**Command Pattern**

**Overview**

A command pattern stores a list of code that is executed at later time or repeatedly as instructed. A client object can prefer to run particular command on one of hidden objects. An object such as invoker transfers command to another object called receiver to execute right code.

Freeman Eric., Freeman Elisa. (2004) explains Concrete pattern has following steps carried out.

* A client creates a concrete command and sets its receiver.
* An invoker holds a command and at some point asks to carry out request by calling it’s **execute()** method.
* An invoker would know how to fulfil the request
* An action is created when execute method is executed
* Any class can act as a receiver which runs action() method to do right action.

**Benefits**

* A command pattern encapsulates a request as an object, and lets parameterize other object for a different command or operation.
* Use of command pattern is best for reversible operation such as light off/on.

**Scenario / Example**

An example of tv remote can be used. A tv remote would have bunch of buttons to go to next channels, turn tv on / off as well if needed. For a list of given channels we want to be able to change channels from remote control and be able to turn tv off/on. It has a set of commands to executed, action raised, and event triggered.

**Before Refactor / After Refactor**

<https://github.com/sppanday/S120-PRT583-Group-A/tree/master/s260598-PandaySurendra/Sprint-2-Deliverables/Task021_CommandPattern/CommandPattern/CommandPattern>

**UML Diagram**

Please find it inside relevant project uml in gitlab

/CommandPatternPatternUML.png

<https://www.lucidchart.com/documents/edit/a0792e09-3c75-4e0f-b362-a50264fcc7bd/HWEp-vi-RSFO>

**Reference**

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