



UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

GRUPO 5: Go VS Java

Integrantes:

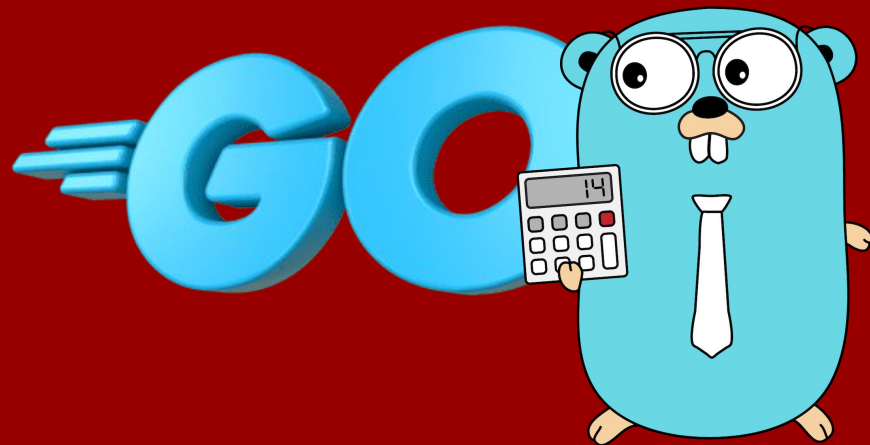
Masia Moreno Luis Nicolás

Calizaya Mamani Santiago Francisco

Poblete Agustín

Sayago Pablo Ezequiel

Historia de Go



Go, a menudo referido como "Golang";

Fue creado por un equipo de ingenieros de Google y anunciado públicamente en 2009.

El objetivo principal era abordar algunas de las deficiencias y complejidades percibidas en otros lenguajes de programación.

Características

- Simplicidad y legibilidad.
- Eficiencia: Compilación estática y recolección de basura eficiente contribuyen a su desempeño.
- Compatibilidad.
- ● Concurrencia: Utiliza un modelo GoRoutines.
- Facilidad de uso: Permite detectar errores en la sintaxis durante la compilación y no durante la ejecución.

Historia de Java



Java se creó en 1991 con el nombre "OAK" por un grupo llamado green team, posteriormente se cambió el nombre por problemas legales y quedó finalmente con la denominación actual JAVA.

- El objetivo de JAVA era crear un lenguaje de programación parecido a C++ en estructura y sintaxis, fuertemente orientado a objetos, pero con una máquina virtual propia, usando el lema “Write Once, Run Anywhere”

BNF de Go

```
<Program> ::= <PackageClause> <ImportDecl> <TopLevelDecl>*
<PackageClause> ::= "package" <PackageName>
<ImportDecl> ::= "import" ( <ImportSpec> | "(" { <ImportSpec> } ")" )
<ImportSpec> ::= [ "." | <PackageName> ] <ImportPath> <TopLevelDecl> ::=
<Declaration> | <FunctionDecl> | <MethodDecl>
<Declaration> ::= <ConstDecl> | <TypeDecl> | <VarDecl> <ConstDecl> ::= "const" (
<ConstSpec> | "(" { <ConstSpec> } ")" )
<TypeDecl> ::= "type" ( <TypeSpec> | "(" { <TypeSpec> } ")" )
<VarDecl> ::= "var" ( <VarSpec> | "(" { <VarSpec> } ")" ) <FunctionDecl> ::= "func"
<FunctionName> <Signature> <FunctionBody>
<Signature> ::= <Parameters> [ <Result> ]
<Parameters> ::= "(" [ <ParameterList> [ "..." ] ] ")"
<Result> ::= <Parameters> | <Type> <ParameterList> ::= <ParameterDecl> |
<ParameterList> "," <ParameterDecl>
<ParameterDecl> ::= [ <IdentifierList> ] [ "..." ] <Type>
```

BNF de Java

```
<compilationUnit> ::= <packageDeclaration>?
<importDeclaration>* <typeDeclaration>*
<packageDeclaration> ::= "package" <packageName> ";"

<importDeclaration> ::= "import" <qualifiedName> ";"
<classDeclaration> ::= ( "public" | "abstract" | "final" )? "class"
<className>
    ( "extends" <superClassName> )? ( "implements"
<interfaceName> ( "," <interfaceName> )* )?
    "{" <classBodyDeclaration>* "}"

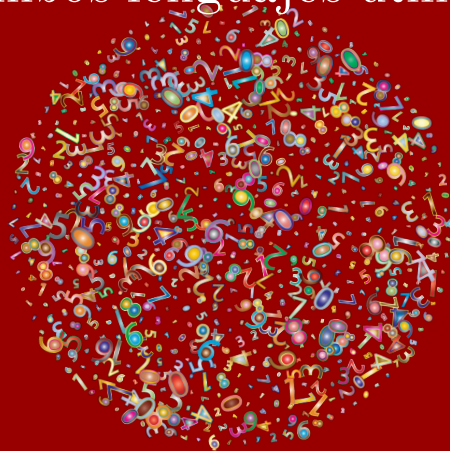
<classBodyDeclaration> ::= <fieldDeclaration> |
<methodDeclaration> | <constructorDeclaration>

<fieldDeclaration> ::= ( "public" | "protected" | "private" | "static" )?
<type> <variableDeclarators> ";"
<methodDeclaration> ::= ( "public" | "protected" | "private" | "static"
| "abstract" | "final" )?
    ( <type> | "void" ) <methodName> <parameters>
<methodBody>
<constructorDeclaration> ::= <modifiers>? <constructorName>
```

Comparación de Rendimiento

Se evaluará el tiempo de ordenamiento en ambos lenguajes utilizando

— un arreglo de 100.000 números aleatorios.





```
package main

import (
    "fmt"
    "math/rand"
    "sort"
    "time"
)

func main() {
    // Crear un arreglo de 100000 números aleatorios
    numeros := make([]int, 100000)
    for i := range numeros {
        numeros[i] = rand.Intn(1000000)
    }

    inicio := time.Now()
    sort.Ints(numeros)
    fin := time.Since(inicio)

    fmt.Printf("El tiempo de ordenamiento en Go fue %s\n", fin)
}
```

```
import java.util.Arrays;
import java.util.Random;

public class SortExample {
    public static void main(String[] args) {
        int[] numeros = new int[100000];
        Random random = new Random();

        // Crear un arreglo de 100000 números aleatorios
        for (int i = 0; i < numeros.length; i++) {
            numeros[i] = random.nextInt(1000000);
        }

        long inicio = System.currentTimeMillis();
        Arrays.sort(numeros);
        long fin = System.currentTimeMillis() - inicio;

        System.out.printf("El tiempo de ordenamiento en Java fue %d ms\n", fin);
    }
}
```


Resultados

Tiempo de Ejecución: Aunque la cantidad de números es grande, los lenguajes modernos como Go y Java están diseñados para manejar tareas como esta de manera eficiente.

A continuación mostraremos los resultados obtenidos en Go y en Java a la hora de ordenar un vector de 100,000 números enteros aleatorios de forma ascendente.

Resultados en Go

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
- - - - -

[Running] go run "c:\Users\Luis\Documents\Facultad\AGA
El tiempo de ordenamiento en Go fue 7.2097ms

[Done] exited with code=0 in 2.954 seconds

[Running] go run "c:\Users\Luis\Documents\Facultad\AGA
El tiempo de ordenamiento en Go fue 6.1115ms

[Done] exited with code=0 in 1.33 seconds

[Running] go run "c:\Users\Luis\Documents\Facultad\AGA
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
- - - - -

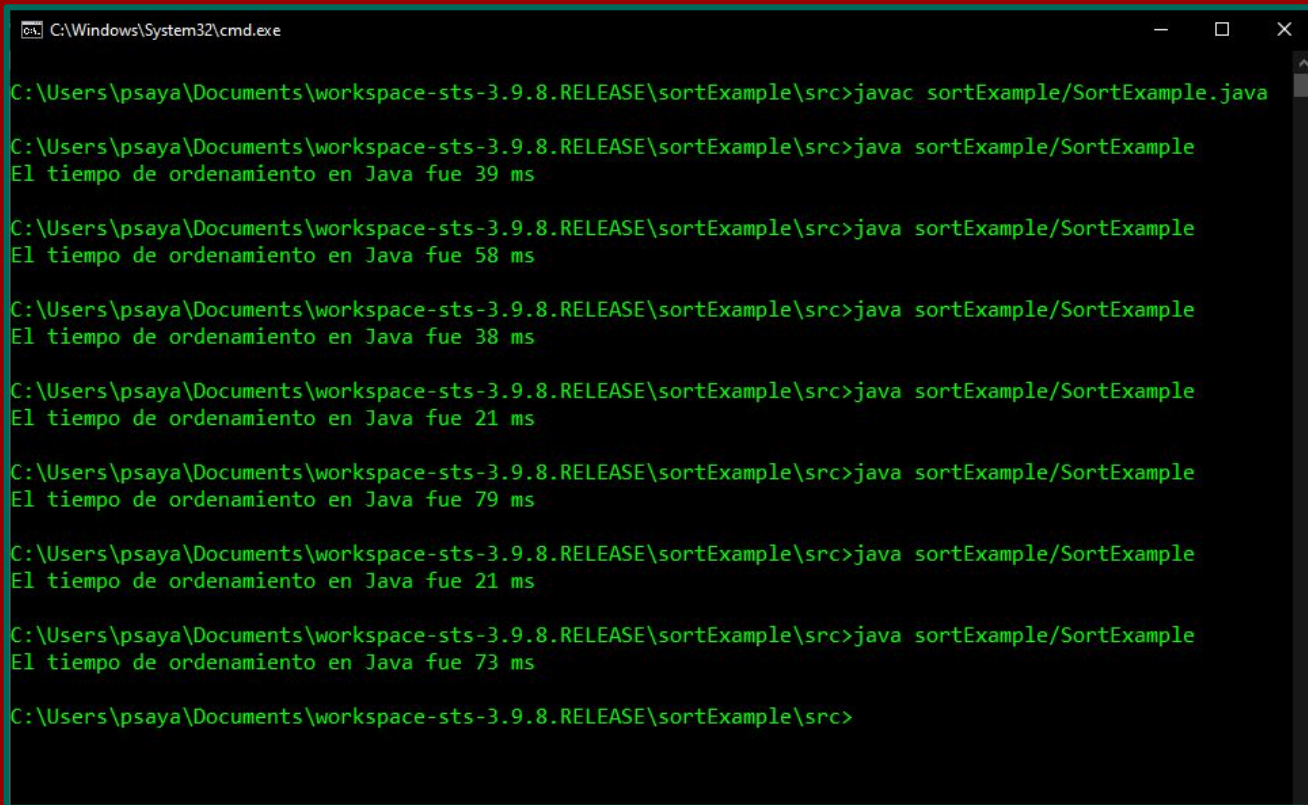
[Running] go run "c:\Users\Luis\Documents\Facultad\AGA
El tiempo de ordenamiento en Go fue 5.3777ms

[Done] exited with code=0 in 0.816 seconds

[Running] go run "c:\Users\Luis\Documents\Facultad\AGA
El tiempo de ordenamiento en Go fue 6.1707ms

[Done] exited with code=0 in 0.881 seconds
```

Resultados en Java



```
C:\Windows\System32\cmd.exe

C:\Users\psaya\Documents\workspace-sts-3.9.8.RELEASE\sortExample\src>javac sortExample/SortExample.java

C:\Users\psaya\Documents\workspace-sts-3.9.8.RELEASE\sortExample\src>java sortExample/SortExample
El tiempo de ordenamiento en Java fue 39 ms

C:\Users\psaya\Documents\workspace-sts-3.9.8.RELEASE\sortExample\src>java sortExample/SortExample
El tiempo de ordenamiento en Java fue 58 ms

C:\Users\psaya\Documents\workspace-sts-3.9.8.RELEASE\sortExample\src>java sortExample/SortExample
El tiempo de ordenamiento en Java fue 38 ms

C:\Users\psaya\Documents\workspace-sts-3.9.8.RELEASE\sortExample\src>java sortExample/SortExample
El tiempo de ordenamiento en Java fue 21 ms

C:\Users\psaya\Documents\workspace-sts-3.9.8.RELEASE\sortExample\src>java sortExample/SortExample
El tiempo de ordenamiento en Java fue 79 ms

C:\Users\psaya\Documents\workspace-sts-3.9.8.RELEASE\sortExample\src>java sortExample/SortExample
El tiempo de ordenamiento en Java fue 21 ms

C:\Users\psaya\Documents\workspace-sts-3.9.8.RELEASE\sortExample\src>java sortExample/SortExample
El tiempo de ordenamiento en Java fue 73 ms

C:\Users\psaya\Documents\workspace-sts-3.9.8.RELEASE\sortExample\src>
```

Conclusiones

A partir de los resultados obtenidos podemos observar que el lenguaje Go es más eficiente que el lenguaje Java en el contexto de las funciones de ordenamiento.

— Esto se debe a que Go presenta una compilación directa a código máquina. Esto significa que el código generado está optimizado para el hardware específico donde se ejecuta, lo que generalmente resulta en un rendimiento más alto. En cambio Java utiliza una máquina virtual (JVM) para ejecutar el código, la máquina virtual realiza verificaciones en tiempo de ejecución haciendo incrementar el trabajo del sistema.

Preguntas

1. ¿Cuál era el objetivo del grupo de los desarrolladores de Green Team con Java?
2. Nombrar 3 características del lenguaje Go.
3. A partir de los resultados obtenidos en la comparación de rendimiento, ¿Cuál es el lenguaje más eficiente a la hora de utilizar funciones de ordenamiento?
4. ¿Por qué se dieron estos resultados?