Introducción al Lenguaje Java

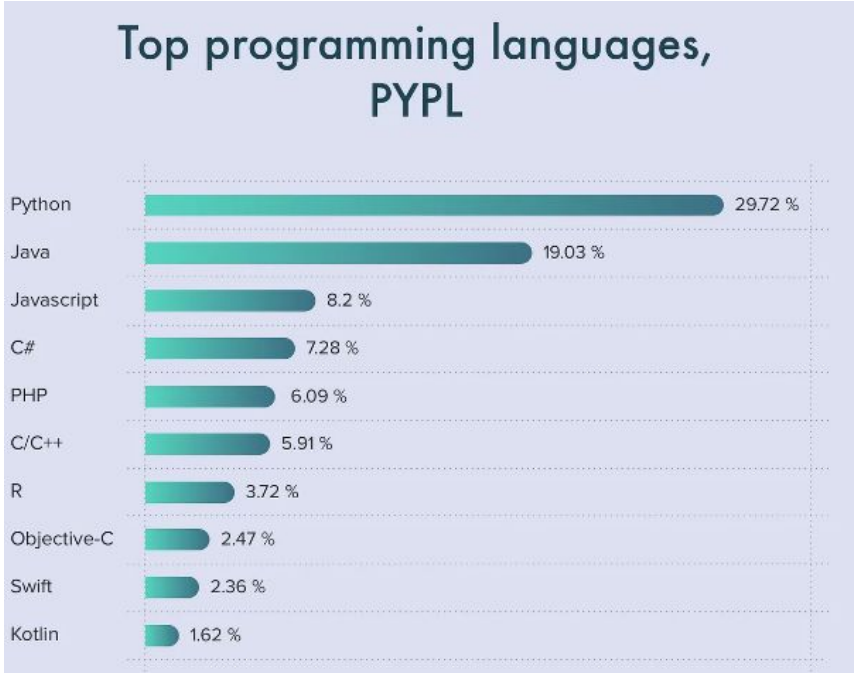
Programación Orientada a Objetos

San Salvador de Jujuy

UNJu - Facultad de Ingeniería
Ing. José Zapana

¿Qué es Java?

- Java es un lenguaje de programación de propósito general, orientado a objetos y de alto nivel, desarrollado por Sun Microsystems en 1995.
- Es ampliamente utilizado para el desarrollo de aplicaciones empresariales, móviles, web y sistemas embebidos.



Introducción a Java

- Java, conocido por su portabilidad y escalabilidad, sigue siendo una opción popular en el desarrollo empresarial. Su robusta infraestructura y la compatibilidad multiplataforma le aseguran vigencia a lo largo del tiempo.

- Aunque su sintaxis puede parecer un tanto extensa, su comunidad activa y la confiabilidad en grandes proyectos empresariales son sus puntos fuertes.

Ventajas y Desventajas

Ventajas

- 1. Portabilidad: 'Write Once, Run Anywhere' (WORA).
- 2. Seguridad: Fuerte gestión de memoria y verificación de bytecode.
- 3. Orientación a Objetos: Facilita la reutilización de código.
- 4. Amplia comunidad y soporte: Gran cantidad de bibliotecas y frameworks.
- 5. Multithreading: Manejo de múltiples tareas simultáneamente.

Desventajas

- 1. Rendimiento: Más lento que lenguajes compilados como C++.
- 2. Consumo de memoria: Requiere más memoria que otros lenguajes.
- 3. Verbosidad: Código más largo en comparación con otros lenguajes modernos.
- 4. Gestión de recursos: Requiere atención para evitar fugas de memoria.

Mejoras en las últimas versiones

- **Java 8:** Introducción de lambdas, Streams API, y la API de fechas.
- **Java 9:** Sistema de módulos (Project Jigsaw) y JShell.
- **Java 10:** Inferencia de tipos con 'var'.
- **Java 11:** Nuevas mejoras en la API HTTP y soporte para ZGC (Garbage Collector).
- **Java 17:** Características de previsualización como patrones de coincidencia y sellado de clases.
- **Java 21 y 22:** Mejoras varias en lenguaje, sus API y rendimiento.
- **Java 23, 24:** Mejoras en rendimiento y seguridad

Tipos de datos primitivos en java 24

Tipo	Categoría	Ejemplo de uso
byte	Entero pequeño	byte edad = 30;
short	Entero medio	short año = 2025;
int	Entero estándar	int salario = 50000;
long	Entero largo	long poblacion = 7800000000L;
float	Decimal simple	float altura = 1.75f;
double	Decimal doble	double pi = 3.14159;
char	Carácter Unicode	char letra = 'J';
boolean	Lógico	boolean activo = true;

Constantes

- Declaración
  - `public static final` ***<type> <name> = <value>;***
- Ejemplos

```
public static final int DAYS_IN_WEEK = 7;
public static final double INTEREST_RATE = 3.5;
public static final int SSN = 658234569;
```

Tipos de datos Objetos en java

- Son instancias de clases
- Tienen métodos, propiedades y comportamientos
- Se almacenan en el heap y su referencia se guarda en variables
- Menor rendimiento que los primitivos en operaciones intensivas.

Tipo de objeto	Descripción	Ejemplo
`String`	Cadena de texto	`String nombre = "José";`
`Integer`, `Double`, `Boolean`	Clases envolventes de tipos primitivos	`Integer edad = 30;`
`Array`	Colección de elementos del mismo tipo	`int[] notas = {8, 9, 10};`
`List`, `Map`, `Set`	Estructuras de datos dinámicas (colecciones)	`List<String> nombres = new A`
Clases personalizadas	Definidas por el programador	`Persona p = new Persona();`

Clases Envolventes en Java (Wrapper Classes)

- Las clases envolventes permiten tratar los tipos primitivos como objetos.
- Son parte del paquete java.lang y se usan cuando se necesita un objeto en lugar de un valor primitivo.
- Permiten un autoboxing y unboxing automático

```
int x = 10;
Integer y = x;          // Autoboxing
int z = y;              // Unboxing
```

Variables estáticas *static*

- Variables estáticas
  - También se les conoce como *class variables*
  - Variables que se asocian a clase no a objeto
  - Variable común a todos los objetos (variables compartidas entre todos los objetos de la clase)
    - Se definen como variable de clase con la palabra clave static
  - Ejemplo : Identificador de cuenta de CuentaBanco. Un número que identifique únicamente a dueño de cuenta.

Ejemplo: variables estáticas

```
public class {
  private String nombre;
   private int balance;
   private int Id;
   private static int proxIdDisponible = 1;

   /** Constructor, establece nombre dueño y balance
   de la cuenta */

    public CuentaBanco(String nombre, int
        balance){ this.nombre = nombre;
        this.balance = balance;
        this.Id =
        proxIdDisponible;
        proxIdDisponible++;
   }
```

Métodos estáticos

- Un método estático pertenece a la clase, no a una instancia. Se puede invocar sin crear un objeto. Son ideales para operaciones utilitarias, cálculos, y lógica que no depende del estado interno de un objeto.
- Características clave
  - Se acceden directamente desde la clase (Clase.metodo()).
  - No pueden usar this ni acceder a atributos no estáticos.
  - Son ideales para funciones reutilizables y puras.
  - Se ejecutan sin necesidad de instanciar la clase.

Clase	Método estático típico
Math	Math.sqrt(25)
Integer	Integer.parseInt("123")
Arrays	Arrays.sort(array)
Clase propia	MiClase.utilidad()

Métodos estáticos - Ejemplo

```java
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class FechaUtil {
    private static final DateTimeFormatter FORMATO_FECHA =
        DateTimeFormatter.ofPattern("dd/MM/yyyy");

    public static String obtenerFechaActual() {
        return LocalDate.now().format(FORMATO_FECHA);
    }
}
```

```java
System.out.println(FechaUtil.obtenerFechaActual());
```