Практическое задание:

В данной части нашего курса мы продолжим работать с репозиторием из прошлого блока. На самом деле, мы с вами понемногу начинаем писать наш сервис, смысл которого будет в мониторинге за состоянием других сервисов. Один из таких “других” сервисов - это так же небольшое приложение, написанное на go, которое умеет отдавать метрики. Называется этот сервис goMetr. Вы можете посмотреть на исходный код сервиса по ссылке https://gitlab.slurm.io/GoForOps/gometr.

Итак, в нашем задании мы должны создать структуру, которая нам в будущем поможет работать с сервисом goMetr. Все в том же репозитории в папке вашего сервиса нужно создать новый файл и назвать его gometr. В этом файле необходимо реализовать структуру GoMetrClient. В ней мы будем хранить url и число секунд для таймаута, поэтому нужно определить поля структуры.

Создадим еще один файл и назовем его checker. В новом файле уже создадим структуру Checker, у которой будет поле items содержащее слайс интерфейсов Checkable.

Тут же нужно объявить 2 интерфейса

- **Measurable** - у которого есть метод получения метрик GetMetrics, который ничего не принимает и возвращает строку
- **Checkable** - которые расширяется интерфейсом Measurable и имеет ряд своих методов:
    - Ping - возвращающий тип ошибки
    - GetID - возвращающий строку
    - Health - возвращающий истина или ложь

Также, для структуры Checker нужен метод Add, который добавляет переданные в качестве параметра элементы в список items.

И еще одна деталь, мы хотим, чтобы при передаче созданной и заполненной структуры Checker в метод fmt.Print() на экран выводились результаты GetID всех добавленных в Checker элементов. Для этого Checker должен соответствовать встроенному в Go интерфейсу Stringer.

И сердце нашего метода - метод Check(). Он должен выполнять проверки Health добавленных в чекер элементов. Если проверка провалена - выводим на экран fmt.Println(идентификатор_элемента + " не работает")

Переключимся снова на нашу структуру GoMetrClient. Теперь нужно дописать ее таким образом, чтобы она соответствовала интерфейсу **Checkable.** (к возвращаемым значениям на данном этапе требований нет)

Теперь, в main файле в функции main() мы можем перенести реализацию написанной там функции GenerateCheck. Нужно удалить эту функцию и в структуре GoMetrClient создать метод getHealth, который вернет структуру HealthCheck. Итак, здесь у нас GetID - это строка (например, это может быть хост сервиса или url), поэтому в структуре HealthCheck поле ServiceID тоже нужно изменить на строку и перенеси в файл с GoMetrClient.

И нам теперь нужно в методе Health структуры GoMetrClient вызвать getHealth и проверить состояние. (для разнообразия, можете в getHealth описать разные состояния для разных ServiceID).

Теперь main функция снова пуста, и нам нужно создать структуру Checker, несколько раз вызвать метод Add, создавая и передавая в нее структуры GoMetrClient.

В результате выполнения fmt.Print(созданная_структура_Checker) мы увидим на экране вывод результатов метода GetID из всех структур GoMetrClient, добавленных внутрь структуры Checker.

После этого выполним метод Check у нашего чекера и посмотрим на вывод, если есть нерабочие зарегистрированные сервисы, нам выйдет сообщение следующего вида:

host1 не работает

P.s. Не забывайте про методы “конструкторы” для структур, о которых мы говорили.