Una vez desarrollado un programa en Java y corregidos todos sus errores de compilación (sintácticos), se procede a su ejecución para comprobar su correcto funcionamiento.

Durante la ejecución, el código compilado (*bytecode*) se carga en la memoria para que el procesador pueda ejecutar cada una de sus instrucciones. Asimismo, los datos del programa, definidos principalmente por variables y objetos, también deben almacenarse en la memoria RAM.

Sin embargo, estos datos no se guardan de forma indiscriminada en un mismo lugar. La Máquina Virtual de Java (JVM) divide la memoria principal principalmente en dos zonas claramente definidas: **el Stack (Pila)** y **el Heap (Montículo)**.

1. Memoria Stack (Pila)

En el **Stack** se almacenan las variables locales de tipos primitivos (como `int`, `char`, `float`, `boolean`, etc.) e instrucciones de ejecución de los métodos.

Los datos primitivos almacenan directamente su valor en la memoria. Por ejemplo:

Java

```
int x = 3;
char letra = 'v';
float numero = 3.8f; // Nota: en Java se usa el punto decimal y el sufijo 'f' para float
```

Además de los tipos primitivos, el Stack también almacena **las referencias** (direcciones de memoria) que apuntan a los objetos guardados en el Heap.

2. Memoria Heap (Montículo)

En el **Heap** se guardan las estructuras de datos dinámicas creadas en tiempo de ejecución, es decir, **los objetos y sus variables de instancia**.

Por ejemplo, supongamos que tenemos la clase `Coche` y creamos una instancia de la misma llamada `coche_rojo`:

Java

```
Coche coche_rojo = new Coche();
```

- El objeto `coche` con todos sus datos e información se creará y guardará en la memoria **Heap**.
- La variable `coche_rojo` (que actúa como una referencia o puntero con la dirección de memoria del objeto) se ubicará en el **Stack**.

Esquema de representación de la memoria

Dado un programa en Java con la siguiente declaración de variables primitivas y un objeto de la clase `Coche`:

```
int x = 10;
char letra = 'a';
float numero = 3.8f;
Coche coche_rojo = new Coche();
```

La distribución de memoria se estructura de la siguiente manera: