

programación visual

MICHEL DZAY MAYRA FERNANDA

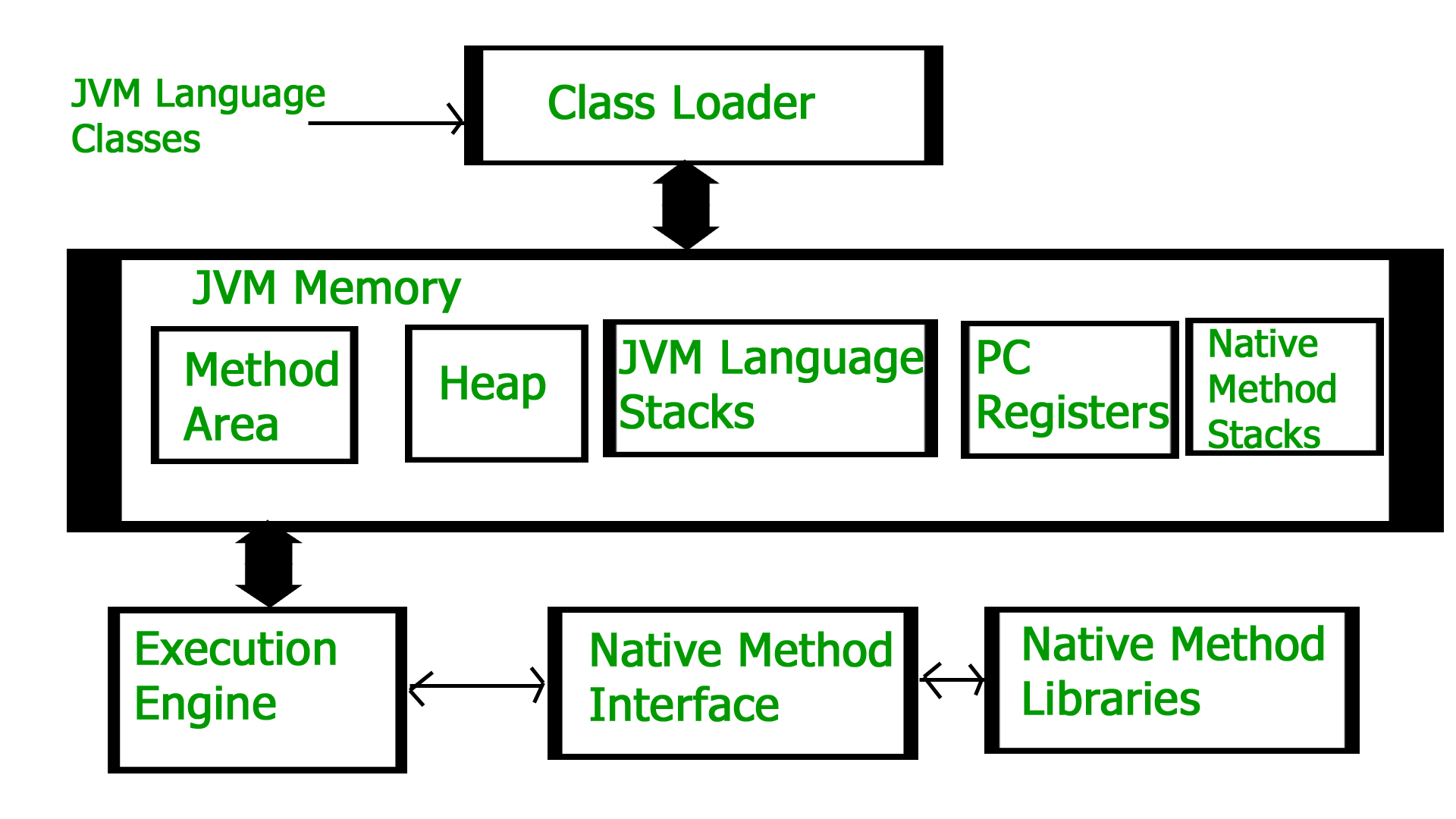




java virtual machine

**Java Virtual Machine (JVM):**

Actúa como un motor en tiempo de ejecución para ejecutar aplicaciones java. En resumen, un lenguaje de alto nivel se compila y en vez de generar lenguaje maquina se generan Bytecodes que el JVM traduce a instrucciones nativas a la plataforma correspondiente con esto conseguimos que un programa java se escriba una vez y se pueda ejecutar en cualquier SO sin necesidad de cambiarlo o ajuste es decir el WORA (Write Once Run Anywhere) existen varios pasos para llevar este proceso los cuales se mencionara a continuación.



**Class Loader Subsystem**

Se encarga de tres actividades:

* Cargando
* Enlace
* Inicialización

**Cargando:**

El cargador de clases lee el archivo.class, genera los datos binarios correspondientes y los guarda en el área de método. Después de cargar el archivo JVM crea un objeto de tipo Class para representar este archivo en la memoria del montón.

**Enlace:**

Realiza la verificación, preparación y opcionalmente la resolución.

* Verificación: garantiza la corrección del archivo.class
* Preparación: JVM asigna memoria para las variables de clase e inicializa la memoria a los valores predeterminados.
* Resolución: remplaza las referencias simbólicas de tipo con referencias directas.

**Inicialización:**

En esta fase todas las variables estáticas se asignan con sus valores definidos en el código y el bloque estático. Se ejecuta de arriba-abajo en una clase o de padre a hijo.

Existen tres cargadores:

* Bootstrap: carga las clases básicas de la API
* Clases de extensión: carga las clases presentes en los directorios de extensiones.
* Sistema/aplicación: carga clases desde la ruta de la clase de la aplicación.

**Área de método de memoria JVM:**

se almacena toda la información de nivel de clase.

**Área de almacenamiento dinámico:**

se almacena la información de todos los objetos.

**Área de pila:**

JVM crea para cada hijo una pila en tiempo de ejecución que se almacena aquí, después de que termine un subproceso será destruida por JVM.

**Registros de PC:**

Almacena la dirección de la instrucción actual de un hilo.

**Método de pila nativa:**

Para cada hilo se crea una pila nativa separada que almacena información del método nativo.

**Motor de ejecución:**

Ejecuta el .class(bytecode). Lee el código de byte línea por línea, usa datos de información presentes en varias áreas de memoria y ejecuta las instrucciones.

* Interprete: interpreta el bytecode línea por línea y luego ejecuta.
* Compilador Just In Time (JIT): se usa para aumentar la eficiencia del interprete, compila todo el código de butes y lo cambia a código nativo.
* Recolector de basura (GC) destruye objetos sin referencia.

**Interfaz nativa de java (JIN):**

Es una interfaz que interactúa con las bibliotecas de métodos nativos y las proporciona para su ejecución.

**Conclusión:**

JVM es el que realmente llama al método principal presente en un código java, tiene muchos procesos que llevan a la compilación del programa en cualquier sistema operativo sin la necesidad de ejecutarse varias veces y tener que hacer cambios o arreglos para realizarse dicha acción sin mencionar que es más eficiente su ejecución gracias al JIT que acelera el proceso del interprete y el recolector de basura autónomo.