**Command Pattern**

**Overview:**

Command pattern is a data driven design pattern and falls under behavioural pattern category. A request is wrapped under an object as command and passed to invoker object. Invoker object looks for the appropriate object which can handle this command and passes the command to the corresponding object which executes the command.

**Details:**

Four terms always associated with the command pattern are,

* *Command*
  + A *command* object knows about *receiver* and invokes a method of the receiver. Values for parameters of the receiver method are stored in the command.
* *Receiver*
  + The *receiver* then does the work. An *invoker* object knows how to execute a command, and optionally does bookkeeping about the command execution.
* *invoker*
  + The invoker does not know anything about a concrete command, it knows only about command interface. Both an invoker object and several command objects are held by a *client* object.
* *client*
  + The client decides which commands to execute at which points. To execute a command, it passes the command object to the invoker object.

**Use cases:**

Command objects are useful for implementing

I have planned to implement Command Pattern for game menu

* GUI buttons and menu items
* [Macro](https://en.wikipedia.org/wiki/Macro_(computer_science)) recording
* [Mobile Code](https://en.wikipedia.org/wiki/Code_Mobility)
* Multi-level [undo](https://en.wikipedia.org/wiki/Undo)
* Networking
* Parallel Processing
* [Progress bars](https://en.wikipedia.org/wiki/Progress_bar)
* [Thread pools](https://en.wikipedia.org/wiki/Thread_pool)
* [Transactional](https://en.wikipedia.org/wiki/Database_transaction) behaviour
* [Wizards](https://en.wikipedia.org/wiki/Wizard_(software))