UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Programación Orientada en Objetos

NOMBRE:

Gabriel Ariel Velasco Molina

TITULO:

Viaje a través de la Evolución de los Lenguajes de Programación

CARRERA:

Tecnología De La Informática

SEMESTRE:

Primero

Introducción

La evolución de los lenguajes de programación ha marcado profundamente la forma en que solucionamos problemas computacionales. Desde lenguajes cercanos al hardware hasta paradigmas avanzados como la Programación Orientada a Objetos (POO), cada lenguaje ha dejado huella en los que lo sucedieron.
En este trabajo se exploran tres lenguajes representativos de distintas etapas: **C**, **Java** y **Python**.

1. Lenguaje C

Origen

- Creado por Dennis Ritchie entre 1969 y 1972 en los laboratorios Bell.
- Diseñado para desarrollar el sistema operativo UNIX.
- Derivado de lenguajes previos como B y BCPL.

Hitos Clave

- **1978**: Publicación del libro *The C Programming Language* ("K&R C"), que estandarizó su sintaxis.
- **1989/1990**: Surgimiento del estándar ANSI C (C89/C90).
- Base para lenguajes modernos como C++, Java, C#, Objective-C, Go.

Influencia Actual

- Fundamental en el desarrollo de sistemas operativos, drivers y software de alto rendimiento.
- Introdujo conceptos esenciales:
    - Punteros
    - Manejo explícito de memoria
    - Estructuras de control modernas (if, while, for)

Ejemplo de Código Representativo

```c
#include <stdio.h>
int main() {
    printf("Hola, mundo en C\n");
    return 0;
}
```

Gráfico Descriptivo

2. Java

Origen

- Creado por James Gosling en Sun Microsystems, en 1995.
- Originalmente pensado para dispositivos electrónicos inteligentes.

Hitos Clave

- Introducción del concepto **"Write Once, Run Anywhere"** gracias a la JVM.
- Inclusión nativa de POO como paradigma central.
- **2006:** Se vuelve open-source con el proyecto OpenJDK.
- **2010:** Oracle adquiere Sun Microsystems y toma control de Java.

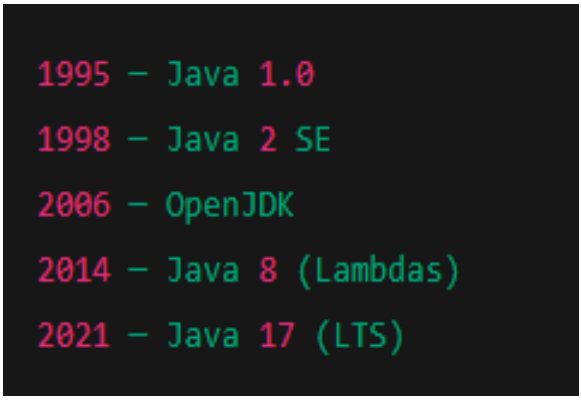
Influencia Actual

- Dominante en:
  - Aplicaciones empresariales
  - Android
  - Servidores web
  - Banca y finanzas

Ejemplo de Código Representativo

```java
public class HolaMundo {
    public static void main(String[] args) {
        System.out.println("Hola, mundo en Java!");
    }
}
```

Gráfico Descriptivo



3. Python

Origen

- Creado por Guido van Rossum en 1991.
- Surgió como sucesor del lenguaje ABC.
- Filosofía: código legible y limpio ("There should be one — and preferably only one — obvious way to do it.").

Hitos Clave

- **2000:** Python 2.0 introduce manejo automático de memoria.
- **2008:** Python 3 rescribe la base del lenguaje.
- Popularización masiva gracias a IA, ciencia de datos y automatización.

Influencia Actual

- Considerado uno de los lenguajes más importantes del siglo XXI.
- Popular en:
  - Machine Learning
  - Data Science
  - Automatización
  - Scripts educativos

Ejemplo de Código Representativo

```
print("Hola, mundo en Python")
```

Gráfico Descriptivo

```
1991 – Python 1.0
2000 – Python 2.0
2008 – Python 3.0
2020 – Muerte de Python 2
```

Comparación y Contraste

Característica	C	Java	Python
Paradigma	Imperativo, estructurado	POO puro	Multiparadigma
Nivel	Bajo/medio	Medio/alto	Alto
Gestión de memoria	Manual	Automática (GC)	Automática
Velocidad	Muy alta	Alta	Media
Sintaxis	Compleja	Verbosa pero estructurada	Simple y legible
Influencia en POO	Base de C++ → inspiró Java	Ejemplo clásico de POO	POO flexible y accesible

Relación con la POO

- **C**: No es orientado a objetos, pero su filosofía influyó directamente en **C++**, que sí implementó POO y sirvió como modelo para Java.
- **Java**: Construido alrededor del paradigma orientado a objetos. Gran responsable de su masificación.
- **Python**: Integra la POO de forma simple y flexible, lo que facilita su enseñanza.

Reflexión Personal

Entender cómo han evolucionado los lenguajes de programación me ayuda a ver el "por qué" de muchas cosas que usamos hoy. C me deja claro lo importante que es conocer cómo funciona realmente el hardware. Java me muestra lo útil que es la estructura y la disciplina de la POO, y Python me recuerda que la simplicidad también puede ser muy poderosa.

Al ver esta evolución, entiendo que la Programación Orientada a Objetos no nació por moda, sino para resolver problemas reales: organizar mejor el código, hacerlo más fácil de mantener y permitir que los proyectos crecieran sin volverse un caos. Por eso ahora valoro más conceptos como la herencia, el encapsulamiento o la modularidad.

En general, conocer esta historia me hace programar con más conciencia, entendiendo que todas las herramientas que usamos hoy existen gracias a muchos años de aprendizaje y mejoras.