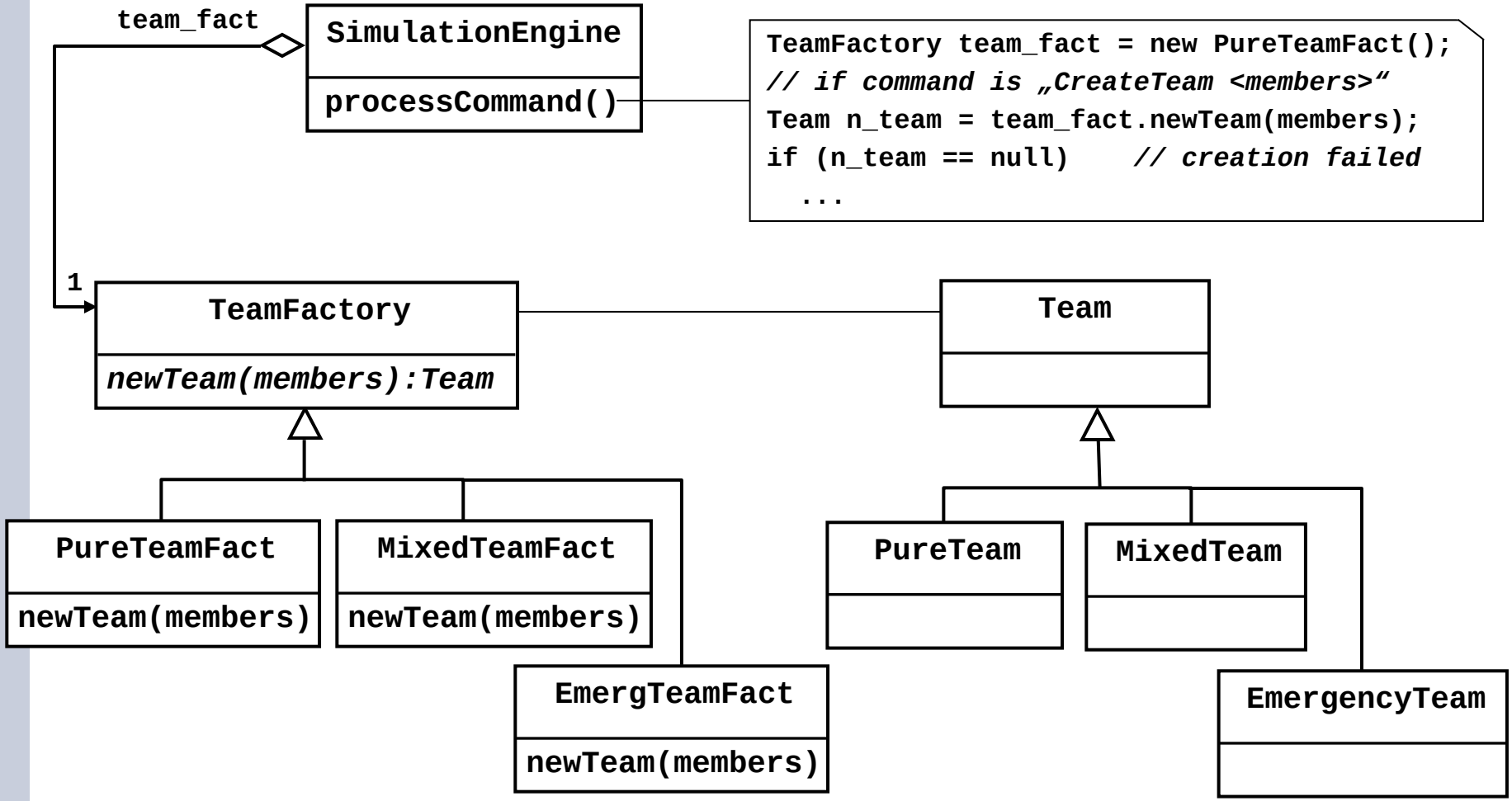
Design Patterns (IV)

- Prototype
- Command
- Memento

UNIVERSITÄT SALZBURG

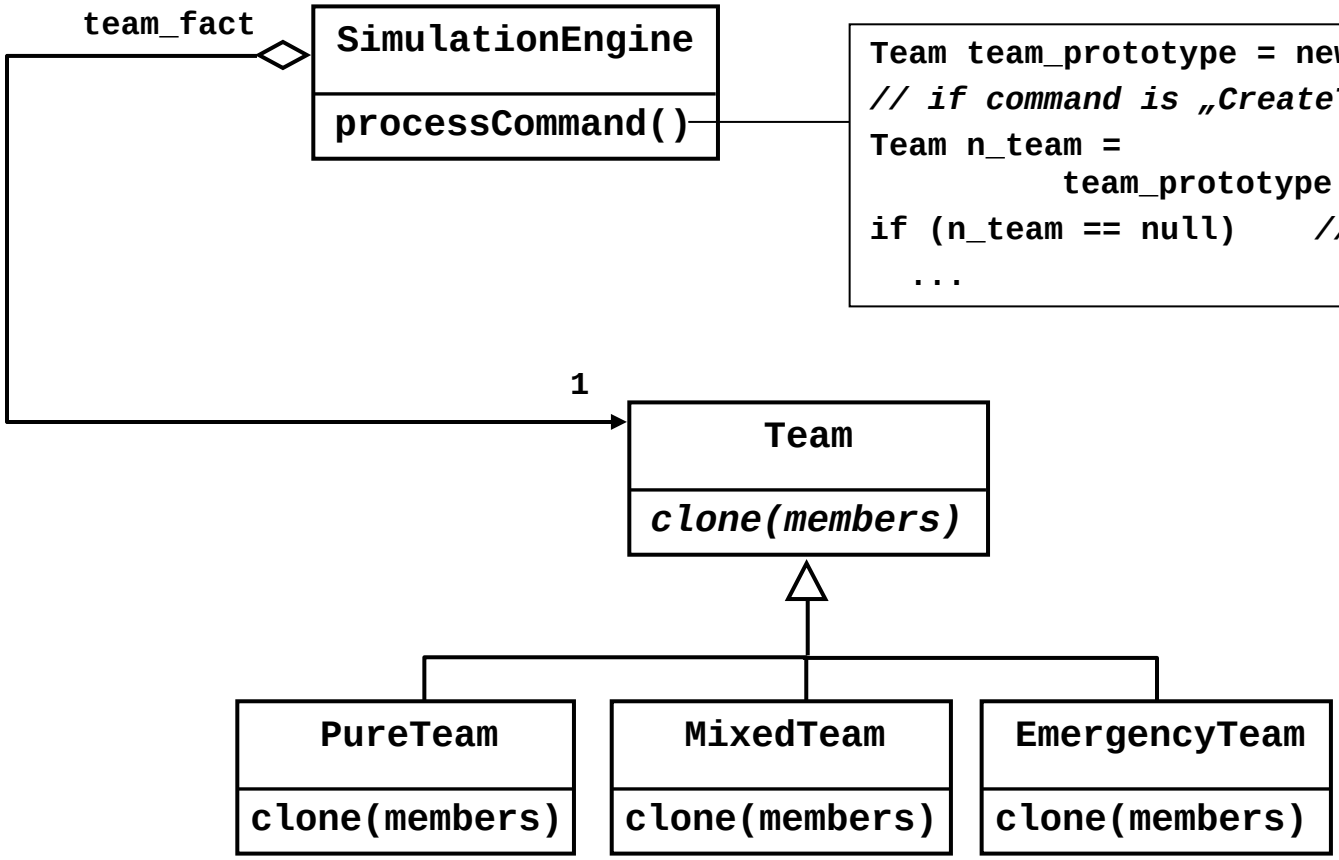
The Prototype Design Pattern: Example (I)



Abstract Factory for creating „team“ objects

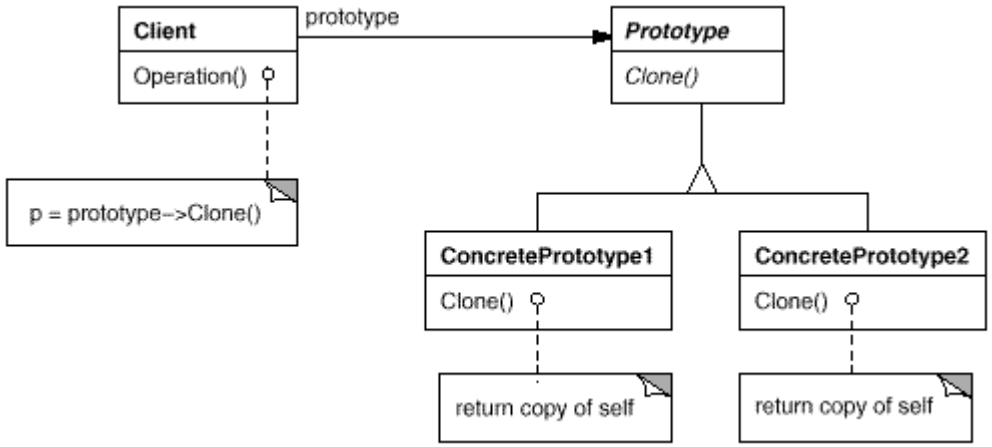
UNIVERSITÄT SALZBURG

The Prototype Design Pattern: Example (II)



Prototype for creating „team“ objects

The Prototype Design Pattern: Structure

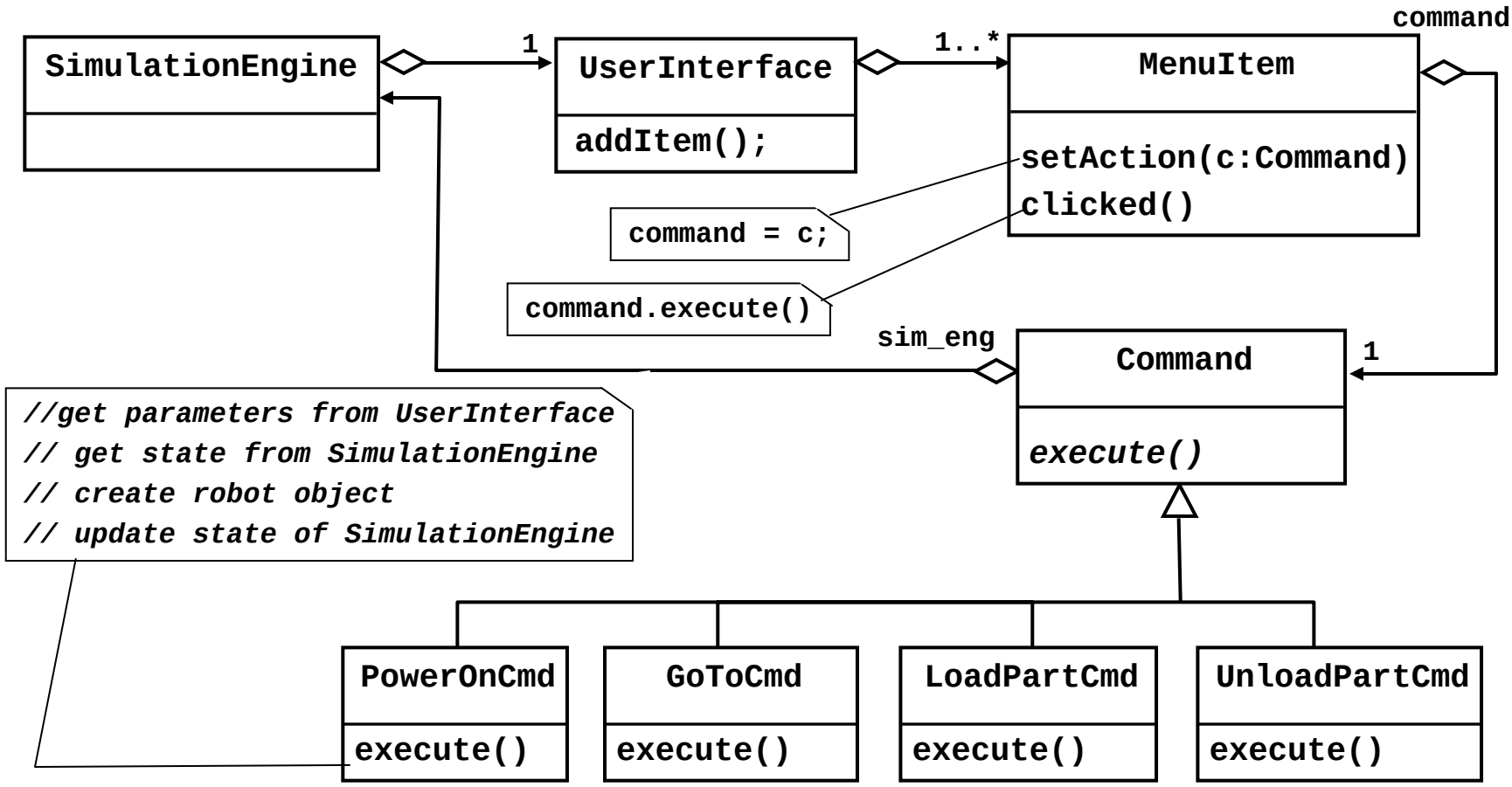


Use:

- When product creation should be decoupled from system behavior
- To avoid spurious subclassing (by eliminating class hierarchies of factories)

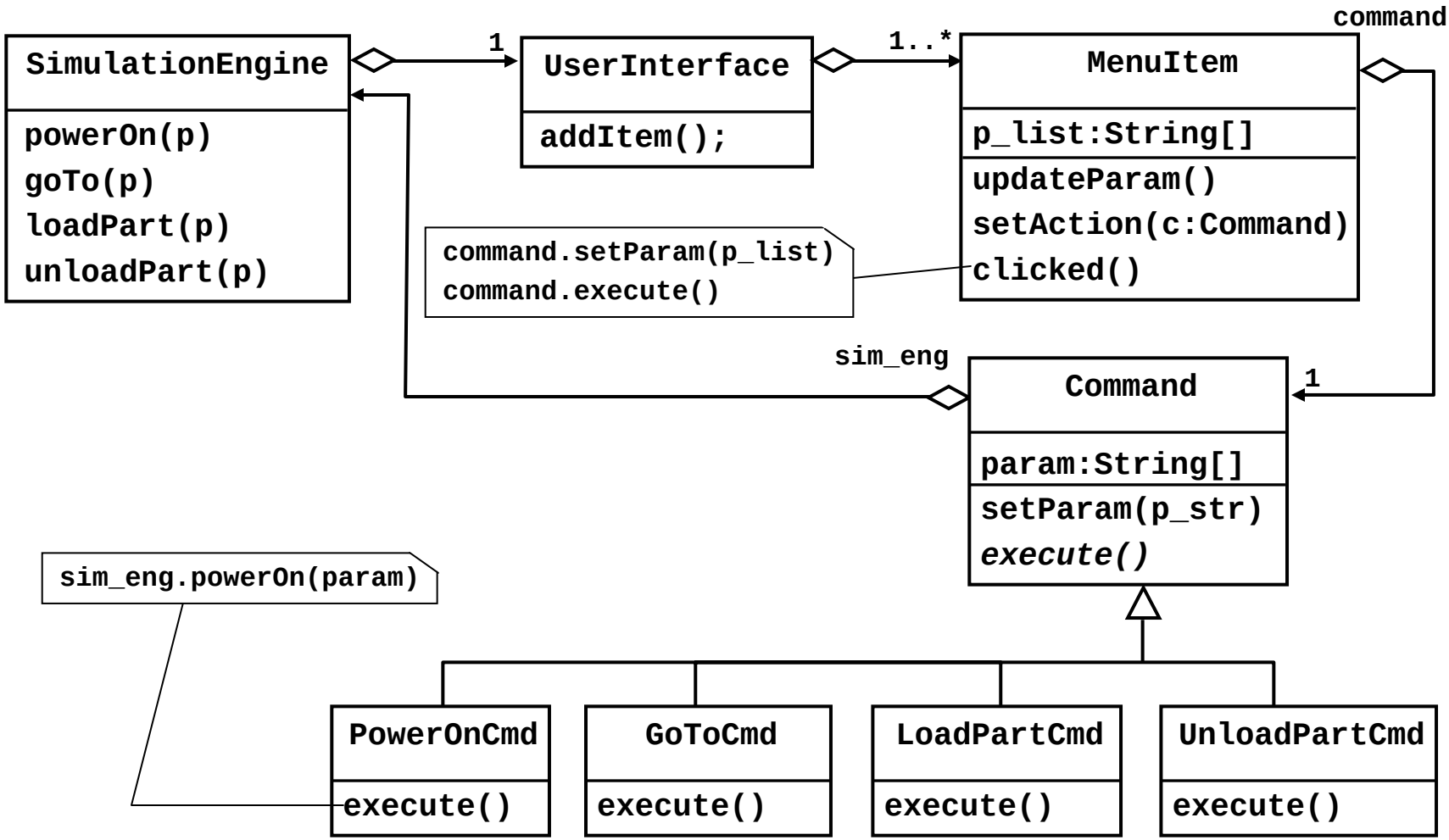
UNIVERSITÄT SALZBURG

The Command Design Pattern: Example (I)



UNIVERSITÄT SALZBURG

The Command Design Pattern: Example (II)



UNIVERSITÄT SALZBURG

The Command Design Pattern: Example (II)

- Usage:

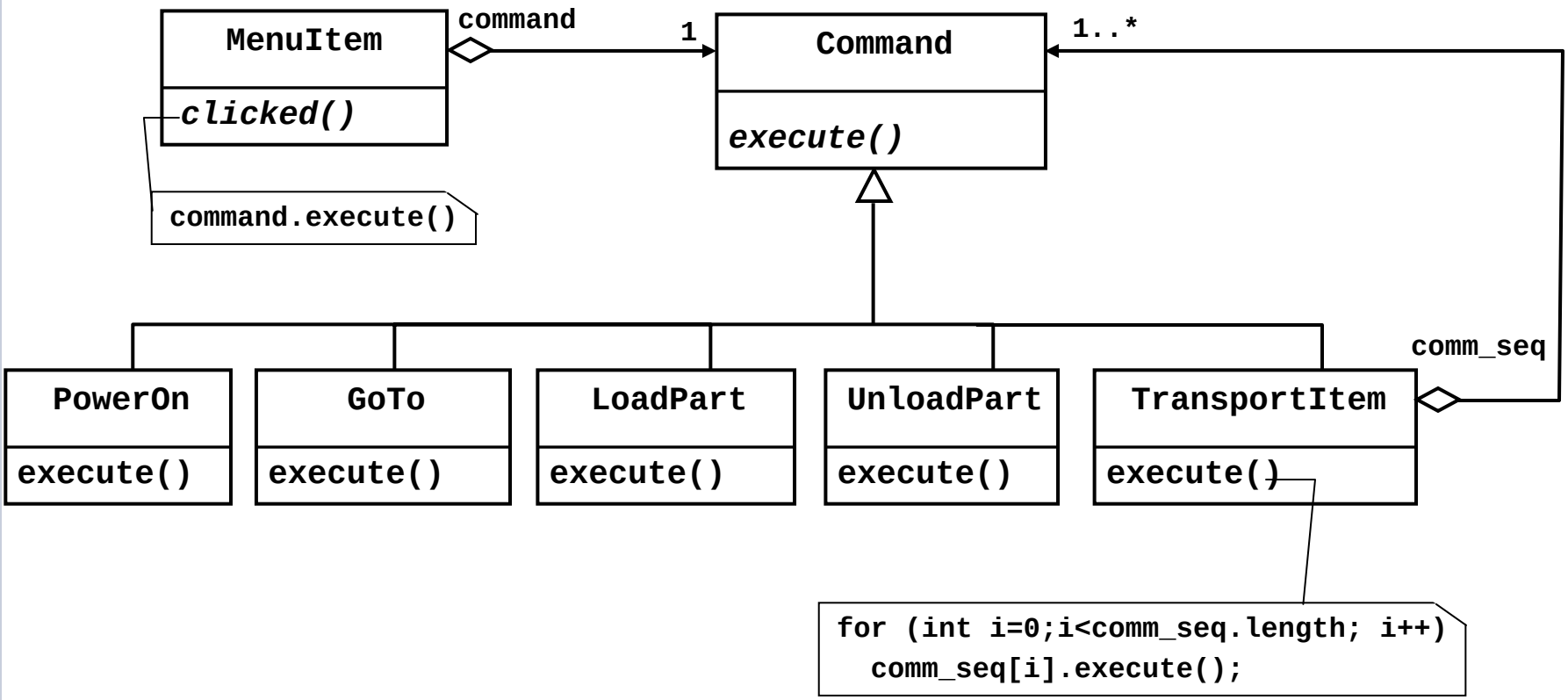
```
MenuItem bt_power_on = new Button(„PowerOn“);
Command cmd_power_on = new PowerOnCmd();
bt_power_on.setAction(cmd_power_on);
addItem(bt_power_on);
```

- With anonymous classes

```
bt_power_on.setAction(new Command{
  public void execute(){sim_eng.powerOn(param);}
});
```

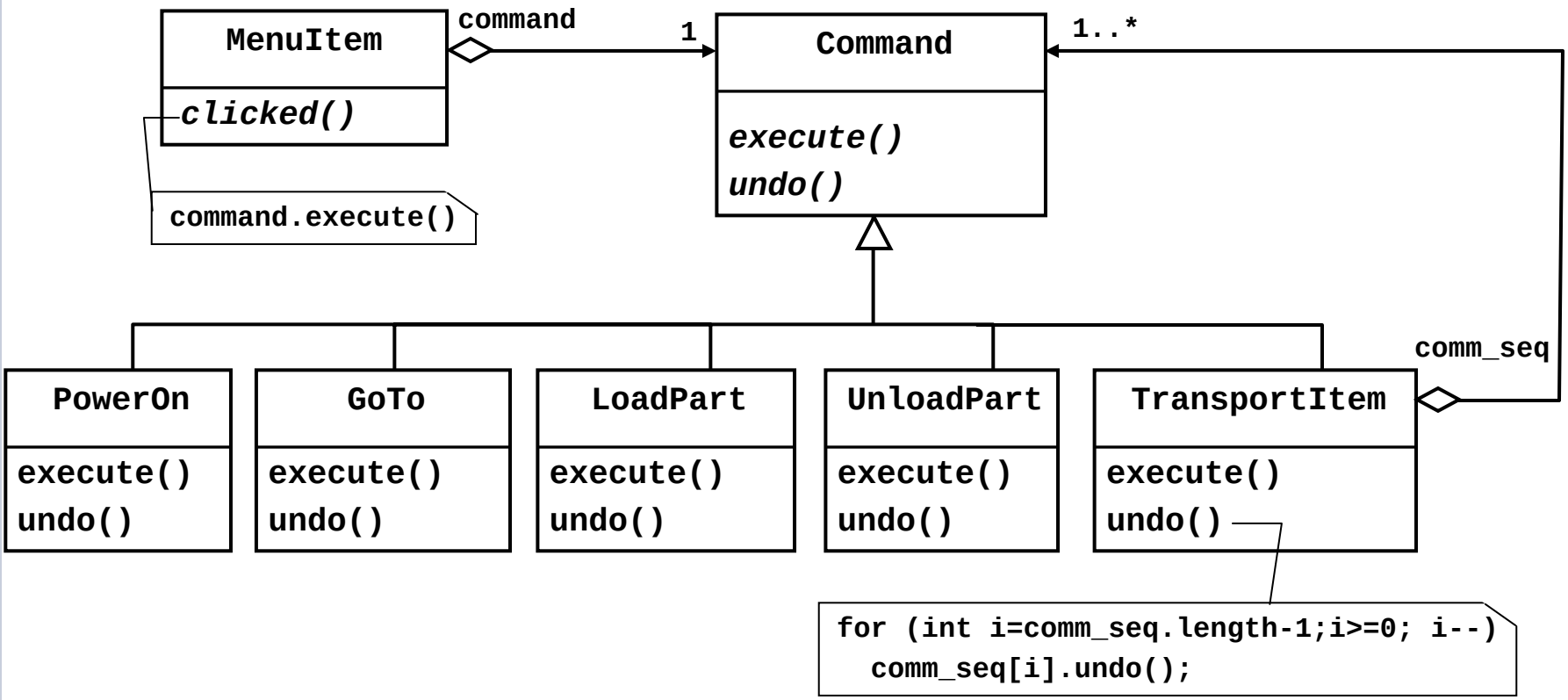
UNIVERSITÄT SALZBURG

The Command Design Pattern: Example (III)



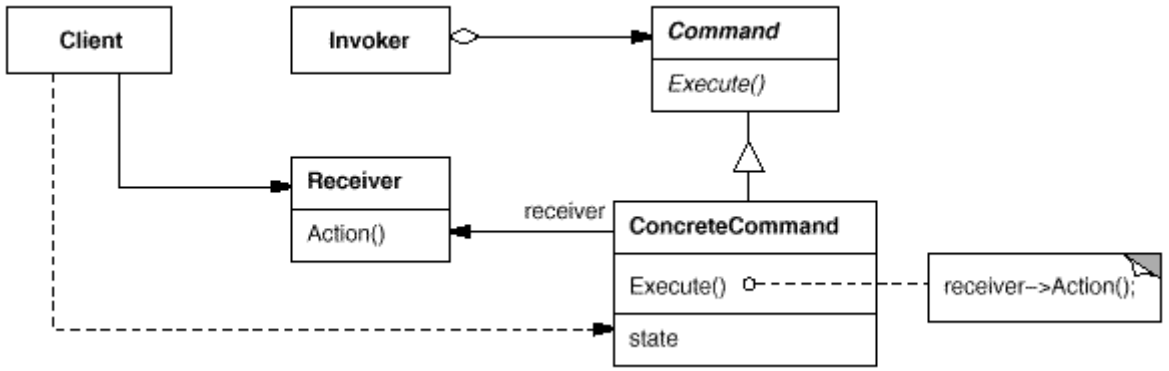
- Command sequence as Composite

The Command Design Pattern: Example (IV)



- Undo operation

The Command Design Pattern: Structure

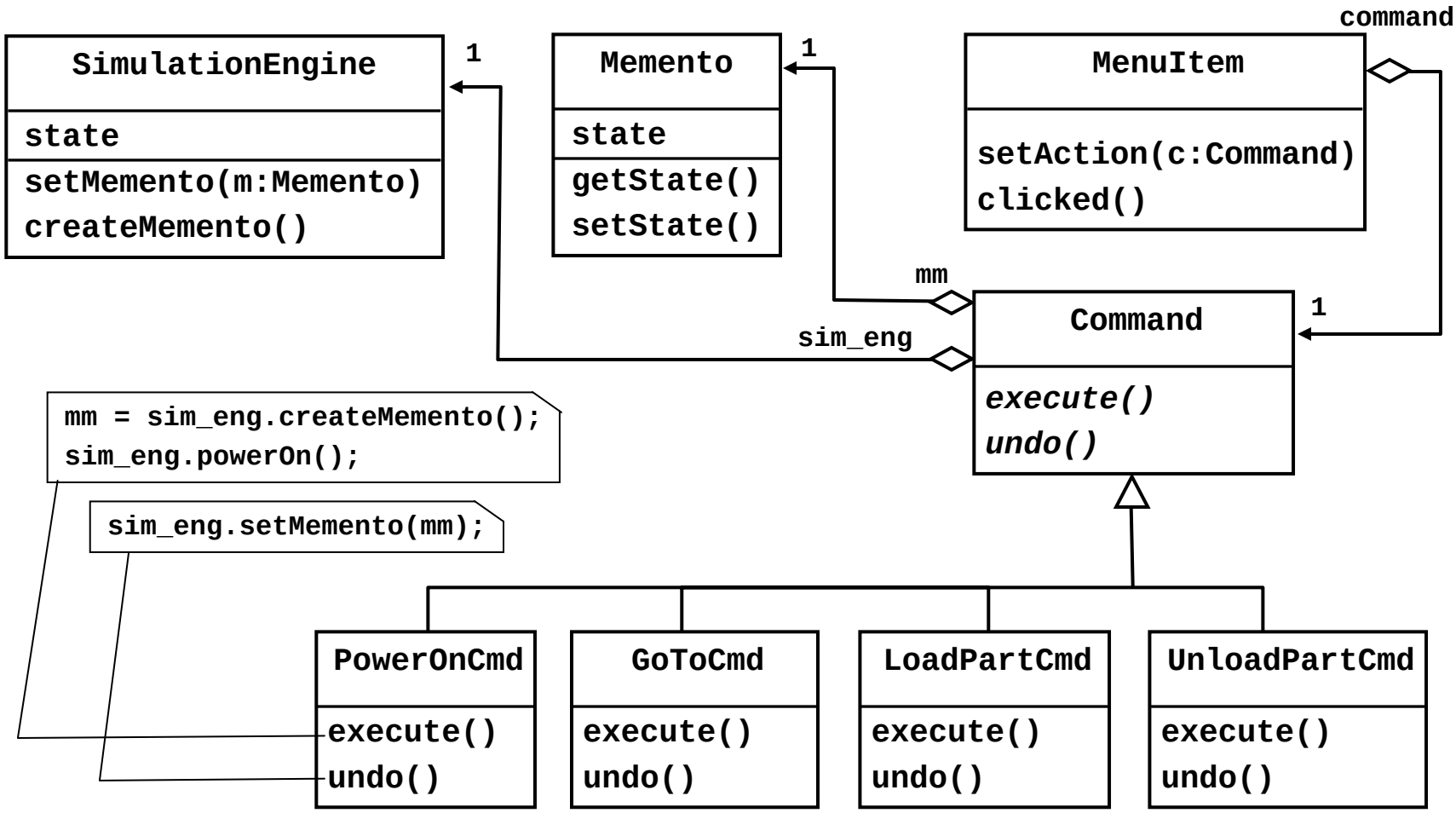


Use:

- To parameterize objects by actions
- Support undo
- To manage persistent commands

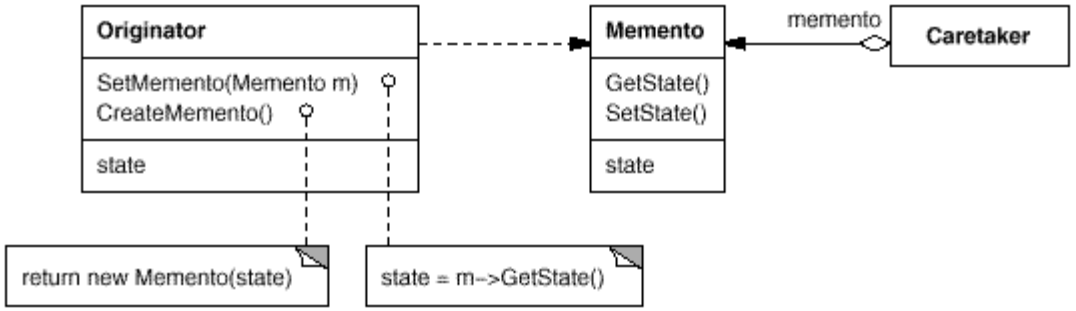
UNIVERSITÄT SALZBURG

The Memento Design Pattern: Example



UNIVERSITÄT SALZBURG

The Memento Design Pattern: Structure



Use:

- When a snapshot of an object‘s state must be saved in order to be restored later
- To hide implementation details of objects whose state should be saved

UNIVERSITÄT SALZBURG