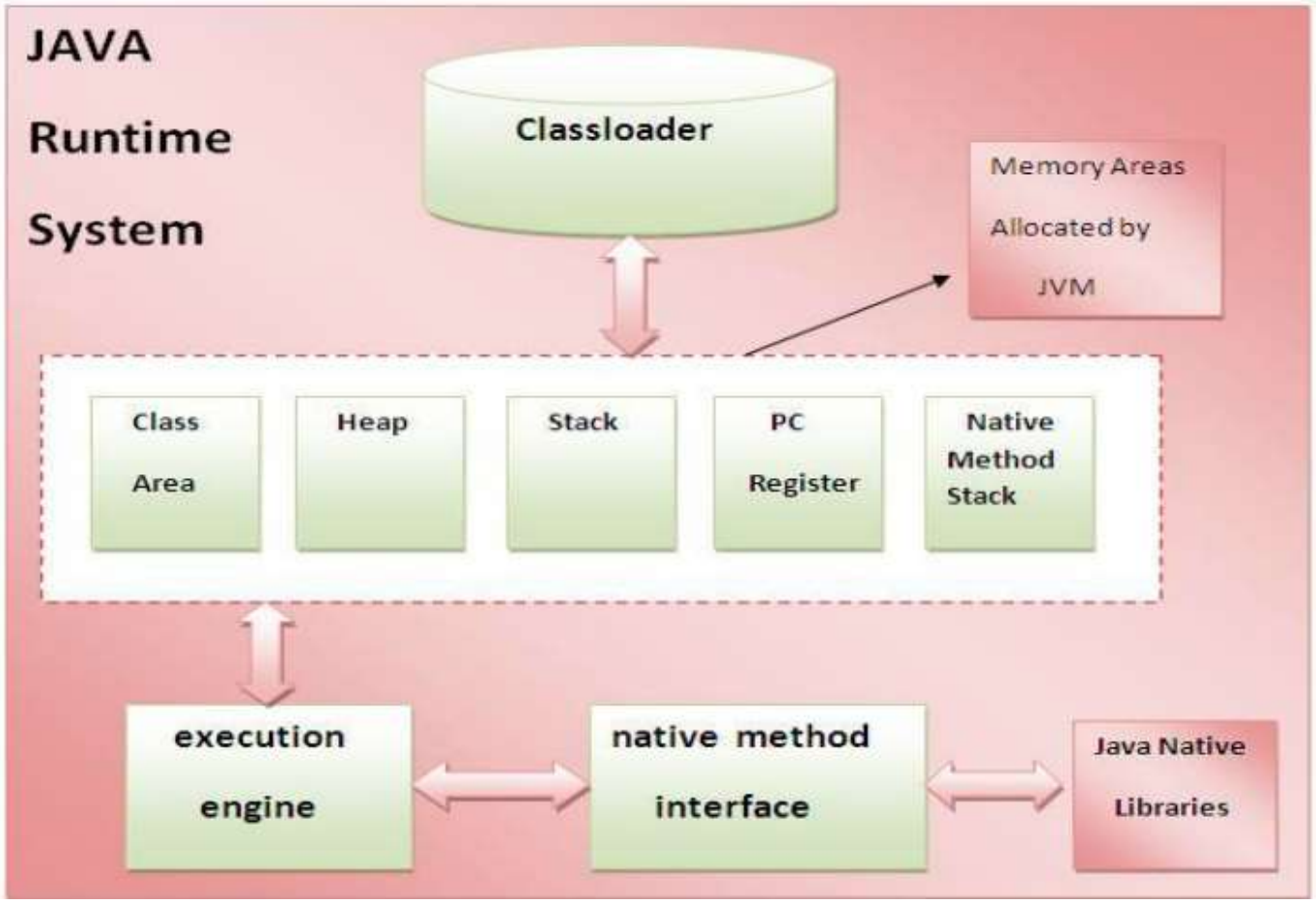
Internal architecture of JVM



Internal architecture of JVM :

Class Loader → Loading, Linking, Initiliazation

Class Area ⇨ It stores **METADATA** (Data about the data) information of the Class.

Heap Area ⇨ Every Java class **object** which we are creating will be stored inside the Heap area.

Inside HEAP area there is one more special memory allocated only for storing String Class Objects which is known as String Constant Pool (SCP).

Stack Area ⇨ It provides temporary memory to store the data and results which were obtained from method executions

PC-Registers ⇨ It maintains the **LOG**-files of Thread executions

Native-Method Stack ⇨ The methods which are not written in Java but those are used in Java are known as **Native methods** and all these native methods are present inside a special memory known as **Native Method Stack**

Execution Engine

- Interpreter ⇨ Interpreter is going to convert byte code instructions into machine understandable format
- Garbage Collector ⇨ The process of destroying all unused or unreferenced objects from the heap area is known as **Garbage Collection**. This process is done by **Garbage Collector**

Class Loader:

The class loader subsystem is used for loading/reading the .class files and saving the bytecode in the JVM method area. It performs three basic activities in following order

- Loading (imports the data)
- Linking (performs verification)
- Initialization (Invokes class variables)

Class / Method Area:

This component holds the class level data of each .class file such as metadata,static variables, the code for the methods etc.

Heap Area:

This component is a part of JVM memory where all the objects and its corresponding instance variables and arrays are stored.

Stack Area:

While running a method, it needs some more memory to store the data and results. This memory is allotted on Java Stacks. [This area plays an important role during the method invocation and returns.]

PC Registers:

This component holds the address of the JVM instruction which is currently executing. Each thread in Java has its own PC register to hold the address of the currently executing instruction

Native Method Stacks:

This component is written in a different language and holds the native method information.

Execution Engine:

This component executes the bytecode which is assigned to the runtime data areas and has two major sub-components i.e.:

- Interpreter: This component reads the bytecode instructions and executes them in a sequential manner.
- Garbage Collection: This component is a part of execution engine which frees up the memory by collecting and removing the unreferenced objects

Native Method Interface:

This allows the Java code to call or be called by the libraries and the native applications (i.e. the programs specific to the hardware and the OS of a system).

Native Method Libraries:

This component is a collection of native C, C++ libraries which are required by the execution engine.

Internal architecture of JVM:

Class loader is present above the JVM

Class loader

Loading---- loads .class file

Whenever we are giving input as java generated .classfile name to JVM, this .class file will be first loaded into the class loader .

Linking

Linking----link your java files with any other dependencies

Class loader is going to connect your Java file with other dependencies.

Your Java file is connected with how many Java files

Your Java file is linked with how many Java files

Your Java file is having how many dependencies all those will be taken care of by the class loader.

Initialization

Initialization-----initialize static variables

Initialization means assigning some values

Declaration means creating a variable

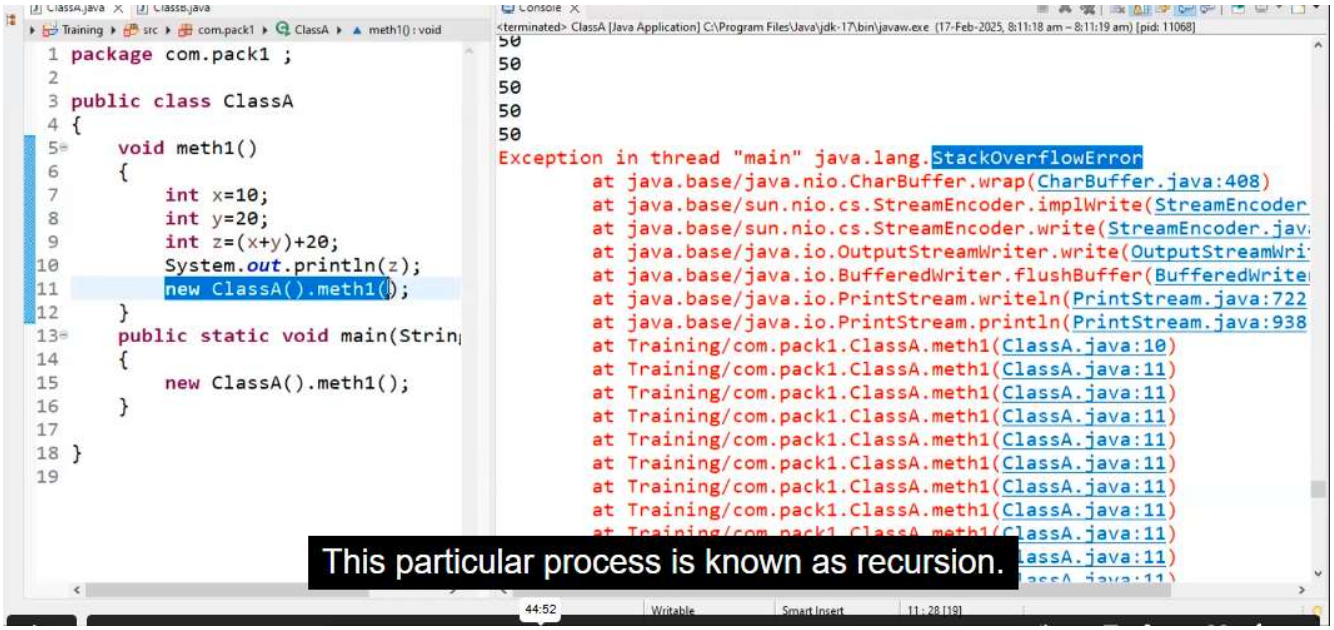
In java we have three types of variables

1. Instance variable
2. Static variable
3. Local variable

For instance, variable and local variable memory will be assigned whenever we are running the Java program

For static variable memory will be assigned not at the time of running, before running only, means at the time of class loading

Static variable will be initialized at the time of class loading by the class loader



In line number 11 we are calling a method by its own

Recursion

It is a process of calling a method from itself

Internal architecture of JVM

5 memory areas are present inside the JVM

1.CLASS AREA

It stores **metadata** (data about the data) information of the class.

What do you mean by data about the data

For example, I have taken a class.

In a class how many variables are there, how many methods are there, how many objects we have created, how many dependencies we are having, how many import statement we have written, how many packages we have imported all these information about our class will be stored inside the class area.

2.HEAP AREA

Every Java class object which we are creating will be stored inside the heap area.

Inside the heap area there is one more special memory allocated only for storing (string class object) which is known as string constant pool (SCP).

3.STACK AREA

It provides temporary memory to store the data and results which were obtained from method executions

Example

Int x=10;

Int y=20;

Int z=(x+y)+20; (here (x+y) is a temporary

For example, we have 3 files on top of the table

1. Electricity bill file
2. gas bill file
3. pay credit card bill

If we pay the electricity bill file, then the order is in

1. Gas will be fine
2. Pay credit card bill

If we remove the first task, then the second task will come to the first place.

The difference between heap area and stack area

HEAP area Provides long lasting memory

STACK area provides temporary memory

4.PC-REGISTERS (PROGRAM COUNTER-REGISTER)

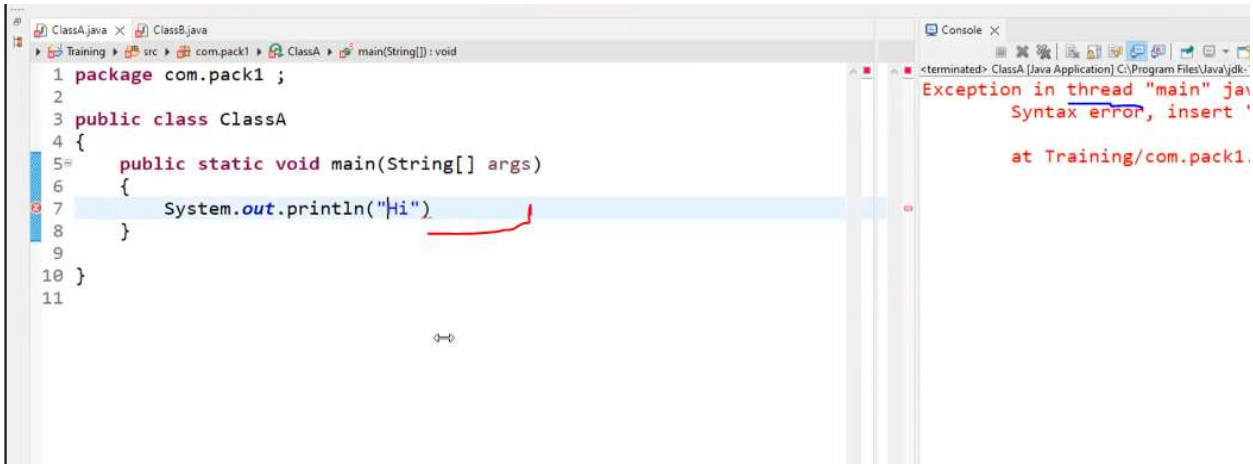
It maintains the log file of thread executions.

Thread

We have this concept like multithreading

Multithreading used to perform multitasking

Multitasking is performing multiple tasks simultaneously at the same time by a single processor in order to optimize the utilization of CPU



Example

split screen on mobile phone

Consider every method has one thread at a particular point of time, which thread is executing, running state, sleeping state, blocked state, waiting state, terminated state complete that log information will be stored inside the log files and those files are presented inside PC region

5.NATIVE-METHOD STACK

The methods which are not written in Java but those are used in Java are known as native methods and all these native methods are present inside a special memory known as native method stack.

EXECUTION ENGINE

INTERPRETER

The interpreter is going to convert bytecode instruction into machine understandable format.

GARBAGE COLLECTOR

The process of destroying all unused or unreferenced objects from the heap area is known as garbage collectors.