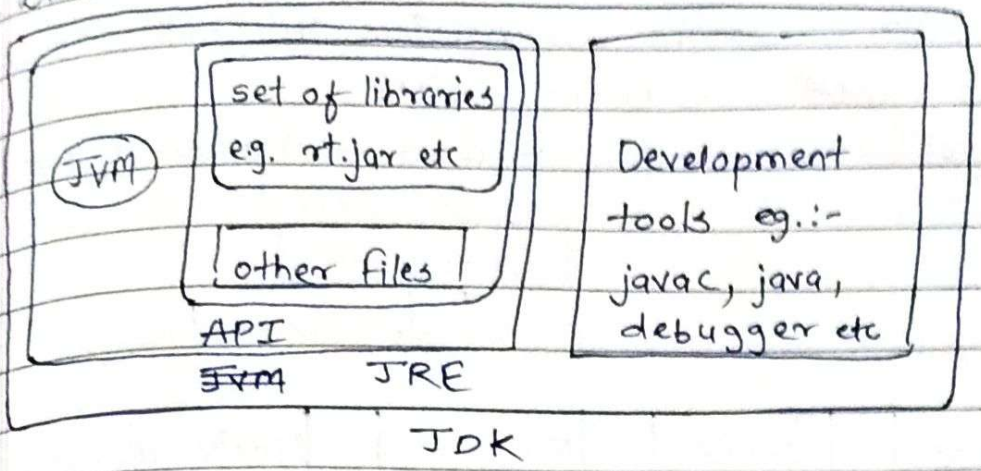
ASSIGNMENT -2

Page No. | Monday
Date | 20 | 3 | 23

Java Basics.

Q. What is difference between JDK, JRE and JVM ?



JDK : JDK is a s/w development environment used for developing java applications and applets. It includes JRE, interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc), and other tools needed.

JDK is a kit that includes :

1. Development tools [to provide an environment to develop app]
2. JRE [to execute java program]

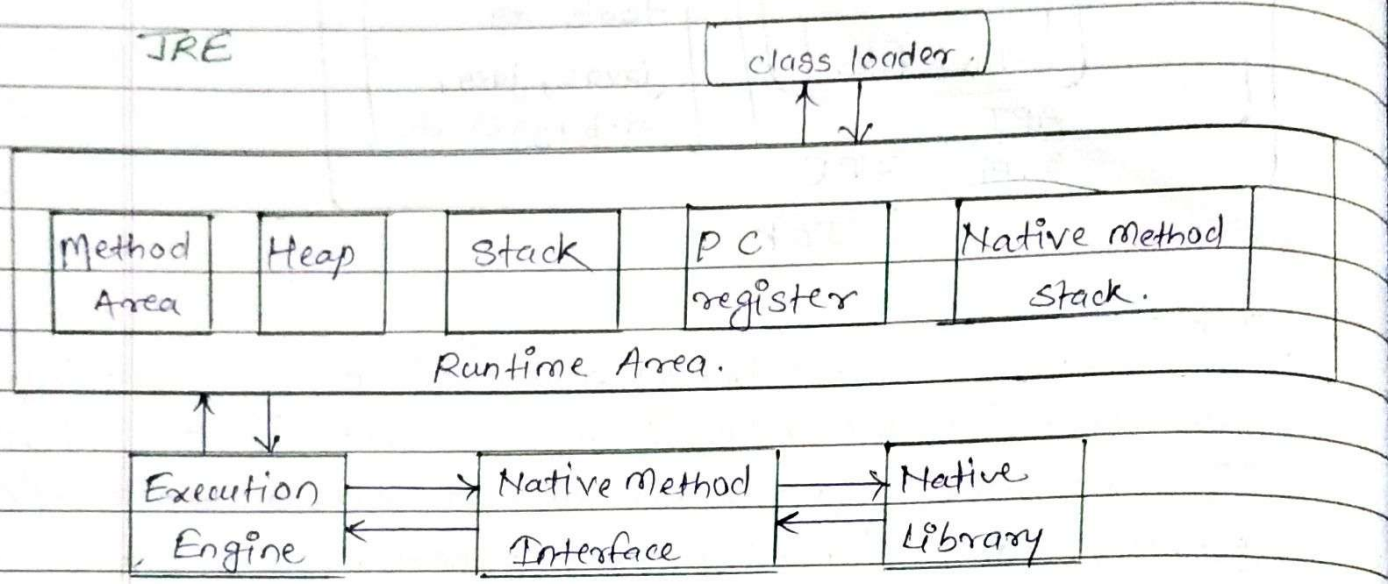
JRE : Java Runtime Environment also written as "Java RTE". provides minimum requirements of executing a java application. It consists of JVM, core classes and supporting files. Basically, JRE is an installat° package that provides an environment to 'only run' the java prog.

JVM : Java Virtual Machine is a important part of both JDK and JRE because it is whatever java program we run it goes to JVM & JVM is responsible for executing the program line by line, hence also known as interpreter. It includes Java Hotspot client and server virtual machines. JVM is specification where working of JVM is specified. But implementation provider is independent to choose algo. Whenever we write a java command to run java class, an instance of JVM is created.

Q. Explain JVM architecture.

JVM is a specification where working of JVM is specified. But Its implementation has been provided by Oracle and others. Its implementation is known as JRE.

Whenever we run a java command an instance of JVM is created.



1) class loader : is a subsystem of JVM which is used to load class files. Whenever we run java program, it is loaded first by the classloader. Three buildin classloaders:

1. Bootstrap classloader
2. Extension class loader.
3. System lapplicat^{n} classloader.

2) class method area : stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap : It is runtime data area in which objects recides.

4) Stack : It holds local variables and partial results, and plays a part in method invocation & return.

Each thread has a private JVM stack, created at the same time as thread. A new frame is created each time a method is invoked & destroyed when it invocation completes.

5) Program Counter Register : contains the address of the JVM instruction currently being executed.

6) Native method stack : contains all the native methods (C, C++) used in applicat°.

7) Execution engine : contains a virtual processor, Interpreter : Read bytecode stream then execute the instructions.
Just in time compiler : Improves performance by identifying repeated code (loops, method calling again & again) & stores it to make things easy for interpreter.

8) Java Native Interface : JNI is a framework which provides an interface to communicate with another application written in another lang like C, C++, assembly, etc. Used to send o/p to the console or interact with OS libraries.

Q.2 What is JIT compiler? [dynamic / runtime compilations]

→ JIT stands for Just in time compiler which is a part of JVM. that optimizes the performance of the application. It is also known as dynamic compilation. More commonly it does bytecode translation to machine code.

It includes two approaches AOT (Ahead OF Time) and Interpretation to translate code into machine code.

following optimizations are done by JIT compiler:

1. Local optimizat°, 2. control flow optimizat°, 3. Dead code elimination, 3. Global optimization.

ADV. of JIT :- 1. Requires less memory usage
2. The code optimization is done at runtime.
3. Uses different levels of optimization.
4. Reduces page faults.

DisAdv of JIT :-
1. Increases complexity of prog.
2. prog. with less line of code does not take benefit of JIT compiler.
3. uses lot of cache memory.

JIT compiler dynamically generates machine code for frequently used bytecode sequences in Java applications during their execution.

3 What is class loader?

Java class loader is a part of the JRE that dynamically loads java classes into the JVM. When an application requires a class at this point Java class loader is called by the ~~JRE~~ JVM.

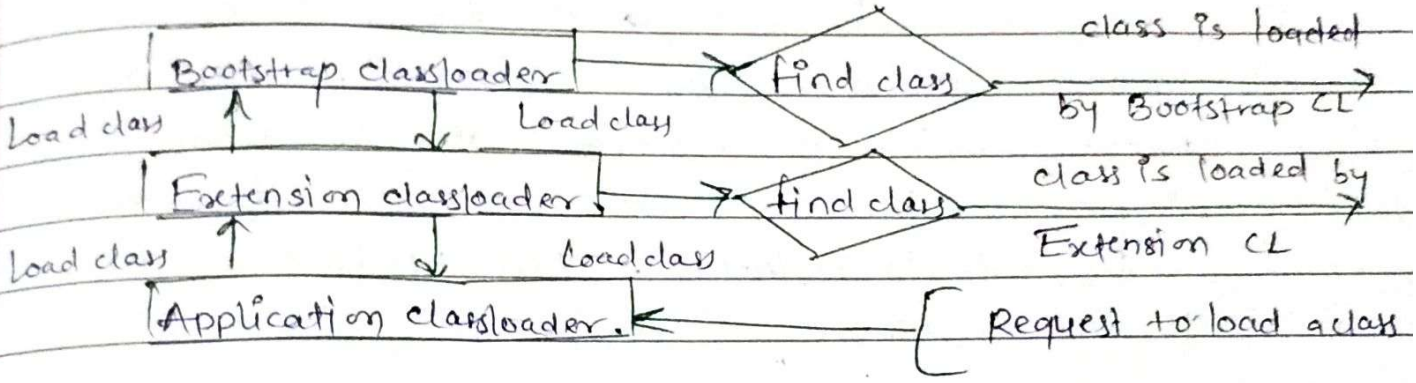
Types of class loader:

1) Bootstrap classloader: Machine code which kickstarts

the operation when the JVM calls it. It is not a java class. Its job is to load the first pure Java classloader. Loads classes from rt.jar Also called as Primordial classloader.

2) Extension Classloader :- is a child of Bootstrap classloader. Loads the extensions of core java classes from the respective JDK extension lib. Loads files from jre/lib/ext.

3) System classloader :- also known as application classloader. Loads the application type classes found in the environment variable. CLASSPATH. This is a child of extension classloader.



Q.4 Explain various memory logical ~~patterns~~ partitions.

Mainly there are 3 types of mem. logical partitions:-

1) Fixed partitioning :⇒ The memory is divided into non-overlapping sizes that are fixed, unmoveable, static. The system divides into fixed size blocks & allocated a block to process when it asks for execution A process may be loaded into a partition of equal / greater size and is confined to its allocated partition. Here as the sizes are fixed internal and external fragmentation arises which decreases the efficiency. The degree of multiprogramming is also fixed. But the implementation of fixed size partitioning is easy & also easier for OS to manage.

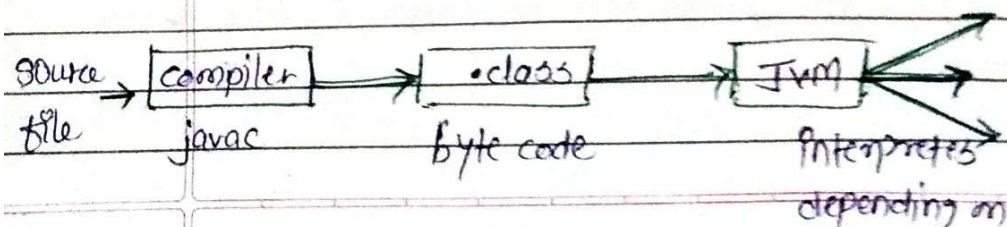
2) Variable partitioning :- Here the memory is divided into ~~k~~ partitions already but they are of ~~ver~~ different sizes. So that if we get some small size process or large size process then we can store it in respective block accordingly. Here too the degree of multiprogramming is fixed ~~and~~ but greater than fixed partitioning. Still we have fragmentation to some extent.

3) Dynamic partitioning :- Here in first place the whole memory is considered as a one block. A process gets memory block at runtime as required. Here degree of multiprogramming ~~is~~ varies. ~~and~~ ~~also~~ Internal fragmentation is avoided here. But we face external fragmentation. Difficult to implement. Here we have no limitation on size of process.

Q.5 What gives java it's 'Write Once Run Anywhere' nature?

⇒ Write once Run anywhere means we can write java code on one machine and can share it on other m/c. This is because the java compiler creates .class file which contains byte codes & is platform independent. All this happens because the JVM connects the byte code & platform.

The JVM takes .class as input and gives platform understandable machine language. as output making java byte code run anywhere.



Q6 Explain history of Java & who Invented Java?

Java was developed in 1991 by 'Green team' lead by James gosling. Publish as open source in 1996 / 95.

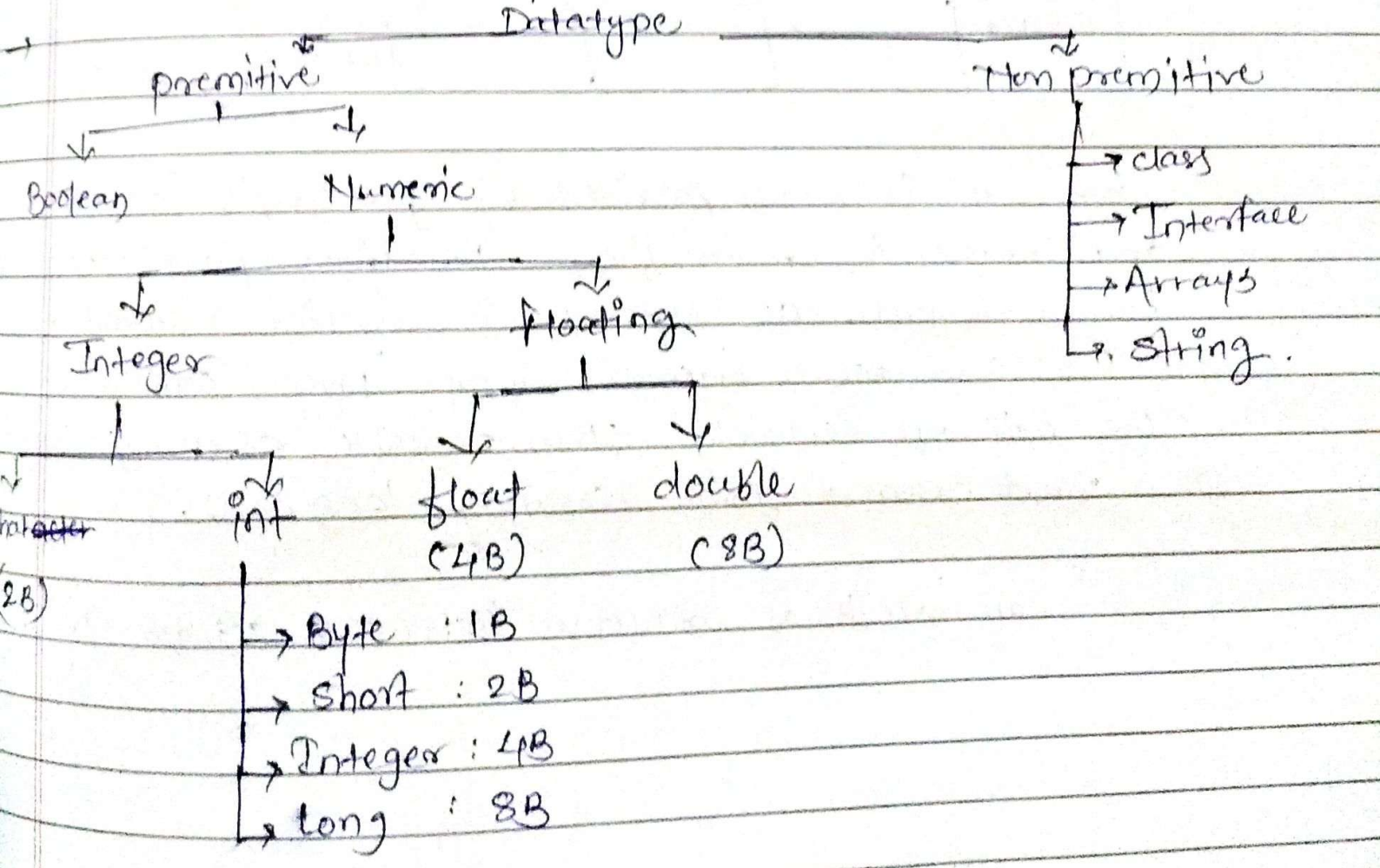
The initial ~~name~~ project ~~for~~ Java name for Java was 'Green' as requirement is to reduce memory & CPU usage.

First name for Java was 'oak' but as it was already in use then they decided to go for famous coffee name 'JAVA'.

First product of Java was a '*7' remote control'.

After this they came up with applet & Hot java browser. & gain popularity from there onwards.

Q7 List various datatypes in Java.



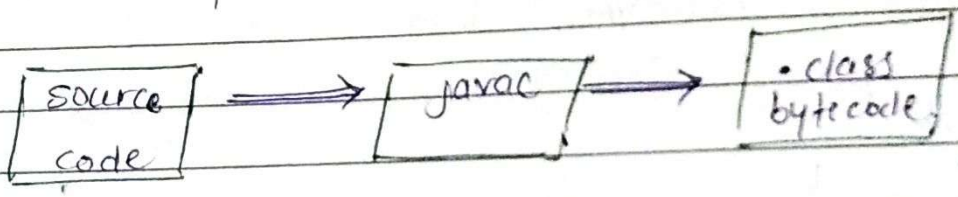
Q. What is difference between System.out.print(), System.out.println(), System.~~out~~.print();

System.out.println() :- Used to print the message but cursor moves to the next line after printing.

System.out.print() :- Used to print the message but cursor remains on the same line.

~~System.out.p~~ System.err.print() :- used to display error message. The op is displayed in red color.

Q What is bytecode ?



Bytecode is a code provided by java compiler as otp and stored in .class file. It gives portability feature to java. as bytecode is machine independent. We can read bytecode using 'javap' command. The bytecode contains JVM readable assembly lang. object oriented java assembly lang code.

We can call it as structure instruction set for JVM.

Page No.
Date

Q. Difference between bytecode & jar machine code.

	Bytecode	Machine code
1)	Bytecode is considered as the intermediate code	Machine code is considered as the low level code.
2)	Bytecode consist of assembly lang code which can be understood by JVM.	Machine code is consists of binary instructions that are directly understood by ~~JVM~~ CPU.
3)	It is non runnable code, generated after compilation of source code and it relies on an interpreter to get executed.	Set of instructions in binary format & directly runnable by CPU. No need of any interpretation.
4)	Platform independent.	Platform dependent.
5)	More readable than Machine level lang.	Non readable.

Q What is diff between jar file and runnable jar file?

	Jar file	Runnable jar file
1)	Jar file is a java application which requires a command line to run, a runnable JAR file can be directly executed by double clicking	Runnable jar file allows a use to run java classes without having to know class names and type them in command prompt, rather the user can just double click on the jar file and the program will fire up.

A (Java Archive) is a package file format typically used to aggregate many java class files associated metadata and resources into one file to distribute application software or lib on the java platform.

A runnable jar allows java classes to be loaded just like when a user clicks one exe file.

Q What is diff bet' Runnable jar file & exe. file?

Runnable jar file	exe file.
1) Jar file are like dead body	- exe. file are like living men.
2) Jar file is the combination of compiled java classes	- Executable jar file is also combination of compiled java classes with main class.

Q. How is C platform dependent language?

1) C is a portable programming language because it is not tied to any HW of system.
2) We can say, it is a HW independent language or platform independent language.
3) That is why 'C' is called 'portable language'.
4) C programs does not depend on actually but the executable file that is generated at the end for running the C programming many depend on a platform

5) When you use os you get other extension for executable files.

Q What is difference between path & class path?

Path	class Path.
1) Path variable is used to set the path for all java software tools like javac.exe, java.exe, javadoc.exe, and so on.	1) classpath variable is used to set the path for java classes.
2) Variable name : PATH Variable value : c:\program files\Java\jdk 1.7.0.21\bin	2) Variable name : classpath Variable value : c:\Programfiles\Java\jre 1.6.0\jre\lib\rt.jar

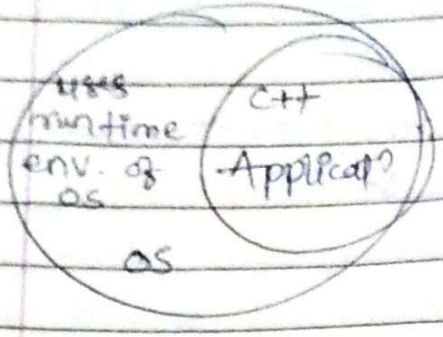
Java features :→ Java T point

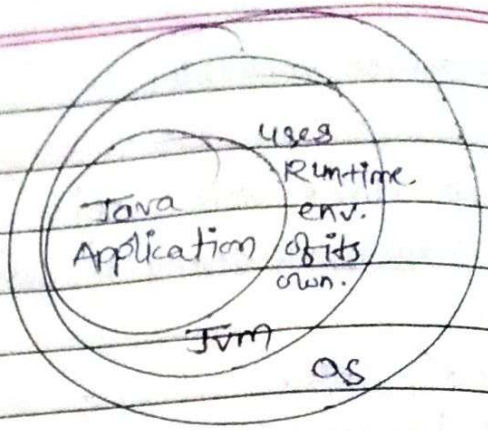
1) simple :⇒ its syntax is simple clean & easy to understand. as it is based on c++ syntax.
- Java has removed many complicated and rarely used features, for eg. explicit pointers, operator overloading etc.
- There is no need to remove unreferenced obj. as there is an Automatic Garbage collection in java.

2) Object oriented :⇒ Everything in java is an object. OOPs programming is a methodology that simplifies s/w development and maintenance by providing some rules. like :
Inheritance, polymorphism, abstraction, encapsulation.

3) Platform independent :⇒ WORA language.
the byte code generated by Java compiler is platform independent and ∴ it can be executed on any platform.
Explain the diagram here.

4) Secured. :⇒ Java is best known for its security. It is secured because :- No explicit pointer.
- Java programs run inside a virtual mch. Sandbox





- classloader :- adds security by seperating the package for the classes of the local filesystem from those that are imported from N/W sources

- Bytecode verifier :- checks the code fragment for illegal code that can violate access rights to obj.

- Security Manager :- determines what resources a class can access such as reading, writing to the local disk.

4) Robust :⇒ Strong.
- uses strong memory management.
- lack of pointers avoids security problems.
- provides automatic garbage collector.
- exception handling & type checking mechanism.

5) Architecture neutral :⇒ No implementation dependant features.
e.g. size of primitive types is fixed.

6) Portable :⇒ carry the java byte code to any platform.

7) Distributed :⇒ - allow to create distributed applications.

8) Multithreaded :- we can write java programs that deal with many tasks at once by defining multiple threads.

9) Dynamic :- Java supports dynamic loading of classes. It means classes are loaded on demand.

Page No.
Date

Java

Q.1 C++ vs

+ platform dependent
1) mainly used for system prog.
3) supports goto statement
4) supports multiple inheritance
5) -*- operator overloading
6) -*- pointers
7) uses compiler only to convert source code into machine code. ∴ platform dependant.
8) supports call by value & call by reference
9) supports structure & union
10) Doesn't have built in thread support. relies on third party. lib.
11) Doesn't support documentation comment.
12) uses header files

- platform independent
- mainly used for app prog. widely used in window-based, web-based, enterprise, mobile app
- doesn't support goto.
- doesn't support multiple inheritance through class but it can be achieved by interfaces in java.
- doesn't support op. overloading
- support pointers internally. But we can't write pointer prog in java.
- Java uses both compiler & interpreter. source code → compiler → bytecode → interpreter executes at runtime generate o/p. ∴ platform independant.
- Supports only call by value. NO call by reference.
- Doesn't support struct union.
- has built in thread support.
- supports (/** */)
- uses import statement.

Page No:
Date:

15) supports virtual keyword to decide whether or NOT to override a fn. - No virtual keyword. All Non-static methods are virtual by default as we can override them.

14) unsigned right shift X - ✓ (>>>)

13) always create new Inheritance tree - uses single inheritance tree as obj. class is parent of all the classes in java.

16) Nearer to H/W - Not so interactive with H/W

17) oo language but single hierarchy is not possible - obj. oriented and single hierarchy is there. everything is obj. in java.