CMPS445 Project: Golang

جـامعة بيروت العربية
BEIRUT ARAB UNIVERSITY

University: Beirut Arab University

Course: Concepts of Programming Languages

Instructor: Dr. Lama Affara

Lab TA: Ziad Ghosn, Salma Ghali

Submission Date: Thursday 4-12-2025

Group Members

1. **Raydan Aridi (ID: 202604102) – rra410@student.bau.edu.lb**
2. **Waseem Bou Hamdan (ID: 202403373) – wmb256@student.bau.edu.lb**
3. **Hasan Farhat (ID: 202401813) – hmf481@student.bau.edu.lb**

Project Resources

- **Project Zip Folder: CMPS445 Project Golang.zip**
- **Project Documentation: README.md**
- **Source Code Repository: https://github.com/RaydanAridiCS/aegis**
- **Presentation Report: CMPS445 Project: Golang Project.pdf**

1. What Is Golang?

Go (Golang) is a fast, simple, and modern programming language created by Google. It is widely used for high-performance backend systems, specifically designed for:

- **Cloud Computing: Docker, Kubernetes, and Terraform are built with Go.**
- **Backend APIs: Microservices and high-performance APIs.**
- **Concurrency: Designed to handle massive concurrent tasks reliably.**
- **Distributed Systems: Networking tools and DevOps CLI tools.**

2. Golang vs. Java: Overview

Feature	Golang (Go)	Java
Origin	**Developed by Google (2009).**	**Developed by Sun Microsystems (1995).**
Philosophy	**Simplicity, high concurrency, fast execution.**	**Portability, stability, enterprise usage.**
Compilation	**Compiles directly into native machine code.**	**Runs on the Java Virtual Machine (JVM).**
Primary Use	**DevOps, cloud-native, distributed systems.**	**Enterprise systems, Android, banking.**
Key Features	**Goroutines, channels, fast compile times.**	**Rich OOP model, massive frameworks (Spring).**

3. Syntax & Structure Comparison

3.1 Golang Syntax

Go uses a minimalist approach. It has no classes (uses structs) and handles concurrency natively.

```
package main
import "fmt"

func main() {
    fmt.Println("Hello, Go!")
}
```

Key Constructs:

- **Packages: Organizational units (package main).**
- **Functions: Primary building block (func).**
- **No Classes: Uses Structs + Methods.**
- **Concurrency: Built-in go keyword.**

3.2 Java Syntax

Java is strictly Object-Oriented. Everything must belong to a class.

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, Java!");
    }
}
```

Key Constructs:

- **Classes: Everything is inside a class.**
- **Objects: Instantiated from classes.**
- **Interfaces: Explicit implementation using implements.**
- **Concurrency: Threads & Executor frameworks.**

4. Concurrency Model

Feature	Golang (Goroutines)	Java (Threads)
Mechanism	Uses Goroutines instead of OS threads.	Uses Thread class or Executors.
Weight	Lightweight (~2 KB memory each).	Heavy (1–2 MB memory per thread).
Communication	Communicates via Channels.	Shared memory (requires synchronization).
Scalability	Can run millions of concurrent tasks easily.	Heavy on system resources at scale.

5. Error Handling

Golang: Uses explicit error value checks.

```
result, err := doSomething()
if err != nil {
    return err
}
```

- ✔ Simple and predictable.
- ✘ Can be repetitive/verbose.

Java: Uses Try/Catch blocks.

```
try {
    doSomething();
} catch (Exception e) {
    e.printStackTrace();
}
```

- ✔ Centralized and clean.
- ✘ Can hide logic flow if overused.

6. Ecosystem & Use Cases

Golang

- **Frameworks: Gin (HTTP), Echo, Fiber, gRPC.**
- **Best For: Microservices, Cloud-native tools, High-performance APIs, Low-memory environments.**

Java

- **Frameworks: Spring Boot, Hibernate, Jakarta EE, Maven/Gradle.**
- **Best For: Enterprise backends, Banking/Insurance systems, Android apps, Big Data (Hadoop).**

7. Performance Benchmark: Prime Number Search

To compare raw performance, we implemented a prime number search algorithm in both languages using For and While loops.

7.1 Java Implementation

```
public class PrimeComparison {
    public static boolean isPrime(int n) {
        if (n < 2) return false;
        for (int i = 2; i * i <= n; i++) {
            if (n % i == 0) return false;
        }
        return true;
    }

    public static void main(String[] args) {
        int[] numbers = {3, 4, 5, 11, 15, 17, 19, 23, 24, 29, 31, 37, 41, 43, 47};

        // --- FOR LOOP ---
        long startFor = System.nanoTime();
        for (int i = 0; i < numbers.length; i++) {
            isPrime(numbers[i]);
        }
        long timeFor = System.nanoTime() - startFor;
```

```
        // --- WHILE LOOP ---
        long startWhile = System.nanoTime();
        int i = 0;
        while (i < numbers.length) {
            isPrime(numbers[i]);
            i++;
        }
        long timeWhile = System.nanoTime() - startWhile;

        System.out.println("FOR loop time:   " + timeFor + " ns");
        System.out.println("WHILE loop time: " + timeWhile + " ns");
    }
}
```

Java Sample Output:

FOR loop time: 18400 ns

WHILE loop time: 20300 ns

7.2 Golang Implementation

```
package main

import (
    "fmt"
    "time"
)

func isPrime(n int) bool {
    if n < 2 {
        return false
    }
    for i := 2; i*i <= n; i++ {
        if n%i == 0 {
            return false
        }
    }
    return true
}
```

```
func main() {
    numbers := []int{3, 4, 5, 11, 15, 17, 19, 23, 24, 29, 31, 37, 41, 43, 47}

    // --- FOR LOOP ---
    startFor := time.Now()
    for i := 0; i < len(numbers); i++ {
        isPrime(numbers[i])
    }
    timeFor := time.Since(startFor)

    // --- WHILE LOOP (Go uses 'for' as while) ---
    startWhile := time.Now()
    i := 0
    for i < len(numbers) {
        isPrime(numbers[i])
        i++
    }
    timeWhile := time.Since(startWhile)

    fmt.Println("FOR loop time:   ", timeFor)
    fmt.Println("WHILE loop time: ", timeWhile)
}
```

Golang Sample Output:

FOR loop time: 63ns

WHILE loop time: 68ns

8. Aegis Project Implementation

Aegis is the specific application developed by our team for this project. It is a command-line interface (CLI) tool written in Go that provides directory-level encryption.

8.1 Core Capabilities

Aegis demonstrates Go's strengths in system programming by implementing the following features:

- **Seal: Encrypts an entire directory using secure, password-derived keys.**
- **Unseal: Decrypts a protected directory back to its original state.**
- **Watch: Monitors a directory for file changes and automatically re-encrypts new data.**

8.2 How to Use the Code

The tool is designed to be simple and intuitive. Below are the standard commands used to operate Aegis:

1. Encrypting a Folder (Seal) To protect a directory, run the seal command followed by the folder path:

```
./aegis seal /path/to/my_folder
```

2. Decrypting a Folder (Unseal) To access the files again, use the unseal command:

```
./aegis unseal /path/to/my_folder
```

3. Automatic Protection (Watch) To keep a folder constantly protected while working, use the watch command:

```
./aegis watch  /path/to/my_folder
```

8.3 Code Structure

The codebase uses the Cobra library for CLI management and is organized as follows:

- **root.go: The entry point that initializes the application.**
- **seal.go: Contains the logic for the encryption command.**
- **unseal.go: Contains the logic for the decryption command.**
- **watch.go: Handles the file system monitoring logic.**

9. Final Verdict & Conclusion

Summary

- **Golang Strengths: Fast compilation, small binary size, amazing concurrency, and simpler syntax.**
- **Java Strengths: Massive ecosystem, mature libraries, and strong Object-Oriented design.**

Experimental Conclusion

Based on our benchmark test searching for prime numbers:

1. **The For loop in Golang executed significantly faster than the equivalent in Java (63ns vs 18400ns).**
2. **The While loop structure followed a similar pattern, with Go outperforming Java.**

Result: For raw computational tasks and iteration speed in this specific scenario, Golang demonstrated superior performance compared to Java.