Давайте посмотрим на организацию модулей на конкретном примере. Итак, давайте рассмотрим, значит, модульности Package на примере моего любимого kubernetes client-go, который является клиентом к официальному IP в kubernetes-a. И что мы здесь видим? Мы видим, значит, кучу папок, ну кроме папки github, которая не относится к Golang, кучу папок. Мы видим здесь какие-то файлы, которые относятся, непосредственно, ко всяким readme и прочим лицензиям. И видим в файле go.mod, которая определяет, что этот, значит, проект является модулем golang.

Что есть в модуле? В модуле есть путь. Это путь, по которому golang будет знать ваш модуль. Тот путь, который будет указываться во всех импортах внутри. Значит, то, что модуль требует, его зависимости. Мы с ними поработаем еще чуть позже, посмотрим на них. И replace-ы, которые заменят какие-то зависимости на конкретные версии модулей. Это нужно, если у вас несколько есть зависимостей, вы хотите какой-то модуль подменить на свой, ну либо на какую-то конкретную версию.

Соответственно, давайте посмотрим на папочки. Посмотрим на папочку informers, например. В папочке informers есть еще папочки и есть, соответственно, файлик, например, factory.go. Если мы в него зайдем, то увидим, что это является частью Package informers. Называется также, как и сама папка. Достаточно удобно, потому что пути совпадают. И, соответственно, когда вы импортируете, у вас нет никаких проблем. Дальше мы видим кучу импортов, и уже идет какой-то код. Импорты нужны для того, чтобы добавить какие-то чужие, соответственно, ну или свои же классы в ваш Package. Package может зависеть от других Package-ей, в этом нет никаких проблем.

Соответственно, если мы зайдем в другие подмодули, то мы увидим, что, например вот в папочке informers batch v1 Package – не informers batch v1, а

просто – Package v1. И, если мы что-то оттуда импортируем, то мы просто присваиваем ему свое имя и указываем конкретный путь, потому что у нас может быть несколько Package-ей с одинаковым именем.

Значит, в принципе, вот эти все Package-и, все вместе, да, все вместе работают в составе модуля. В одном Package может включаться контент из других Package-ей. Соединять 2 Package-а в одном файлике, соответственно, нельзя. Каждый Package должен лежать в своем файле.