STRATEGY AND TYPE
OBJECT
DANIEL NAEL
Note
ContextBehaviour()
{
    strategy.InterfaceBehaviour();
}

ConcreteStrategyA
+
InterfaceBehaviour()
Does it look familiar?
Strategy

Define a family of algorithms, encapsulate each one, and make them interchangeable.

Strategy lets the algorithm vary independently from clients that use it.

State

Allow an object to alter its behaviour when its internal state changes. The object will appear to change its class.
STRATEGY - COMPARISON

**Strategy**
- Strategies do not know of each other's existence.
- Assignment of strategies to the context is out of scope of the strategy pattern.
- Applications may have only a single concrete strategy and use the pattern to support plugins and modifications.

**State**
- High class coupling between states.
- States often transition the context into another state based on input.
- Application should have multiple states for the context. Otherwise it only increases complexity without adding any benefits.
TYPE OBJECT
What’s wrong here?
What's wrong here?

How can we fix this?
TYPE OBJECT - INTENT

Allows the flexible creation of new “classes” by creating a single class, each instance of which represents a different type of object.
TYPE OBJECT - STRUCTURE

**Typed object**
- Has a contextual role in the application
- Encapsulates a type object
- Behaviour depends on data from type object

**Type object**
- Provides data to the typed object
- May provide method implementations to the typed object
**Type Object - Comparison**

**Strategy**
- Generally assigned once in the lifecycle of the context.
- Works great with flyweight
- A dynamic way to provide the context class with algorithms.

**Type object**
- Generally assigned once in the lifecycle of the context.
- A dynamic way to provide the context class with type and data.
- Requires a solution for creating and editing type data outside application and deserializing it into type objects.
- Works great with flyweight
UNITY SCRIPTABLE OBJECTS

- Can’t be directly attached to gameobjects
- Great for use as Type objects in Unity
- Assets in the project or deserialized during runtime
- **Support reading and writing from json**
- Can reference other scriptable objects

We do not use these during the course, but you can start learning from here: