Solutions and Structure

There are 2 solutions, they are under “config1” and “config2” packages. The difference between them the first one was having **2 managers** where each one is responsible for its own type to say for example **YamlManager** and **JsonManager** however I’ve built the second solution based on one Manager that might have different types of files to say for example you want to compare and see difference between **json** config file with **yaml** config file.

Packages of applications were meaningfully named however I tried here to describe how they are built or structured.

1) **external** is having only 1 struct **Package** and I treated as it is coming from another library as you sent (no change)

2) conf is having an extension to **Package,** it is a decorator for the **Package** where it allows to extend its functionality easily.

conf already has a **json** or **yaml** configuration implementation. difference algorithms or strategies that will be built or implemented to read config files. Both of them are implementing IConfig interface that unifies their contract or behavior

4) **comparison** package where it has different comparer implementation (different strategies), I’ve included “**SimpleComparer**” that is based on reflect’s **DeepEqual** and “**GoComparer**” that is based on google’s package named **go-cmp t**hat is used for comparison or differences. Later we will use this comparer to be **injected** and returned by the **Factory method pattern** and only one instance will be used during the application’s life (based on **Singleton pattern**) to be able to compare between objects.

5) contracts has all behaviors that support different forms of implementations, **polymorphism,**

Assumptions:

- This was built to show only my thoughts on how I am designing and structuring the code
- I used Goroutines in Go (Runnable/Thread in Java or Async-Task in C#) to support parallelism of function calls to avoid requests overloads or throttling.
- Contracts used **heavily** to simplify changes and hooking up new functionalities
- I’ve added only one testing case under **config1** named **TestYmlManager_Compare** as only a sample
- In one of the comparer, I depended on google’s go-cmp to compare or differentiate betweens objects.

Language: Go 1.8
Design patterns: Factory Method, Singleton, Decorator, Strategy
IDE: Jetbrains Goland
VCS: Git/Github