**Which structures exist and wtf are they?:**

Structures are a software engineer’s lingo for describing different types of flowcharts/diagrams. Afaik there are 4 of these shit critters:

The **Hierarchy Pattern** is basically the idea of linked concepts being subspaces of each other. An example could be student, who can be part of a group, where groups are part of a semester. In this scenario, each student is a subspace of what forms a group, while groups are subspaces that form the semester. In programming lingo, it would be like creating a class called *warship*, and then have other classes extend it, such as *destroyer* or *heavy cruiser****.***

The **Relation Pattern** on the other hand is all about connecting objects through concepts. For example, a *person* may be related to physical objects like a *car* through the concept of *ownership*. Do note that concepts can also be objects in the diagram, assuming the square thingies *are* objects. Other examples could be two *people* related through the concept of *friendship*.

Then there’s the **Descriptor-Item Pattern (Genstandbeskrivelses mønsteret).** This structure is a bit different, in that the linking between objects is based around the description of a thing, and its relation to each instance of said thing. It’s essentially just an abstract definition of classes and objects. The short end of it is, we have some definition of a thing, and every object which is made from this definition will all have a set of specific information. It’s hard to explain, but it sort of just makes sense in the head if you just see it as classes and objects.

Lastly there’s the **Role Pattern**. This pattern is basically the idea that objects can be assigned to each others as extensions. The idea that some object, such as a *Person*, can host another object in the form of a role, such as *Employee, Visitor, Criminal, etc.*, which adds functionality and responsibility to the host.

**Related terms:**

So those patterns up there? They got some related terms or some shit that describes them. These are:

**Generalization (Hierarchy),** Classes extend other classes. Nothing more needs to be said on this matter.

**Association (Relation),** Shit that in some contexts interact will be linked together. Think of a human and a car. The human drives the car. Alternatively, the car drives the m’fucking human way into next week. *Shit’s related yo.*

**Aggregation (Role),** a deconstruction of an object to view all the parts it is made of. Say you have an aggregation of a car, that car would then branch down to an engine, some wheels, which then branch further based on what those are made of.

**Cluster (Descriptor),** everything is related in *some* way. A car can be related to a motor in *some* way, while a cylinder can be related to a motor in *some* way. Meanwhile, the car can also be related to a personal car in *some* way, while the personal car can be related to a taxi in *some* way. All these connections make up the cluster *car*.

~~There doesn’t really seem to be a way to pinpoint wtf a cluster is connected to, as it’s sort of steaming hotpot of everything mentioned so far. And let me tell you, that hotpot is of the~~ *~~shitty~~* ~~kind.~~

**Why is it only object structures that have “mangfoldighed”?:**

It is because “mangfoldighed” is the idea that an object can contain other individual or intervals of objects. Classes cannot do this, as they can only describe the type of shit they *can* have, not what they *will have*.

**What is “nedarvning”?:**

Classes/Objects can inherit structure and information from other classes/objects.

**Phone classes and events:**

Phone:

Classes:

Call

Phonebook

Peeps

Log

Events:

Call received

Call requested

Call initiated

Call ended

Call logged

Other caller saved

Other caller removed

Log accessed

Log cleared

Log exported

**Quiz shit for later analysis:**







