M T W T F S S

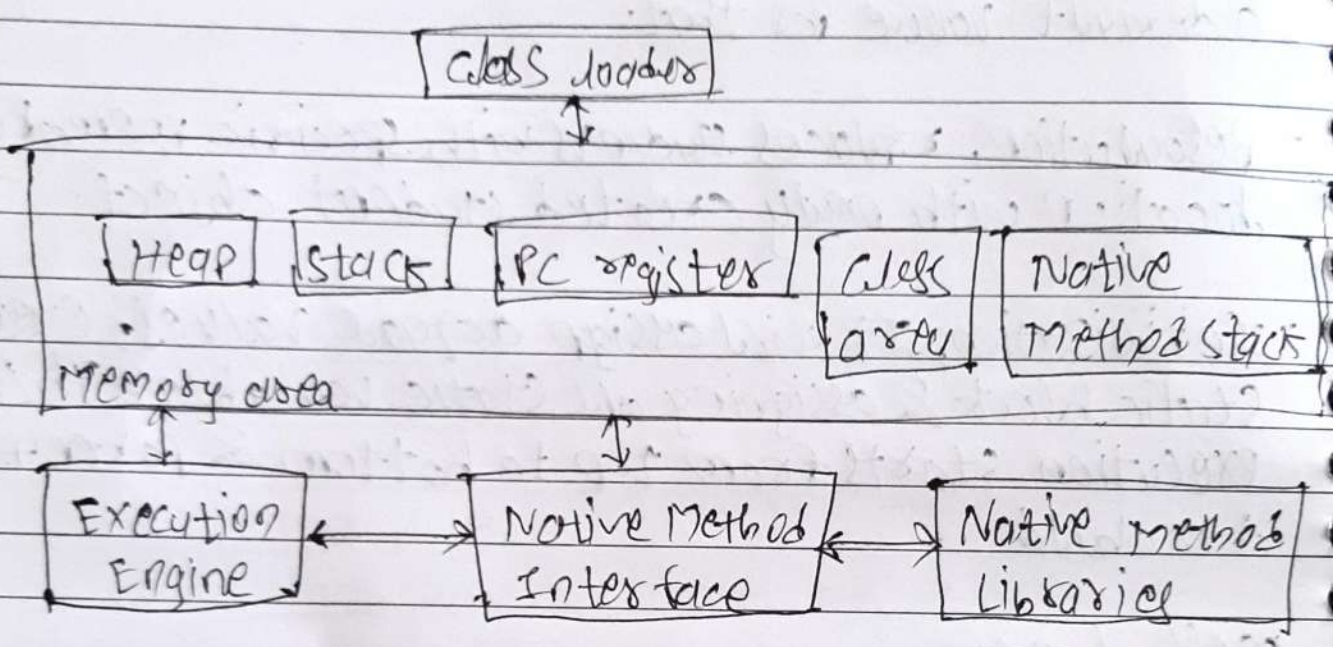
Page No.: YOUVA

Date:

Assignment-2

Q.1) Explain the components of the JVM.

→ There are 5 major components inside JVM.



- Class loader -

- It has 3 major responsibilities. Loading, Linking, Initialization.
- Main responsibility is taking a class & loading into the memory (ram) for execution from secondary memory.
- It is responsible for reading .class file, generates binary data saving it in method area.
- Linking is divided in 3 parts - verification, preparation, Resolution.
- In verification, it will check whether the class file is safe to execute or not.

M T W T F S S

Page No.: YOUVA

Date:

- In preparation, if you use an instance level variable or static variable in your class. it assign a default value for that.

- Relocation, replaces students with specific memory location with newly created student object.

- Initialization, It will assign actual values, execute static block & assigning all static variables. This execution starts from top to bottom & parent to child.

Method area -
- It has 5 parts.
- method area / Heap area -
  - Load all class information JVM has only one method area & heap area.
  - It is also called as thread memory.

- Stacks -
  - keep method information, local variables.
  - A separate runtime stack is created for every thread.
  - All details are stored here until completion of method.

M T W T F S S

Page No.: YOUVA

Date:

- Pc Register -
  - It holds information about next execution.
  - It stores address of currently executing JVM Instruction.
  - Separate Pc register is created for every thread.

- Native method stack -
  - Thread creates this kind of memory & thread is at a whole new level.

- Native Method Area -
  - This is stack that can support native methods that are written in different language.

- Execution Engine -
  - It is responsible ~~for~~ to run particular program.
  - It is divided in 3 parts.
  - It is responsive in executing byte code by converting it into machine code. & also interact with memory area.
  - Interpreter -
    - Responsible to read, interpret & execute java program line by line.

- JIT compiler -
  - Main job is to overcome interpreter disadvantage of slowness during execution.

M T W T F S S

Page No.:	YOUVA
Date:	

- It works only for repeated methods not single method.
- By lowering overall compilation time it increases performance.

- Garbage Collector -
- Used to free up memory by collecting & removing objects from heap area when there is no reference

- Native method libraries -
- Contains libraries required for execution engine.

- Native method interface (JNI) -
- Acts as bridge between Execution Engine & Native method library.
- It also provide non-programming language packages allowing developer to write code in various language.

M T W T F S S

Page No.: YOUVA

Date:

Q.2 JDK JRE JVM

1) Java Development Kit — Java Runtime Environment — Java Virtual Machine.

JDK -

- Java Development Kit, an environment of software development used for developing applications.
- It consists JRE & development tools.
- It has 3 editions.
  - Java SE
  - Java EE
  - Java ME
- It has JVM, loader, interpreter, compiler, Javadoc, jar.

JRE -

- Java Runtime Environment & Java RTE
- Set of tools designed for running softwares
- Implementation JVM & JRE provides runtime environment.
- User need JRE to run any java program

JVM -

- Java virtual machine, provide environment for java applications & code.

- Converts Java byte code to machine language.
- It is capable of running programs written in other languages, compiled in java bytecode.
- It doesn't exists physically.
- It is essential part of JRE.
- It perform following task.
  - Provides Runtime Environment
  - Verified code
  - Loads code
  - Executes code

Q.3) What is role of JVM in java? & How does JVM executes Java code?

→ JVM is crucial in Java as it serves as runtime environment for executing Java code. Its primary role includes interpreting or compiling bytecode, managing memory, handling exceptions, ensuring platform independence, & providing security features such as bytecode verification & sandboxing. Essentially JVM allows Java progs. to run on any device or operating system that has a compitable JVM implementation.

JVM executes Java code in steps→

i) Source code is compiled by Java compiler into byte code.
ii) JVM loads bytecode classes dynamically as needed during execution.
iii) JVM verifies bytecode to ensure it is Java language specification & does not violate security constraints.
iv) JVM interprets bytecode instructions or use JIT compiler to translate bytecode into machine code.
v) JVM manages memory allocation, deallocation, & garbage collection to ensure efficient memory usage.

vi) Exception handing. JVM handles exceptions & runtime errors gracefully, allowing Java prog. to recover from unexpected situations.

vii) JVM implementations does optimizations like, JIT compilation to improve performance.

Q.4) Memory management systems of JVM.

Method area -

- It has 5 parts.
- method area / Heap area -
  - Load all class information JVM has only one method area & heap area.
  - It is also called as thread memory.

- Stacks -
  - Keep method information, local variables.
  - A separate runtime stack is created for every thread.
  - All details are stored here until completion of method.

M T W T F S S

Page No.: YOUVA

Date:

- Pc Register -
  - It holds information about next execution.
  - It stores address of currently executing JVM Instruction.
  - Separate Pc register is created for every thread.

- Native method stack -
  - Thread creates this kind of memory & thread is at a whole new level.

- Native Method Area -
  - This is stack that can support native methods that are written in different language.

Q.5) What are JIT compiler & its role in JVM? what is bytecode & why is it important for Java?

→

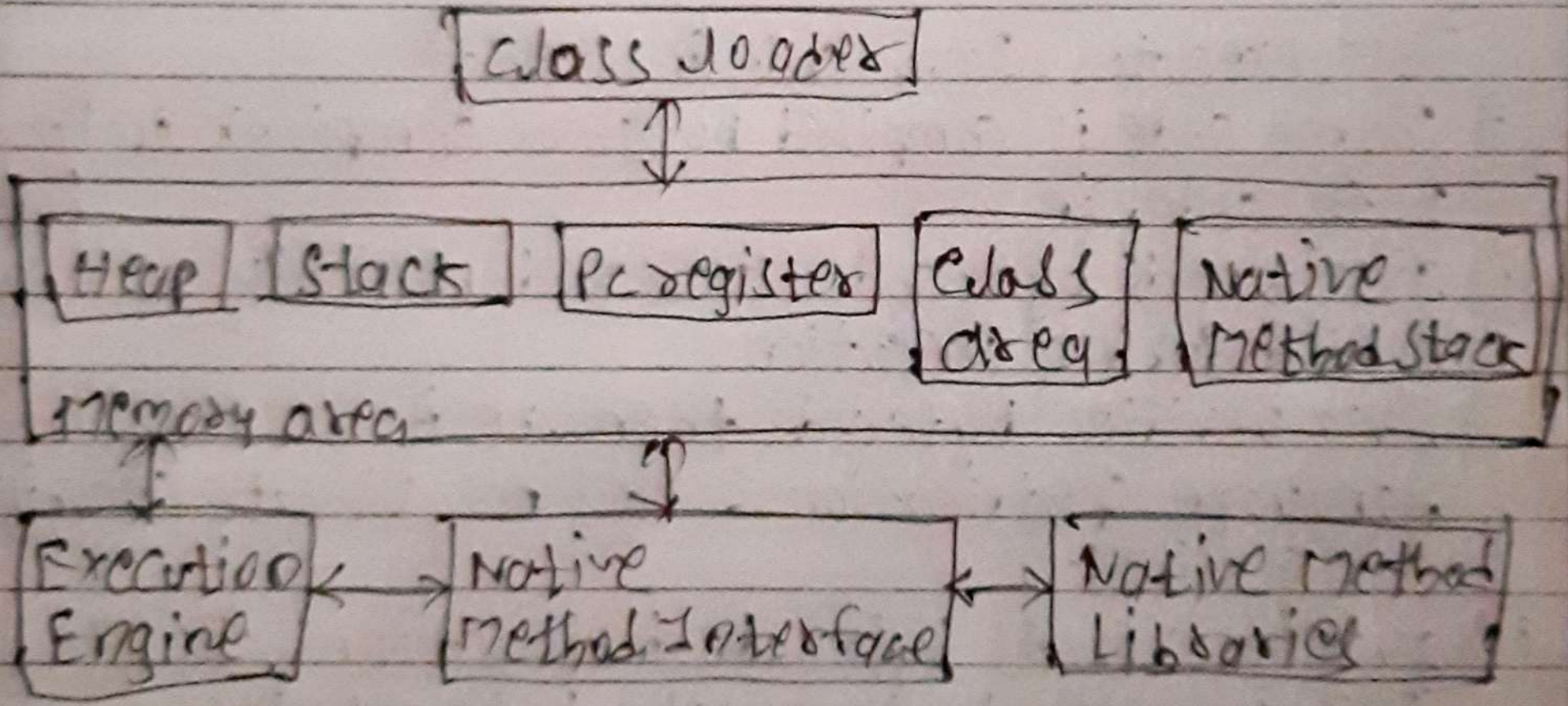
Main job is to overcome interpreter disadvantage of slowness during execution.

- It works only for repeated methods not single method.
- By lowering overall compilation it increases performance.

Java bytecode is instruction set of JVM crucial for executing programs written in Java language & other JVM-compatible languages. Each byte code operation in JVM is represented by single byte, hence name bytecode. is compact form of execution.

Q6) Describe architecture of JVM.

→



Class loader →

- Loads class files into memory. It is responsible for finding & loading class files from file system, network & other sources & then converting them into binary form.

Runtime Data Area –

This is area where data structure used by JVM are allocated. It consists of several components.

- Method area →

It stores class metadata, including method bytecode, fields & method information, constant pool & static variables.

- Heap →

Here objects are created by Java program are allocated.

It's runtime data area shared among all threads & it's ~~the runtime~~ managed by garbage collector for memory management.

Stack →

Each thread in application has its own stack, which stores method invocations & local variables. It's last-in, first-out (LIFO) data structure.

PC Registers →

Each thread has its own Program counter (PC) register, which holds address of currently executing instruction.

Native Method Stack →

It's used for native method execution (methods written in language other than Java).

Execution Engine-
It executes Java bytecode. It has 2 compone
- Interpreter →
It reads by bytecode & execute the codes
- Reading native machine code instructions.
It's portable but relatively slow.

- Just-In-Time (JIT) compiler →
- JIT compiles frequently executed bytecode into native machine code at runtime, optimizing performance compiled code is then cached for future use.

Native method Interface (JNI)-
It enables Java code to call & be called by native applications & libraries written in other languages. It bridge between Java runtime & native code.

Native method Libraries -
These contains native methods that provide functionality beyond JRE.

Q. 7) How Java achieve platform independence through JVM?

Java is platform independent because it is compiled to bytecode that can be run on any device that has JVM. It means we can write code on windows system & then run it on Linux or mac os system. without making any changes in code.

Q8)

Class loader →

- Loads class files into memory. It is responsible for finding & loading class files from file system, network & other sources & then converting them into binary form.

In Garbage collection process, collector scans different parts of heap, looking for objects that are no longer in use. If an object no longer has references to it from elsewhere in application, the collector removes object, freeing up memory in the heap. This process continues till all unused objects are successfully reclaimed.

To ensure garbage collector work efficiently, JVM separates heap into parts & then collectors use mark-and-sweep algorithm to traverse these parts & clear out unused objects.

How it works:-
- There are two phases: mark & then sweep.
- When object is created in heap, it has mark bit that is set to 0 (false).
- During mark phase collector traverse object trees starting at their roots.
- When object root is reachable, mark bit set to 1 (true).
- mark bit for unreachable objects is changed.
- During sweep, collector traverse heap & free memory from all items with mark bit 0 (false).