Go 언어 개요 및 환경 설정

1. Go 언어와 Java 간단 비교

Go는 구글이 개발한 언어로, 단순함, 효율성, 그리고 동시성 프로그래밍 지원에 중점을 둠. Java가 전통적인 객체지향 패러다임에 깊이 뿌리 내린 반면, Go는 현대적인 개발 환경의 요구사항을 반영하여 더 간결하고 빠른 컴파일 속도를 자랑함.

항목	Go 언어	Java
개발사	Google	Oracle
출시년도	2009	1995
문법	간결, 직관적	비교적 복잡, 객체지향
빌드 속도	매우 빠름	상대적으로 느림
동시성	고루틴(Goroutine) 내장	Thread, 병렬 처리 지원
메모리 관리	가비지 컬렉션(GC) 자동	가비지 컬렉션(GC) 자동
타입 시스템	정적 타입, 컴파일 언어	정적 타입, 컴파일 언어
실행 환경	바이너리 직접 실행	JVM 필요
크로스 플랫폼	지원(Windows, macOS, Linux)	지원(Windows, macOS, Linux)
패키지 관리	내장 모듈/패키지 시스템	외부 빌드 도구(Maven, Gradle)

Go와 Java의 주요 특징 비교 (표)

구분	Go 언어 특징	Java 특징
문법	간결한 문법	객체지향 문법
컴파일/실행	빠른 컴파일, 단일 바이너리	JVM 기반, 상대적으로 느림
동시성	고루틴(Goroutine) 내장	스레드(Thread)
실행 환경	바이너리 직접 실행	JVM 필요
생태계	내장 패키지, 단순 구조	다양한 프레임워크

---

2. 실습 환경 준비

2.1. Go 설치

1. **공식 사이트**에서 자신의 OS에 맞는 설치 파일 다운로드 후 설치.
2. 설치 시 `C:\Go` (Windows) 또는 `/usr/local/go` (macOS/Linux) 경로에 설치되며, 환경 변수가 자동으로 설정됨.
3. 터미널(명령 프롬프트)에서 버전 확인 명령어로 설치 확인: (`version`에 `--`없음)

```
$ go version
```

2.2. 개발 도구 (IDE/에디터)

이 포스팅에서는 Visual Studio Code를 기본으로 사용한다.

필수 익스텐션

- **Go (필수 확장):** Google에서 공식 지원. 코드 자동 완성, 정의로 이동, 디버깅, 테스트 등 Go 개발에 필요한 거의 모든 기능을 제공.
- **Code Runner:** `Jun Han`이 개발한 멀티 랭귀지 실행도구. Java, C/C++, JS, PHP, Python, Go등 다양한 언어를 단축키 `Ctrl + Alt + N`으로 실행시킬 수 있다.

> 원래 Go언어를 실행시키기 위해서는 약간의 설정 과정이 필요하지만 아직까지는 기초 문법 수준이므로 Code Runner만으로 충분하다.

---

3. 첫 번째 Go 프로그램

3.1. 소스 코드 작성

워크스페이스에 `01-hello`와 같은 실습 폴더를 만들고, 그 안에 `main.go` 파일을 생성.

실습 예제: `/01-hello/main.go`

```
/** 1. 패키지 선언 */
// 이 파일이 실행 가능한 프로그램임을 나타낸다.
// Java의 `public class Main { ... }`과 유사하게 프로그램의 진입점을 정의한다.
package main

/** 2. 외부 패키지 임포트 */
// `fmt` 패키지는 문자열 포맷팅, 입출력 등을 다루는 Go의 표준 라이브러리이다.
// Java의 `import java.util.*;`와 유사하게 필요한 기능을 가져온다.
import "fmt"

/** 3. 프로그램 시작점 */
// `main` 함수는 Go 프로그램이 시작될 때 가장 먼저 실행되는 함수
// Java의 `public static void main(String[] args)`와 동일한 역할을 한다.
func main() {
    /** 4. 콘솔에 문자열 출력 */
    // `fmt` 패키지의 `Println` 함수를 호출하여 괄호 안의 내용을 콘솔에 출력한다.
    // Java의 `System.out.println()`과 유사
    fmt.Println("Hello, Go!")
}
```

코드 분석:

1. `package main`: 이 파일이 **실행 가능한 프로그램**임을 나타냄. `main` 패키지의 `main` 함수가 프로그램의 시작점이 됨.
2. `import "fmt"`: 문자열 포맷팅, 입출력 등을 다루는 표준 라이브러리인 `fmt` 패키지를 가져옴.
3. `func main()`: 프로그램이 시작될 때 가장 먼저 실행되는 함수.
4. `fmt.Println(...)`: `fmt` 패키지의 `Println` 함수를 호출하여 괄호 안의 내용을 콘솔에 출력.

3.2. 실행 및 디버깅 (Code Runner 사용)

소스코드에서 단축키 `Ctrl + Alt + N`을 누른다.

---

4. 실습 시 참고사항

- **GOPATH와 Go Modules**: 과거에는 `GOPATH`라는 환경 변수를 통해 의존성을 관리했으나, Go 1.11 버전 이후로는 **Go Modules** 사용이 표준. `go.mod` 파일이 있는 디렉토리에서는 `GOPATH`를 신경 쓰지 않아도 됨.

  > 신경쓸거 없다는 의미

- **자동 포맷팅**: Go는 `gofmt`라는 강력한 코드 포맷터를 기본 제공. VSCode의 Go 확장은 파일 저장 시 자동으로 코드를 `gofmt` 스타일로 정리해주므로, 코드 스타일을 일관되게 유지하기 매우 편리함.
- **사용하지 않는 변수/패키지**: Go 컴파일러는 사용하지 않는 변수나 임포트한 패키지가 있으면 에러를 발생시킴. 코드를 항상 깔끔하게 유지하도록 강제하는 Go의 특징 중 하나.