Chapter 3: Common Patterns: *if someone missed to catch it yet* 😉

```
Object  Vk      VkInstance
Types   Vk      VkInstanceCreateInfo
Funcs   vk      vkCreateInstance()
Enums   VK_     VK_STRUCTURE_TYPE_INSTANCE_CREATE_INFO

Extensions
    KHR:- Khronos authored,
    EXT:- multi-company authored

Creating "VkZZZ" object
    1. take `VkZZZCreateInfo` --> fill it up
    2. call `vkCreateZZZ()`
    3. also `vkDestroyZZZ()` before closing your app

    4. Some objects get "allocated" rather than "created"
        `VkZZZAllocateInfo` --> `vkAllocateZZZ` --> `vkFreeZZZ`

    5. Sometimes there will be `.zzzCreateInfoCount` & `.pZZZCreateInfos`
                        e.g. `.queueCreateInfoCount` & `.pQueueCreateInfos``
            -> So you could like pass in an array/vector
            -> You will see this in lots of other places

Getting List/Properties
    1. vkEnumerateZZZ() --> \see `[Chapter2.1.] vkEnumeratePhysicalDevices()` example
```

-- | -- | -- | ----------

7. `sType` & `pNext`
    - *Many Vulkan structures include these two common fields*
8. `sType` :-
    - *It may seem somewhat redundant, but this information can be useful for the* `vulkan-loader` *and actual* `gpu-driver-implementations` *to know what type of structure was passed in through* `pNext` *.*
9. `pNext` :-
    - *allows to create a linked list between structures.*
    - *It is mostly used when dealing with extensions that expose new structures to provide additional information to the* `vulkan-loader` *,* `debugging-validation-layers` *, and* `gpu-driver-implementations` *.*
        - *i.e. they can use the* `pNext->stype` *field to know what's ahead in the linked list*

-- | -- | -- | ----------

```
10. Do remember to check the `Valid Usage` section within each manual-page
```

Two Questions I keep on pondering 😗

```
- a) Would this make sense to someone else?
- b) Would this make sense to a 5 year old?
```

Keywords in this file

```
ChapterZZZ => **"ChapterZZZ"** Unknown WIP/TBD Chapter
  REY_DOCs =>
ChapterZ.Z => **_Chapter1.2_**
```