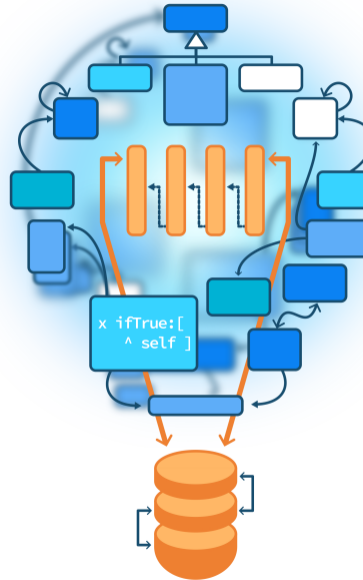


# Command Design Pattern

Actions as objects

S.Ducasse, L. Fabresse, G. Polito, and P. Tesone



# Goals

- Little motivation
- Power of reification of actions
- Command Design Pattern



# Imagine a scriptable robot

```
testExecute
| rb b |
rb := RbsRobot new.
rb startLocation: 4@1.
rb execute:
'dir #east
mov 2
mov 3
dir #north
mov 3'.
self assert: rb position equals: 9@4
```



## Execute (first version)

```
RbsRobot >> execute: aString
```

```
orders := aString splitOn: Character cr.
```

```
orders := orders collect: [ :each | each splitOn: Character space ].
```

```
orders do: [ :each |
```

```
  each first = 'mov'
```

```
    ifTrue: [ self move: (Object readFrom: each second) ]
```

```
    ifFalse: [ each first = 'dir'
```

```
      ifTrue: [ self direction: (Object readFrom: each second) ] ] ]
```



## Execute (more actions)

```
RbsRobot >> execute: aString
```

```
orders := aString splitOn: Character cr.
```

```
orders := orders collect: [ :each | each splitOn: Character space ].
```

```
orders do: [ :each |
```

```
  each first = 'mov'
```

```
    ifTrue: [ self move: (Object readFrom: each second) ]
```

```
    ifFalse: [ each first = 'dir'
```

```
      ifTrue: [ self direction: (Object readFrom: each second) ]
```

```
      ifFalse: [ each first = 'drop' ]]
```

```
    ...
```

```
    each first = 'pick'
```

```
    ...
```

```
    each first = 'return' ]
```



# Analysis

- Each time we add a new order we **have to modify** execute:
- Imagine if a `mov` order cost a lot
  - Better to have one over many ones
  - `mov 10 mov 10 mov 10 -> mov 30`
  - Not simple to perform a simple path optimization
- How to replay the exact low-level executions



# Command Design Pattern

**Intent from the book:** Encapsulate a request or operation as an object, thereby letting you parametrize clients with different operations, queue or log request, and support undoable operations



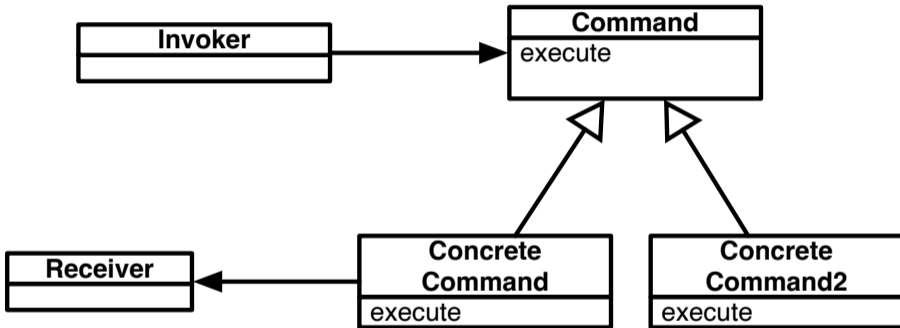
# A command

- A command is a reification of an order/action
- A command encapsulates an action and optionally its context
  - menu item
  - log action
- Commands are often the basis for Undo





# Command core



# Robot direction command

```
RbsCommand << #RbsDirectionCommand  
  slots: { #direction };  
  tag: 'Commands';  
  package: 'Robots'
```

```
RbsDirectionCommand << handleArguments: aCollection  
  direction := aCollection first asSymbol
```

```
RbsDirectionCommand << executeOn: aRobot  
  aRobot direction: direction
```



# Robot move command

```
RbsCommand << #RbsMoveCommand  
  slots: { #distance };  
  tag: 'Commands';  
  package: 'Robots'
```

```
RbsMoveCommand << handleArguments: aCollection  
  direction := Object readFrom: aCollection first
```

```
RbsMoveCommand << executeOn: aRobot  
  aRobot move: distance
```



# Modular execution logic

```
RbsRobot >> executeCommandBased: aString
```

```
orders := aString splitOn: Character cr.
```

```
orders := orders collect: [ :each | each splitOn: Character space ].
```

```
orders do: [ :each |
```

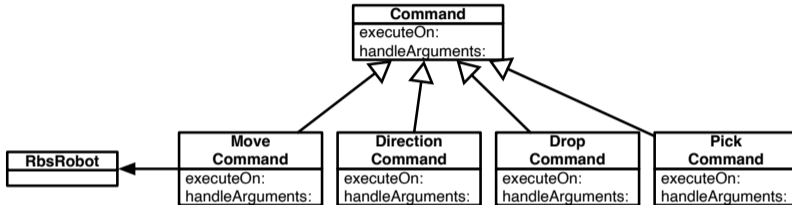
```
    (self commandClassFor: each first) new
```

```
    handleArguments: each allButFirst;
```

```
    executeOn: self ]
```



# Analysis of extensibility in place



- Each command is **responsible** for handling its **own data**
- Each command encapsulates its state, applicability and action
- Can now manipulate actions (log, sort, undo....)

# Command cons

- Not all operations should be turned into Command objects
- Produce large hierarchies of simple classes
- Pay attention not to externalize key object behavior
  - a class should still be complete
  - better if a command represents an existing behavior



# Conclusion

- Commands are first class actions
- Adapted for manipulation of actions (undo, replay)



Produced as part of the course on <http://www.fun-mooc.fr>

# Advanced Object-Oriented Design and Development with Pharo

A course by

S.Ducasse, L. Fabresse, G. Polito, and P. Tesone



Except where otherwise noted, this work is licensed under CC BY-NC-ND 3.0 France  
<https://creativecommons.org/licenses/by-nc-nd/3.0/fr/>