UTN Frre

# VARIABLES EN JAVA

# ¿QUÉ ES UNA VARIABLE?

Una variable es un contenedor que almacena datos que pueden cambiar durante la ejecución de un programa. Cada variable tiene un nombre, un tipo de datos, y un valor.

# TIPOS DE VARIABLES EN JAVA

1. VARIABES PRIMITIVAS
    a. Enteros: byte, short, int, long
    b. Punto Flotante: float, double
    c. Carácter: char
    d. Booleano: boolean
2. Variables de Referencia:
    a. Objetos y arreglos: cualquier clase o tipo definido por el usuario

# DECLARACIÓN Y ASIGNACIÓN DE VARIABLES

Para declarar una variable en Java, necesitas especificar el tipo de datos seguido del nombre de la variable.

EJEMPLO: tipo nombreVariable;

# ASIGNACIÓN DE VALORES

Puedes asignar un valor a una variable en el momento de su declaración o posteriormente.

```
// Declaración
 int numero;
 numero = 10; // Asignación
```

```
// Declaración y Asignación Simultánea
int edad = 25;
```

# EJEMPLOS DE VARIABLES PRIMITIVAS

## 1. Enteros

int edad = 30;
long distancia = 123456789L; // 'L' indica que es un long literal

## 2. Punto Flotante

float temperatura = 36.6F; // 'F' indica que es un float literal
double pi = 3.14159;

## 3. Carácter

char inicial = 'A';

## 4. Booleano

boolean esMayor = true;

# EJEMPLOS COMPLETO

**Vamos a crear un programa completo que utuliza diferentes tipos de variables**

# REGLAS DE NOMBRES DE VARIABLES EN JAVA

## Buenas Prácticas en Nombres de Variables en Java

Los nombres de variables en Java son esenciales para la legibilidad y mantenibilidad del código. Seguir buenas prácticas y convenciones de nomenclatura ayuda a que el código sea más comprensible y profesional. A continuación, se detallan las reglas y buenas prácticas para nombrar variables en Java, con ejemplos.

# REGLAS BÁSICAS PARA NOMBRES DE VARIABLES

1. Debe comenzar con una letra, un símbolo de dólar ($) o un guion bajo (_):
   - Correcto: nombre, _nombre, $nombre
   - Incorrecto: 1nombre, -nombre
2. No puede contener espacios ni caracteres especiales:
   - Correcto: nombreCliente
   - Incorrecto: nombre cliente, nombre-cliente
3. No puede ser una palabra reservada de Java:
   - Correcto: int numero
   - Incorrecto: int int
4. Distingue entre mayúsculas y minúsculas:
   - nombre, Nombre, NOMBRE son variables diferentes.

# BUENAS PRACTICAS

1. Usar Camel Case para Nombres de Variables:
   - Utilizar camel case para nombres de variables, comenzando con minúscula y cada nueva palabra con mayúscula.
   - Ejemplo: nombreCompleto, numeroDeTelefono
2. Ser Descriptivo y Claro:
   - Los nombres de las variables deben describir claramente su propósito.
   - Ejemplo: edadPersona, precioProducto
3. Prefijos y Sufijos Claros (si es necesario):
   - Usar prefijos como is o has para variables booleanas.
   - Ejemplo: isActivo, hasSaldo

# BUENAS PRACTICAS

4. Evitar Nombres de Variables de una Sola Letra:
   - A menos que se usen en bucles pequeños o contextos muy específicos.
   - Ejemplo: int i en un bucle for, pero preferir indice fuera de ese contexto.
5. No Abusar de Abreviaturas:
   - Usar nombres completos siempre que sea posible para mayor claridad.
   - Ejemplo: totalPiezas en lugar de totPzs
6. Usar Nombres Significativos Incluso en Variables Temporales:
   - Asegurarse de que los nombres de las variables temporales también sean descriptivos.
   - Ejemplo: sumaParcial en lugar de temp

# EJEMPLOS DE BUENAS Y MALAS PRÁCTICAS

Buenas Prácticas

```
public class EjemploBuenasPracticas {
        public static void main(String[] args) {
            int edadPersona = 25; // Claro y descriptivo
           boolean isActivo = true; // Prefijo claro para booleanos
           double salarioMensual = 1500.50; // Uso de camel case y descriptivo
        }
 }
```

# EJEMPLOS DE BUENAS Y MALAS PRÁCTICAS

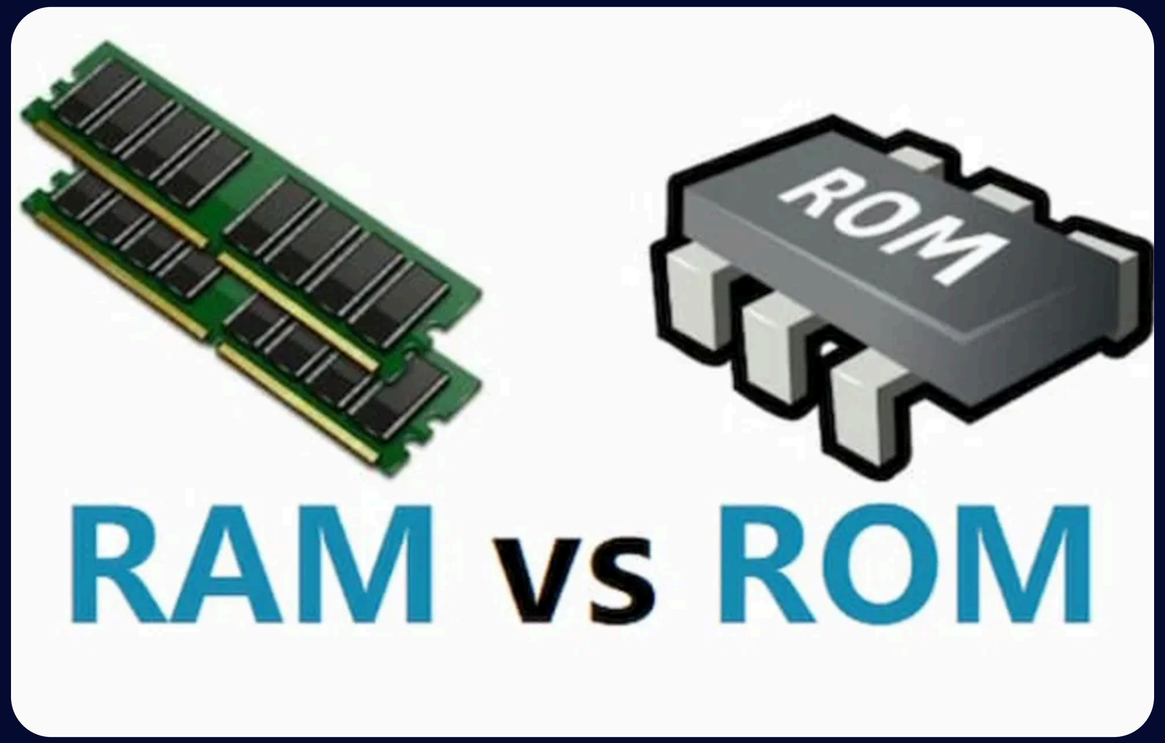
Malas Prácticas

```
public class EjemploMalasPracticas {
        public static void main(String[] args) {
                int e = 25; // No descriptivo
                boolean b = true; // No claro
                 double sal = 1500.50; // Abreviatura no clara
        }
 }
```

UTN Frre

# MEMORIA RAM Y ROM: RELEVANCIA EN EL MANEJO DE VARIABLES EN JAVA

Cuando hablamos de programación y manejo de variables en Java, es útil entender la diferencia entre los tipos de memoria de una computadora, específicamente la RAM (Random Access Memory) y la ROM (Read-Only Memory). A continuación, te explico estos conceptos y su relevancia en el contexto de Java.



# ÁREAS DE MEMORIA EN LA JVM DENTRO DE LA MEMORIA RAM

La JVM gestiona la memoria en diferentes áreas de la RAM, cada una con su propósito específico:

1. **Stack:**
   a. Es el área de memoria donde se almacenan las variables locales, así como los valores de variables primitivas definidas dentro de un método.
   b. Cada vez que se llama a un método, se crea un nuevo marco (frame) en el stack que contiene las variables locales y los datos necesarios para la ejecución del método.

**2.Heap:**

**a.** Es el área donde se almacenan los objetos y sus datos asociados.

b. La memoria en el heap es gestionada automáticamente por el recolector de basura (Garbage Collector).

# FUNCIONAMIENTO DEL GARBAGE COLLECTOR

El Garbage Collector (GC) es una parte crucial de la JVM que administra la memoria heap. Su trabajo principal es:

- Liberar memoria: Recolectar objetos que ya no son accesibles y liberar la memoria que ocupan.
- Optimizar el uso de memoria: Garantizar que la memoria se utilice de manera eficiente y minimizar la fragmentación.

El proceso de recolección de basura ocurre automáticamente y se activa según las necesidades de la aplicación y las políticas de la JVM.

# RELEVANCIA DEL SISTEMA HEXADECIMAL EN JAVA

Cuando se almacenan variables en la memoria RAM, la representación hexadecimal es útil para:

- Direcciones de Memoria: Los sistemas operativos y las herramientas de depuración suelen usar hexadecimal para mostrar las direcciones de memoria.
- Depuración y Análisis: Permite una visualización compacta y legible de los valores binarios.

# EJEMPLO DE USO DE HEXADECIMAL PARA REPRESENTAR DIRECCIONES DE MEMORIA

Aunque no podemos directamente acceder y manipular direcciones de memoria en Java, podemos mostrar los valores donde están almacenadas las variables y cómo se visualizan.

- En resumen: Sistemas de Numeración
  - Binario: El más básico para computadoras, usa solo 0 y 1.
  - Octal: Menos común hoy, pero era usado en computadoras más antiguas.
  - Decimal: El sistema que usamos a diario.
  - Hexadecimal: Muy útil en informática para representar datos binarios de manera compacta y legible.

# EXPLICACIÓN BREVE DE CLASES Y OBJETOS EN JAVA

¿Qué es una Clase? Una clase en Java es una plantilla o molde que define las propiedades (atributos) y comportamientos (métodos) que los objetos creados a partir de la clase pueden tener. Es un concepto central en la programación orientada a objetos.

- Atributos: Características o propiedades de la clase.
- Métodos: Acciones o comportamientos que la clase puede realizar.

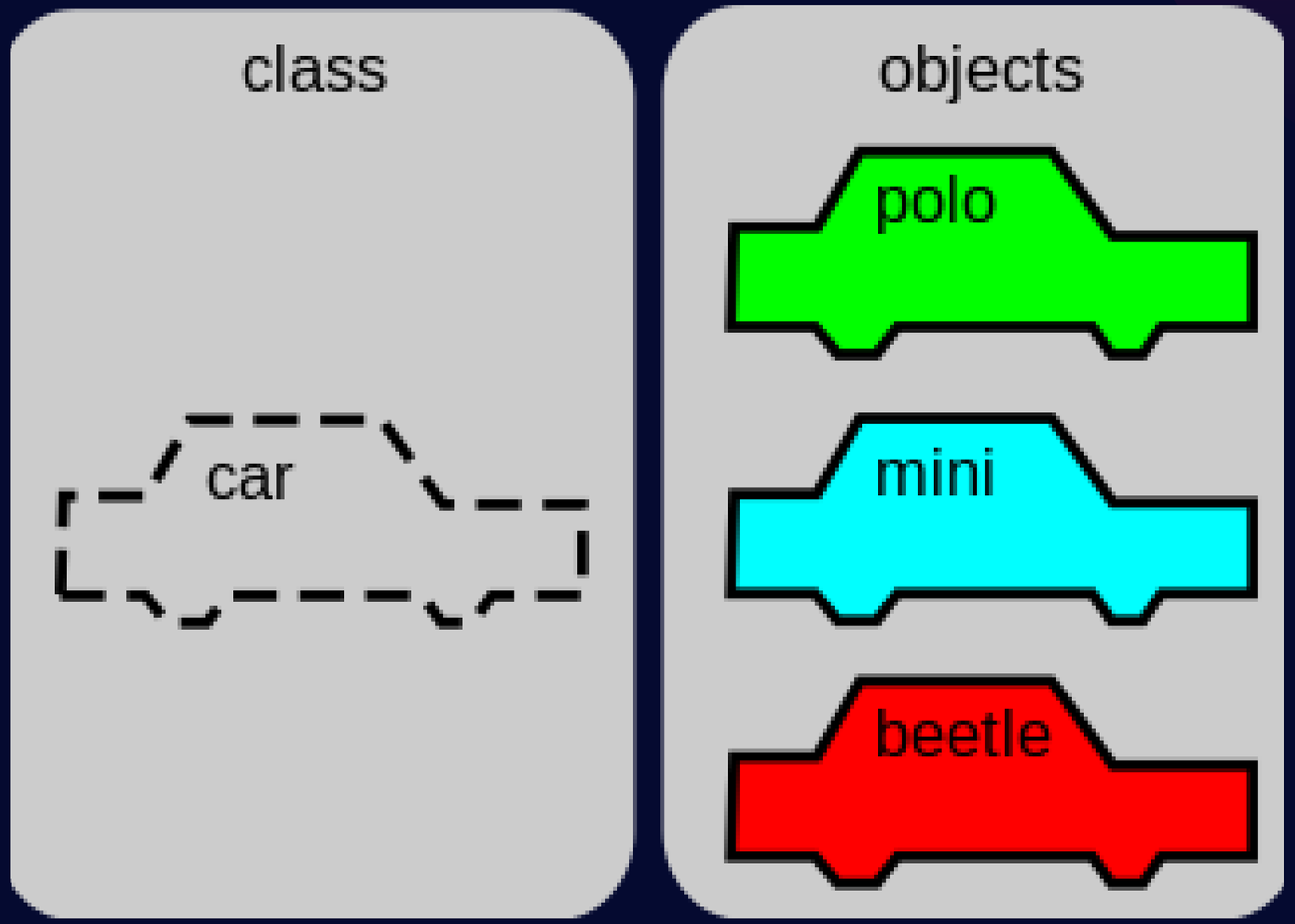
# ¿QUÉ ES UN OBJETO?

Un objeto es una instancia de una clase. Cuando creamos un objeto, estamos utilizando la plantilla definida por la clase para crear una entidad concreta con valores específicos.

- Instancia: Una copia concreta de la clase.
- Estado: Valores actuales de los atributos del objeto.

## Analogía Sencilla

Clase: Imagina un plano de construcción de una casa. El plano describe cómo se verá la casa y qué características tendrá. • Objeto: Es la casa real construida a partir del plano. Cada casa construida a partir del mismo plano es una instancia del plano (la clase).


class
car
objects
polo
mini
beetle

```java
public class Main {
    public static void main(String[] args) {
        // Crear un nuevo objeto de la clase Object
        Object obj1 = new Object(); // Crea un nuevo objeto de tipo Object
    }
}
```

- Clase: Object es una plantilla básica que Java proporciona para crear objetos.
- Objeto: obj1 es una instancia de esta plantilla, una entidad concreta que se crea en la memoria.