memory by Java Class loader. So called as static initialization block.

* SOP: System : it is class (java.lang)
out : it is defined in System class with public, static, final.
println: This method inside PrintStream class which is of type PrintStream object at public keyword

MATRIKAS

Date: / /

* Constructor Chaining: This operator we can call other constructor in same class.

* .Dispose: Used to free unmanaged resource like files, database connection etc at any time.

* Finalize: Can be used to manage free unmanaged resource (when you implement like files, database connection etc held by an object before that object is destroyed.

* Shutdh Shutdown Hooks: special construct that allows developers to plug in piece of code to be executed when jvm is shutting down.

* Final keyword: If a class, variable, method is final they cannot be changed, or overridden.

Finally block: Exception. Executed when exception is handled or not.

Finalize method: Clean up processing before garbage collection.

* Checked Exp: Normal exp.
Unchecked Exp: Connected to outside resource, I/O exp

Date: / /

* Dynamic method dispatch?

A obj = new C; //runtime compile time.
obj.show(); //runtime polymorphism / dynamic method.

* Interface: same as abstract like class, all methods in interface are abstract

* Array	Array List
(1) fixed size DS	(1) Expandable collection
(2) Cannot change length	(2) Can change.
(3) Both primitive & objects	(3) Cannot store primitives and only store objects.

* HashSet	HashMap
(1) Implements Set interface	(1) Implements Map interface
(2) In hash-set we store objects (elements or values)	(2) Hashmap is used for storing key & pairs. It stores it maintains the mapping of key

Hash Map	Hash Table
(1) HashMap is non synchronized	(1)

Vector	ArrayList
(1) Uses array DS	Uses array DS
(2) Synchronized.	Non-synchronized
(3) dynamically resizeable	(3) Dynamically resizeable increases by 50% when size is increased

Date: / /

*Synchronized collection classes: for thread safe.
① arrays ② Properties ③ vector ④ Hashtable.

* Iterator.	Enumeration.
① In Iterator, we can ~~dd~~ read & remove element while traversing element in collection.	① Using Enumeration, we can only read element during traversing element in collection.
Iterator	List Iterator
① Can traverse in one direction only (forward)	① List iterator traverse both in forward & backward direction.
* String Buffer	String Builder.
① Synchronized / thread safe	Non Synchronized / Not thread safe

*Serialization: a mechanism of converting state of an object into byte stream.
Deserialization: Converting an ~~object~~ byte streams to Object.
*State data members and transient data members are not saved via serialization process.
*
*Metadata: structured information about a document.
*Byte code: called as portable code form of instruction set designed for efficient execution by software developer.

MATRIKAS

Date: / /

*Reflection: It is an API which is used to examine or modify the behaviour of methods, classes, interfaces at runtime.

*Java applet: Small application written in Java programming langage, or another programming language that compiles to Java bytecode & designed to run in the java bytecode.
5 methods → init(), start(), stop(), paint(), stop()

*Thread: A thread is an independent path of an execution within a program. Many threads can run currently within a program.
*2 types of thread: User & Daemon. User one is primary thread. JVM will wait for user thread to complete its task before terminating.

*Cycle of thread: (1) Runnable: after a newly born thread is started, the thread becomes runnable, but the thread is executing its task.
(2) ~~Waiting~~ Waiting: Sometimes, a thread transition to be in waiting state while the thread waits for another thread to perform a task.

MATRIKAS

Date: / /

*Race Condition: Can be avoided by proper Thread Synchronization in critical sections.
Thread synchronization can be achieved using a synchronized block of java code

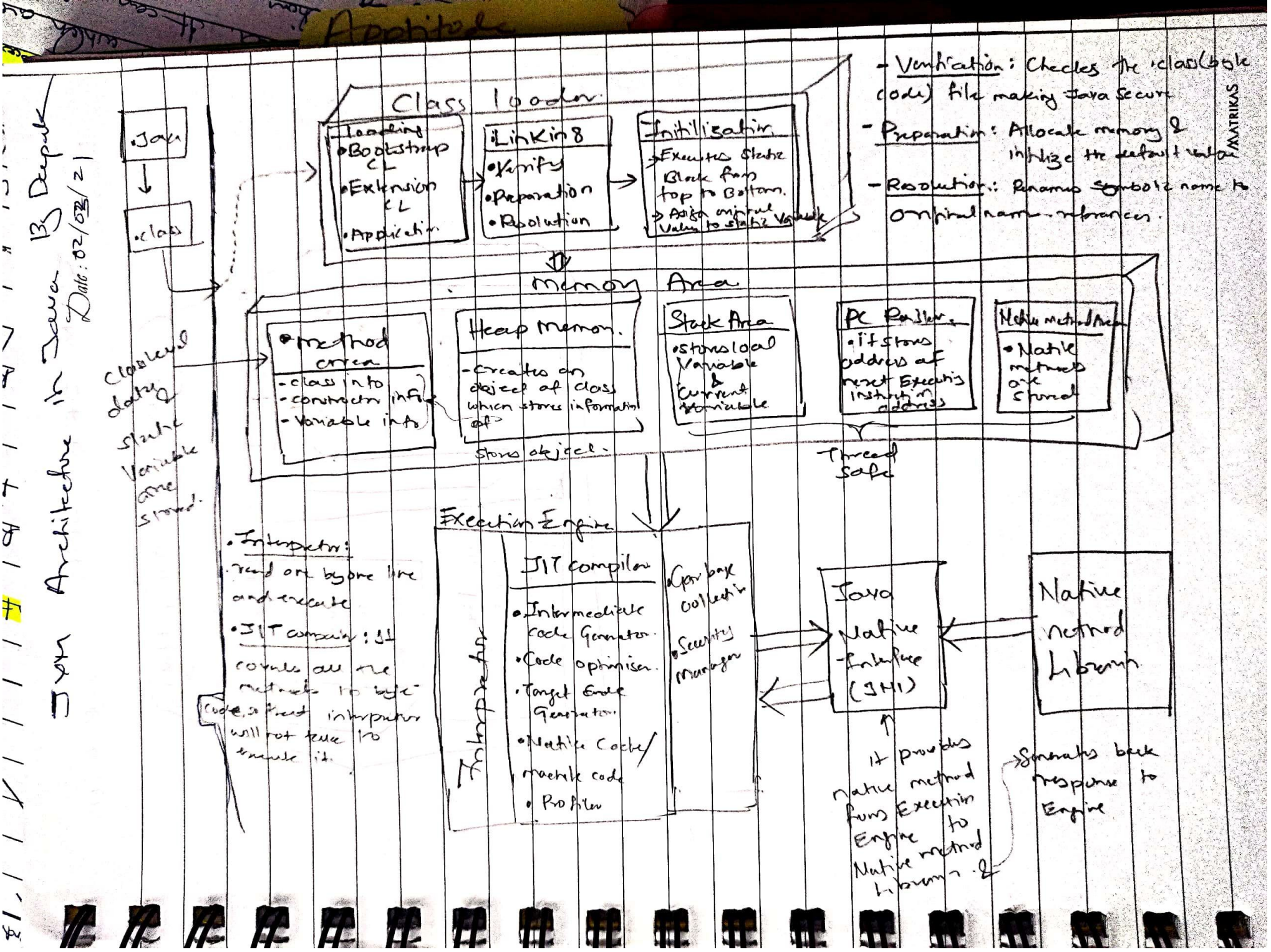
*How Synchronization is acquired?

Declare a method as synchronized, then it is known as synch method. When a thread invokes a synchronized method, it automatically acquires the lock for that object releases it after the thread completes the task.

Java Architecture in Server

By Deepak

Date: 02/02/21



JVM JRE JDK

Date: 02/05/21

Deepak