**State Pattern**

**Overview**

As the name symbolizes ‘state pattern’ is used in application to show state changes of one to another by allowing object to alter its behaviour when it’s internal state changes. Object will appear to change its class. It is a simple to understand design pattern and very useful as well in implementation. Freeman Eric., Freeman Elisa. (2004).

**Characters of state pattern**

According to Freeman Eric., Freeman Elisa. (2004) state pattern has following attributes/characters.

* A context is the class that has number of internal states.
* An interface defines common states objects can have; when same interface is implemented but it can be used interchangeably.
* Many states are possible to exist.
* Objects that implement interface defined by ‘concrete-state’ can handle request from context. When context changes state, its behaviour will change as well.
* Request to change state starts from context and it eventually gets passed on to state where multiple objects listen to state changes and use of function changes states from one to another.

**Benefits**

Objects can encapsulate their own internal states and talk to each other when state changes by passing messages.

**Scenario / Example**

In a payment service in EFTPOS, payment is done through card or cash. When card or cash is entered machine processes payment. There are many states during that time such as if cash is entered it calculates amount to be paid, amount entered, calculate change, return change or ask for more money if entered amount is less than amount required.

**Before Refactor**

Before refactor, without using state pattern, possibly it’s hard to manage states as normal objects which do not implement interface and talk to each other by listening to state change request would not be able to achieve different states.

<https://github.com/sppanday/S120-PRT583-Group-A/tree/master/s260598-PandaySurendra/Sprint-2-Deliverables/Task019_StatePattern/StatePattern/StatePattern/Before>

**After Refactor**

After refactor, objects would make usage of interface implementation and handle request through ‘state.handle()’ function.

<https://github.com/sppanday/S120-PRT583-Group-A/tree/master/s260598-PandaySurendra/Sprint-2-Deliverables/Task019_StatePattern/StatePattern/StatePattern/After>

**UML Diagram**

Please find it inside relevant project uml in gitlab

/StatePatternUML.png

Alternatively visit https://app.creately.com/diagram/EeSx5TqcgmX/edit

**Reference**

Freeman Eric., Freeman Elisa. 2004. 1st ed. O’Reily Media Inc. 1995, Sebasttopo, CA