CMPSCI 220 Programming Methodology

08: Command Pattern

Based on Head First Design Patterns

Objectives

Command Pattern

- Learn how objects can encapsulate invocation.
- Apply the command pattern in Scala.
- Learn how to evolve the command pattern to support undo.

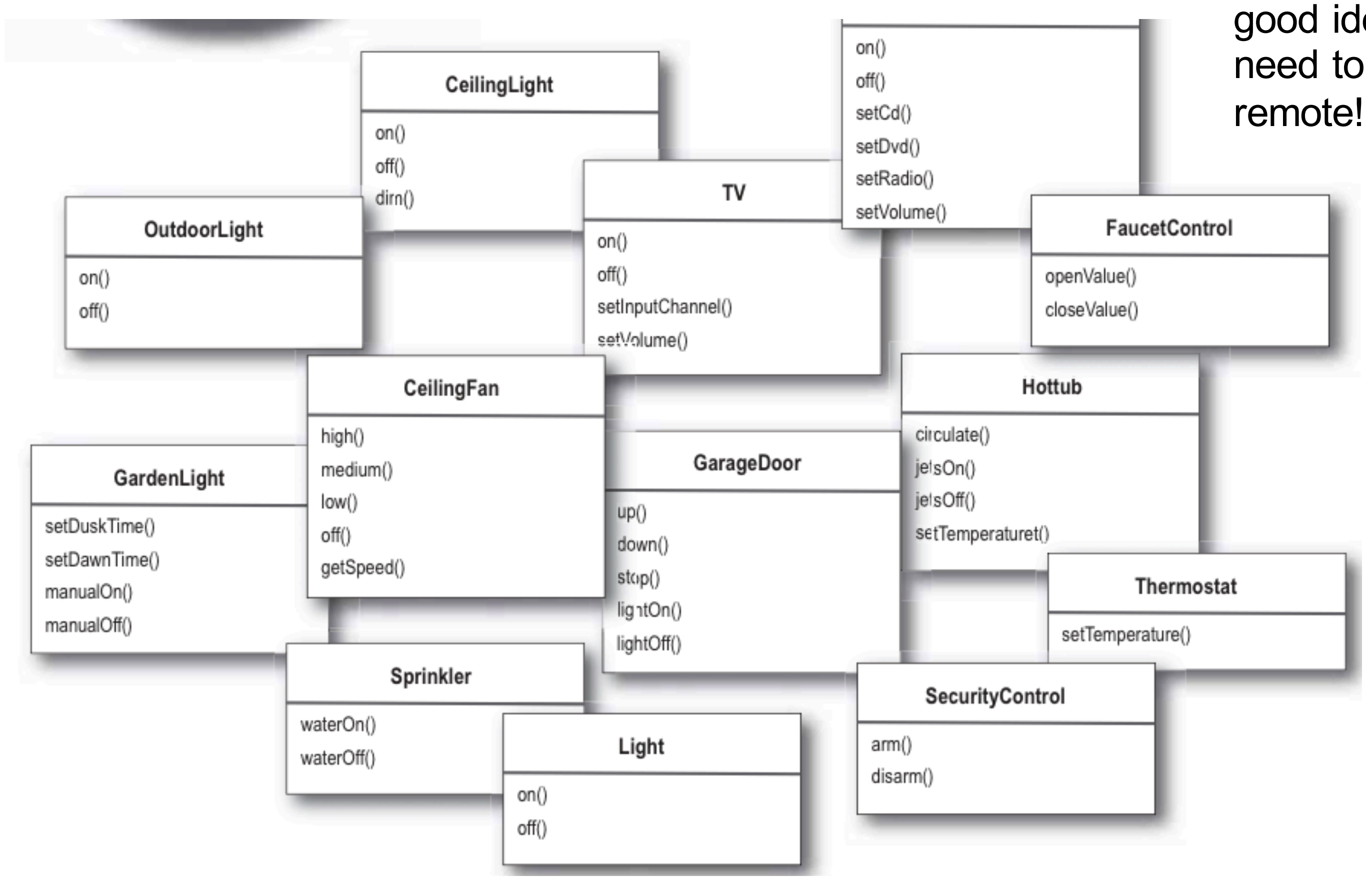
Free Hardware!

Home Automation or Bust!

You have been hired to implement an API for programming a remote for home automation. The goal is to make it as easy and as flexible as possible to allow the remote to be easily reprogrammed.

Are you up for the challenge?

The Vendor Classes

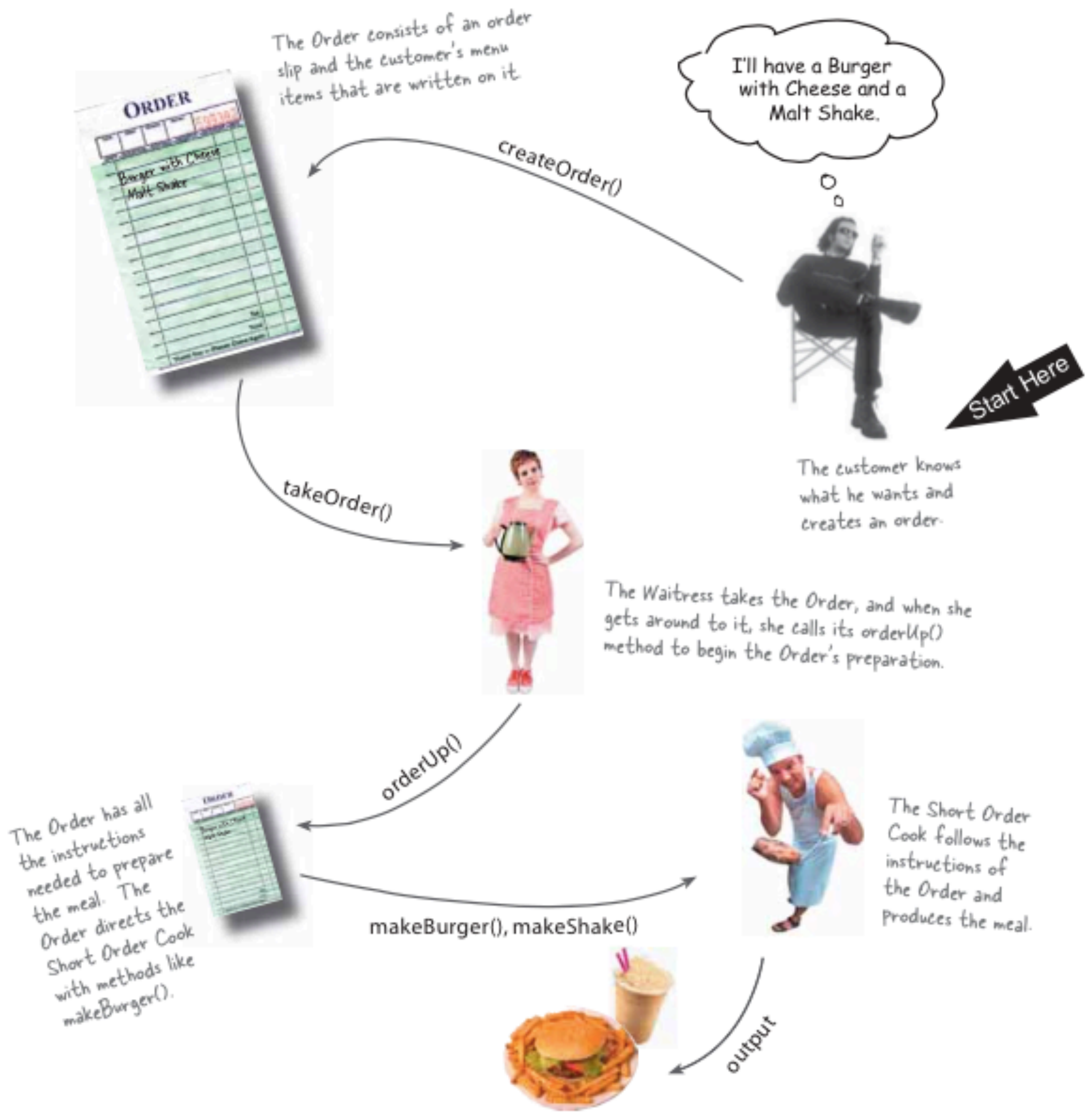


This should give you a good idea of what we need to control from the remote!

A Brief Introduction to the Command Pattern

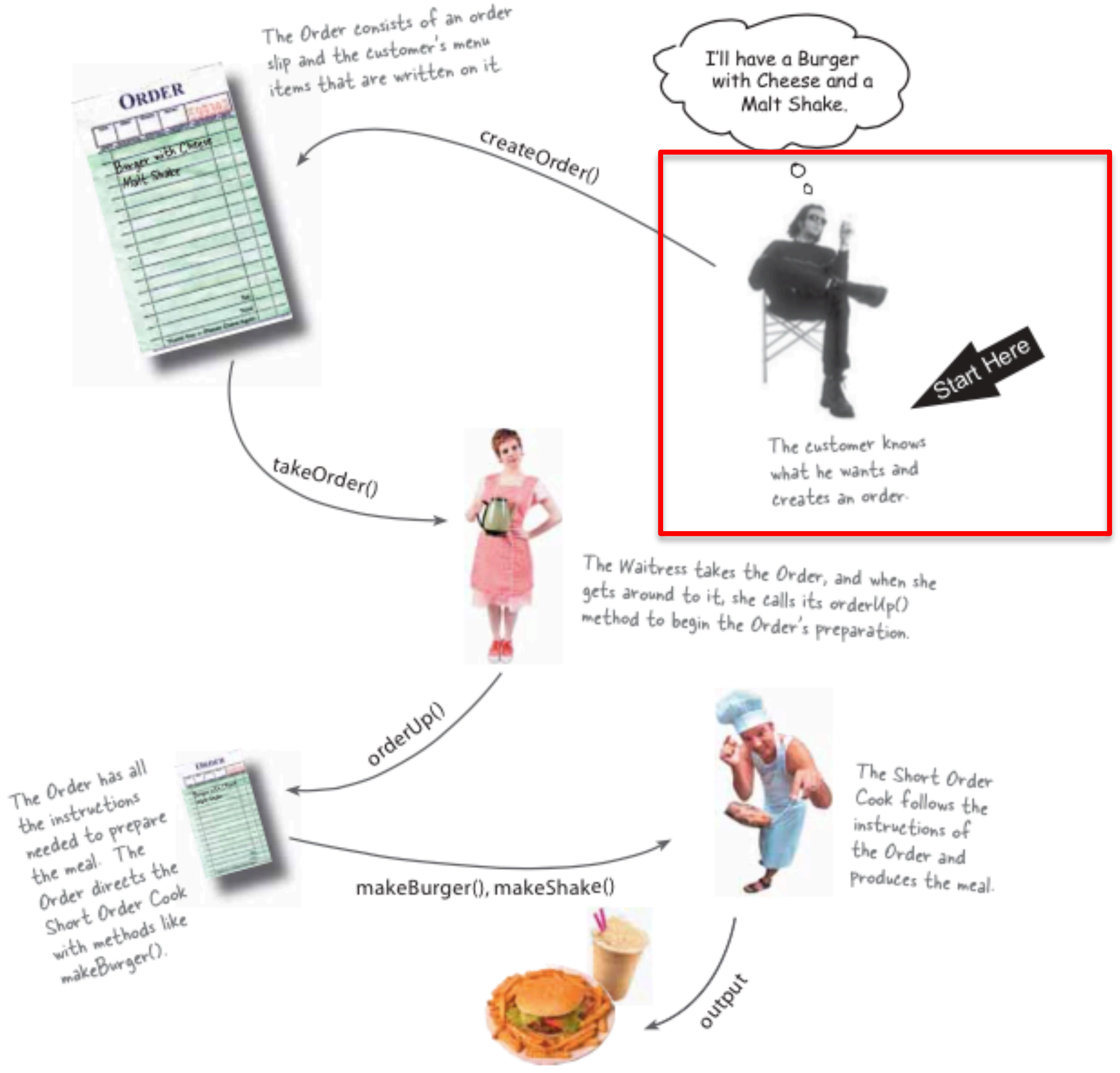
1. You, the Customer, give the Waitress your Order.
2. The Waitress takes the Order, places it on the order counter and says “Order up!”
3. The Short-Order Cook prepares your meal from the Order.

Let us look a
this a little
closer…


The Order consists of an order slip and the customer's menu items that are written on it.
ORDER
Burger with Cheese
Malt Shake
I'll have a Burger with Cheese and a Malt Shake.
createOrder()
Start Here
The customer knows what he wants and creates an order.
takeOrder()
The Waitress takes the Order, and when she gets around to it, she calls its orderUp() method to begin the Order's preparation.
orderUp()
The Order has all the instructions needed to prepare the meal. The Order directs the Short Order Cook with methods like makeBurger().
makeBurger(), makeShake()
The Short Order Cook follows the instructions of the Order and produces the meal.
output

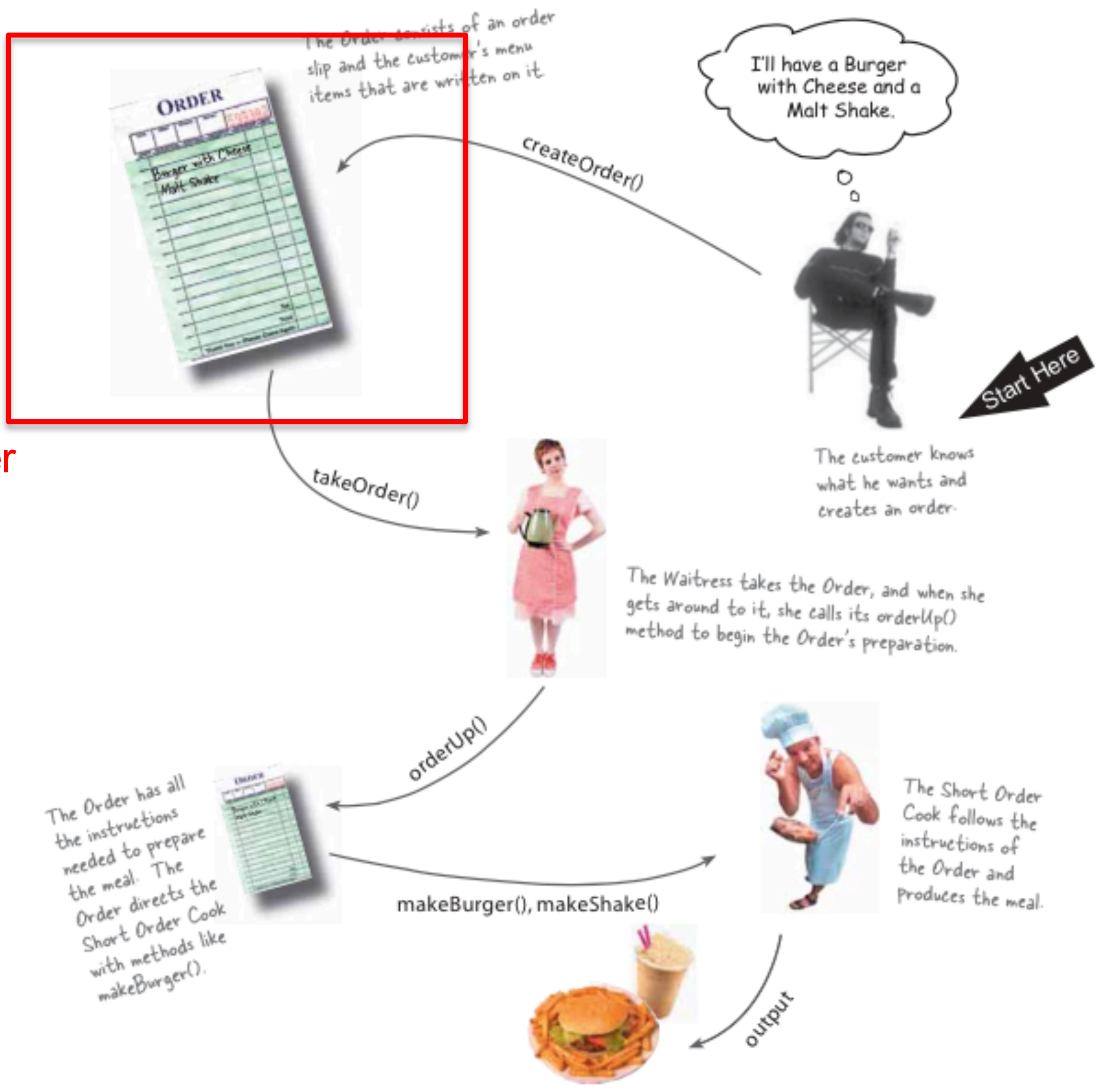
Let us look a this a little closer…

The customer knows what he wants and creates an order.


The Order consists of an order slip and the customer's menu items that are written on it.
I'll have a Burger with Cheese and a Malt Shake.
createOrder()
ORDER
Burger with Cheese
Malt Shake
Start Here
The customer knows what he wants and creates an order.
takeOrder()
The Waitress takes the Order, and when she gets around to it, she calls its orderUp() method to begin the Order's preparation.
orderUp()
The Order has all the instructions needed to prepare the meal. The Order directs the Short Order Cook with methods like makeBurger().
makeBurger(), makeShake()
The Short Order Cook follows the instructions of the Order and produces the meal.
output

Let us look a this a little closer…

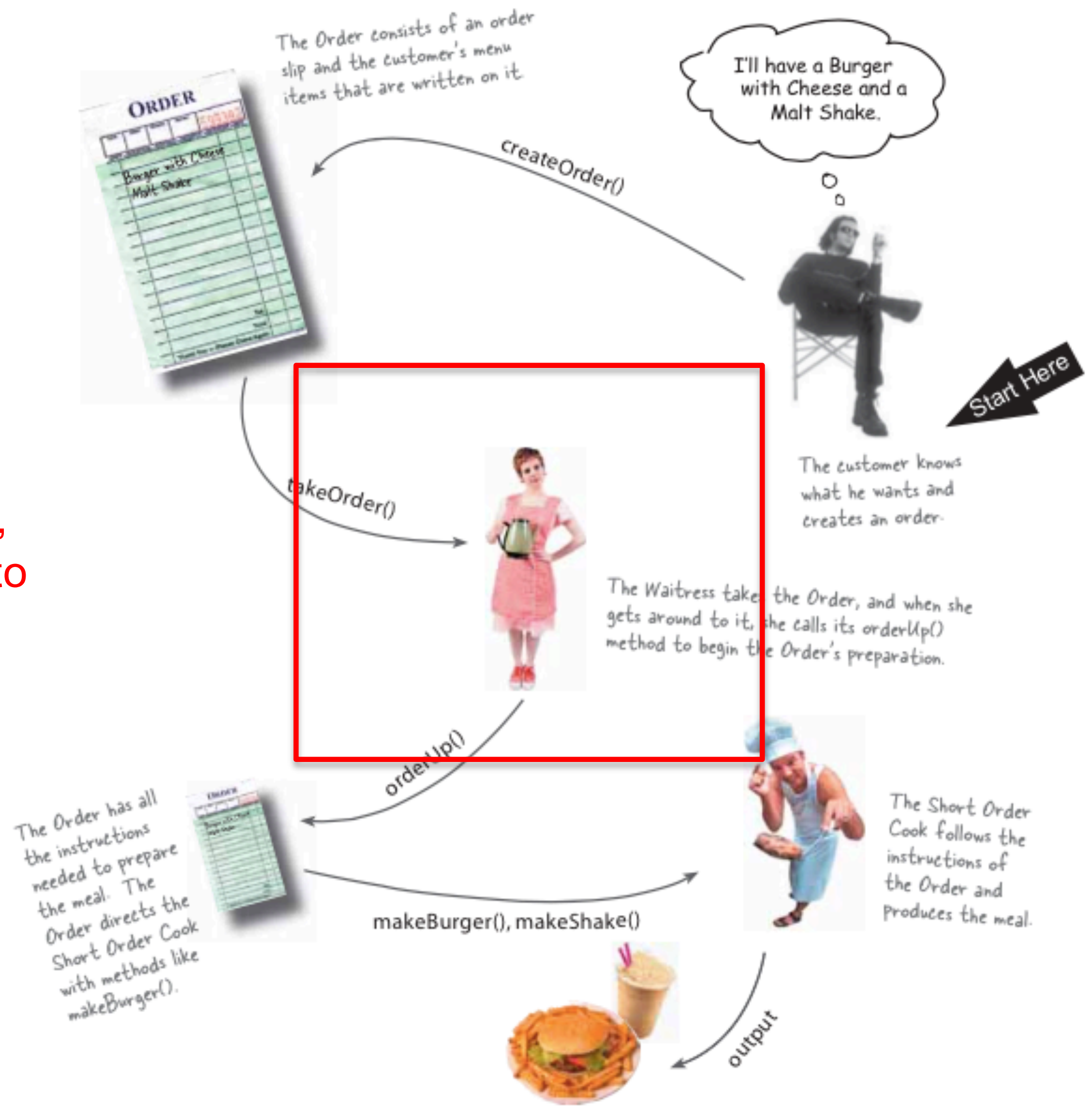
The order consists of an order slip and the customer's menu items that are written on it.



The Order consists of an order slip and the customer's menu items that are written on it.
ORDER
Burger with Cheese
Malt Shake
createOrder()
I'll have a Burger with Cheese and a Malt Shake.
Start Here
The customer knows what he wants and creates an order.
takeOrder()
The Waitress takes the Order, and when she gets around to it, she calls its orderUp() method to begin the Order's preparation.
orderUp()
The Order has all the instructions needed to prepare the meal. The Order directs the Short Order Cook with methods like makeBurger().
makeBurger(), makeShake()
The Short Order Cook follows the instructions of the Order and produces the meal.
output

Let us look a this a little closer…

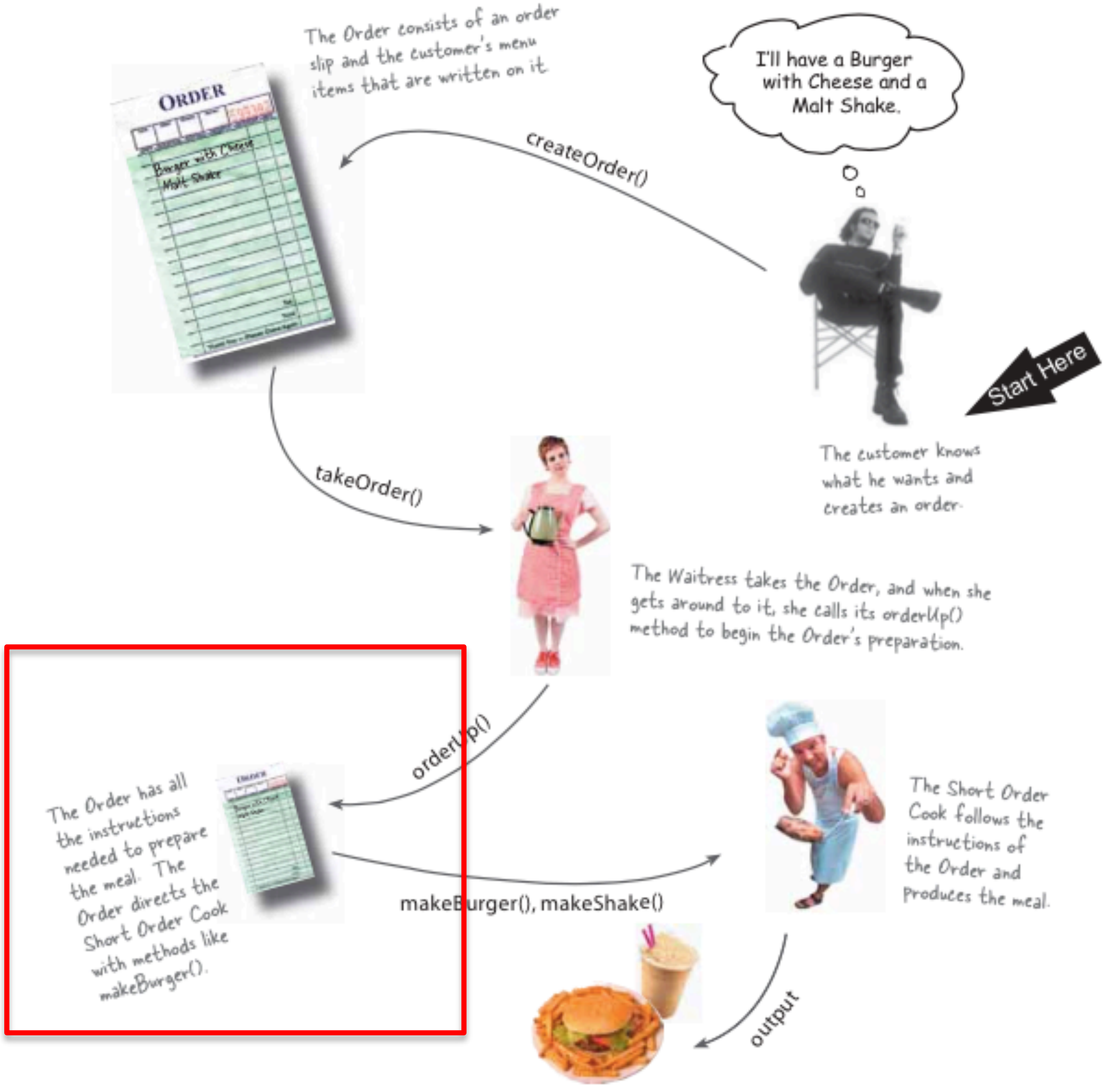
The wait staff takes the order, and when they get around to it, they call its orderUp() method to begin the Order's preparation.



The Order consists of an order slip and the customer's menu items that are written on it.
ORDER
Burger with Cheese
Malt Shake
I'll have a Burger with Cheese and a Malt Shake.
createOrder()
Start Here
The customer knows what he wants and creates an order.
takeOrder()
The Waitress takes the Order, and when she gets around to it, she calls its orderUp() method to begin the Order's preparation.
orderUp()
The Order has all the instructions needed to prepare the meal. The Order directs the Short Order Cook with methods like makeBurger().
makeBurger(), makeShake()
The Short Order Cook follows the instructions of the Order and produces the meal.
output

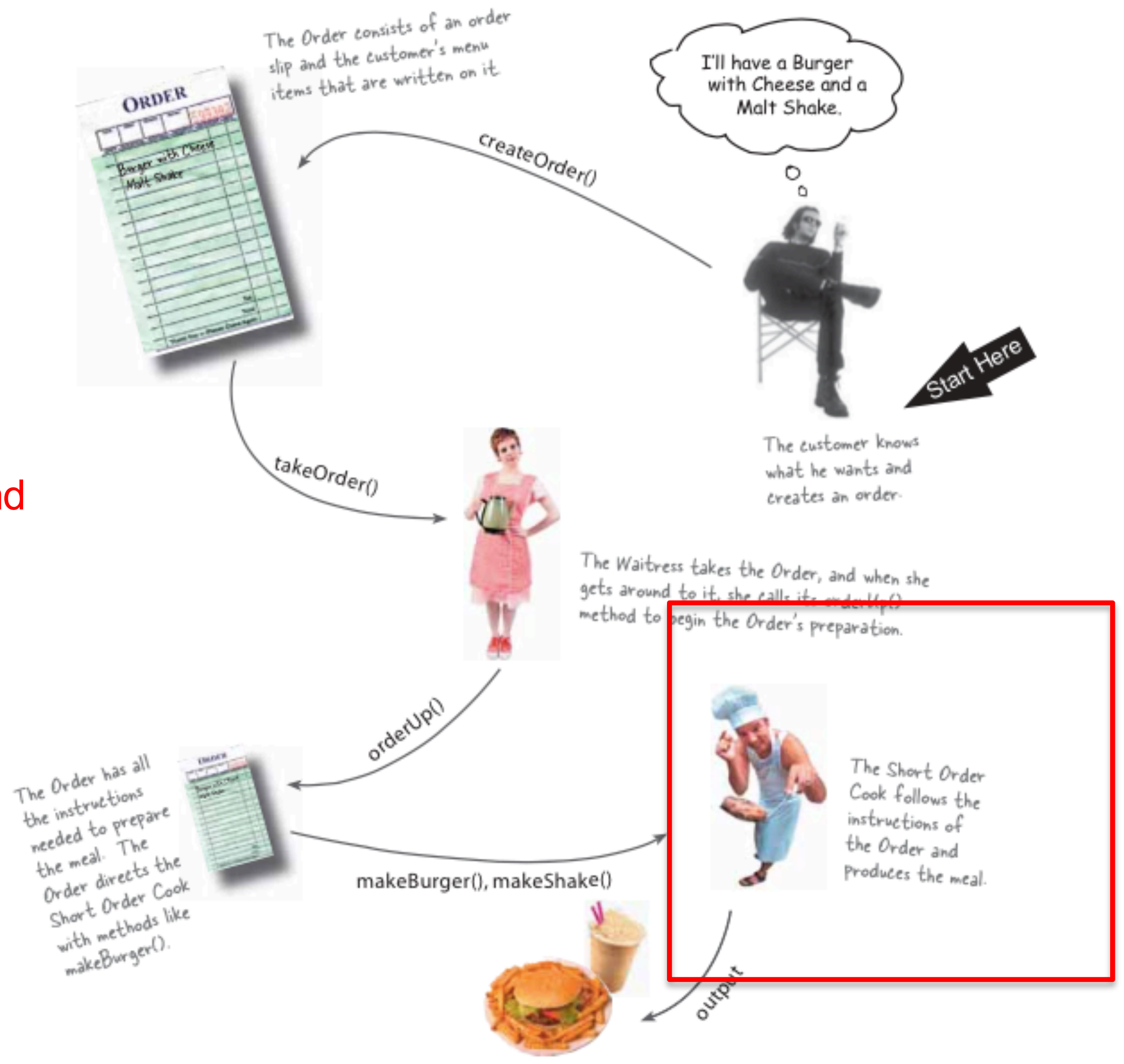
Let us look a this a little closer…

The Order has all the instructions necessary to prepare the meal. The Order directs the Short Order Cook with methods like makeBurger()


The Order consists of an order slip and the customer's menu items that are written on it.
ORDER
Burger with Cheese
Malt Shake
createOrder()
I'll have a Burger with Cheese and a Malt Shake.
Start Here
The customer knows what he wants and creates an order.
takeOrder()
The Waitress takes the Order, and when she gets around to it, she calls its orderUp() method to begin the Order's preparation.
orderUp()
The Order has all the instructions needed to prepare the meal. The Order directs the Short Order Cook with methods like makeBurger().
makeBurger(), makeShake()
The Short Order Cook follows the instructions of the Order and produces the meal.
output

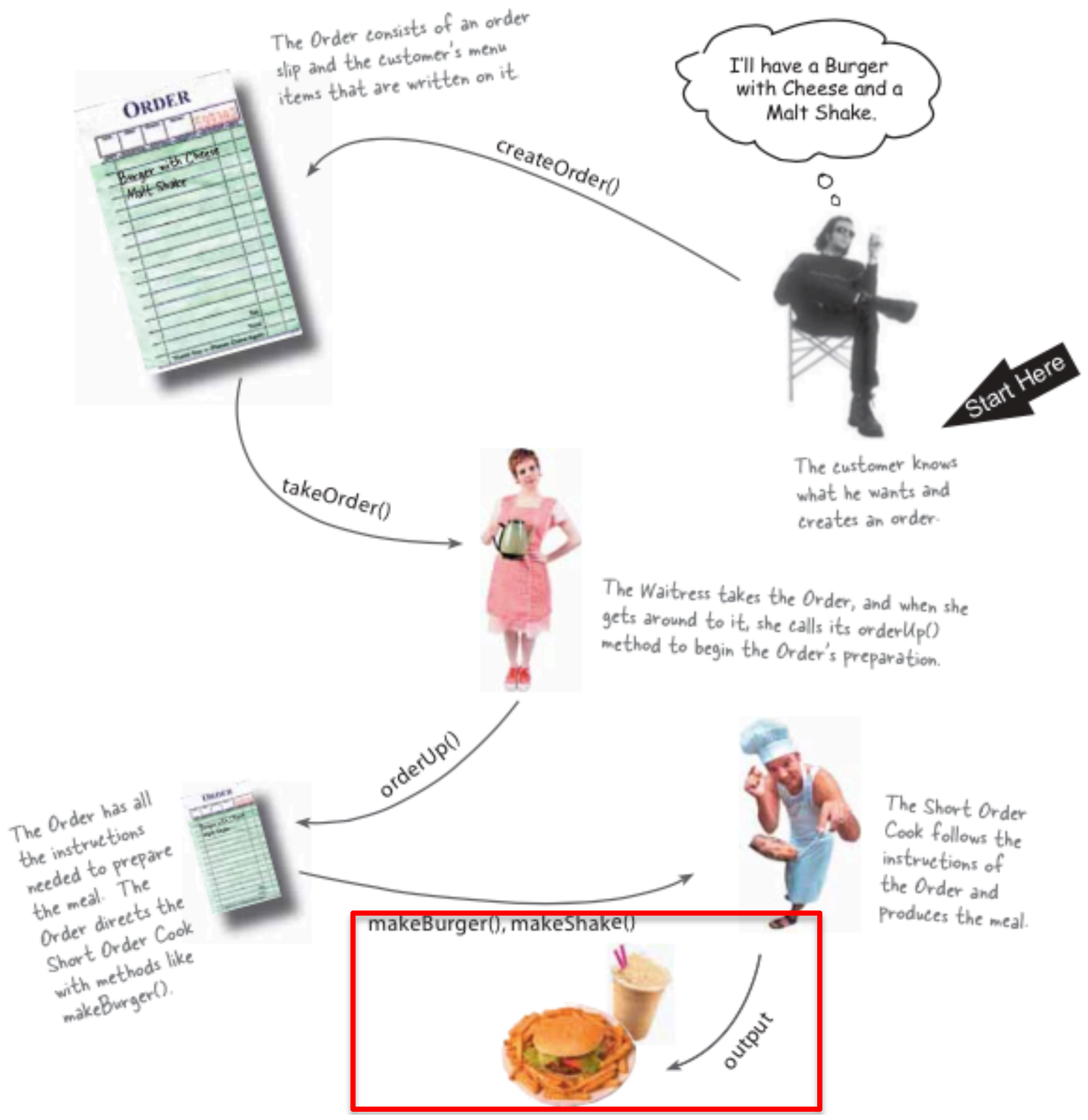
Let us look a this a little closer…

The Short Order cook follows the instructions of the Order and produces the meal.

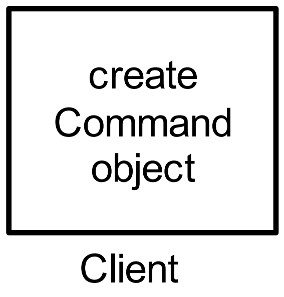

The Order consists of an order slip and the customer's menu items that are written on it.
I'll have a Burger with Cheese and a Malt Shake.
createOrder()
Start Here
The customer knows what he wants and creates an order.
takeOrder()
The Waitress takes the Order, and when she gets around to it, she calls its orderUp() method to begin the Order's preparation.
orderUp()
The Order has all the instructions needed to prepare the meal. The Order directs the Short Order Cook with methods like makeBurger().
makeBurger(), makeShake()
The Short Order Cook follows the instructions of the Order and produces the meal.
output

Let us look a this a little closer…

Finally, we have the meal as output.

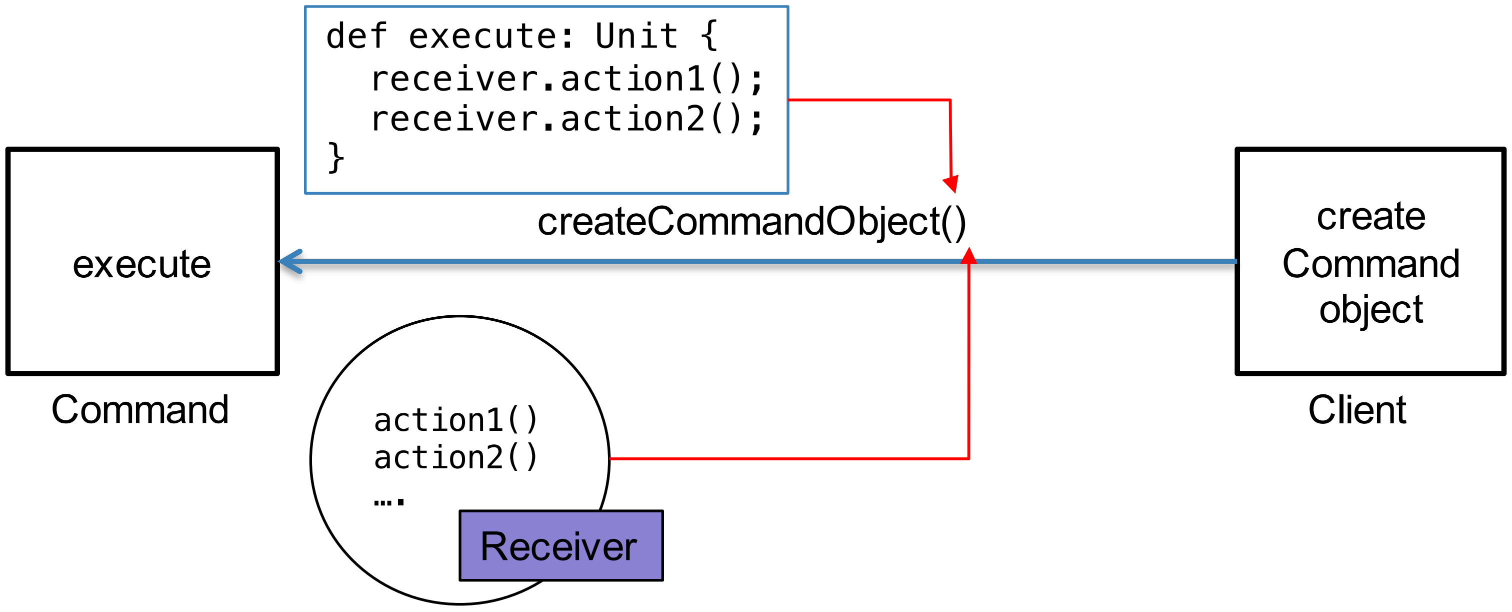

The Order consists of an order slip and the customer's menu items that are written on it.
ORDER
Burger with Cheese
Malt Shake
I'll have a Burger with Cheese and a Malt Shake.
createOrder()
Start Here
The customer knows what he wants and creates an order.
takeOrder()
The Waitress takes the Order, and when she gets around to it, she calls its orderUp() method to begin the Order's preparation.
orderUp()
The Order has all the instructions needed to prepare the meal. The Order directs the Short Order Cook with methods like makeBurger().
The Short Order Cook follows the instructions of the Order and produces the meal.
makeBurger(), makeShake()
output

From Diner to Command Pattern



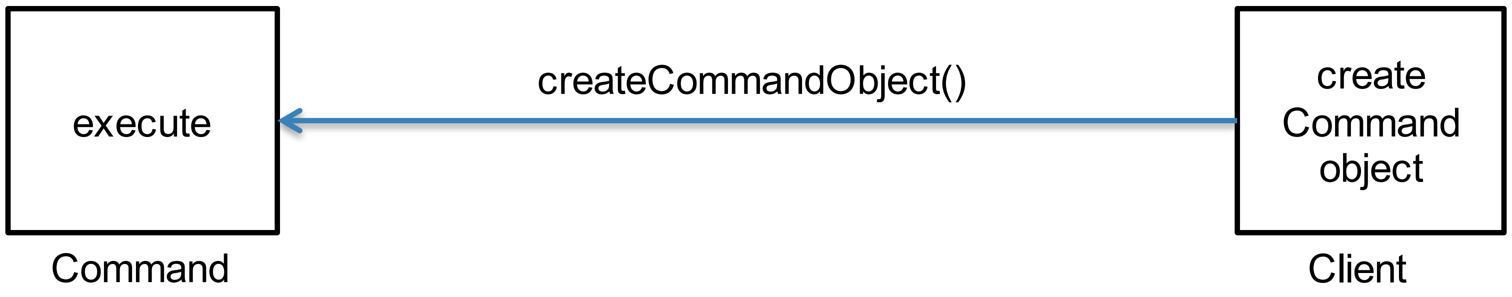
The client is responsible for creating the command object. The command object consists of a set of actions on a receiver.

From Diner to Command Pattern


```
def execute: Unit {
  receiver.action1();
  receiver.action2();
}
```

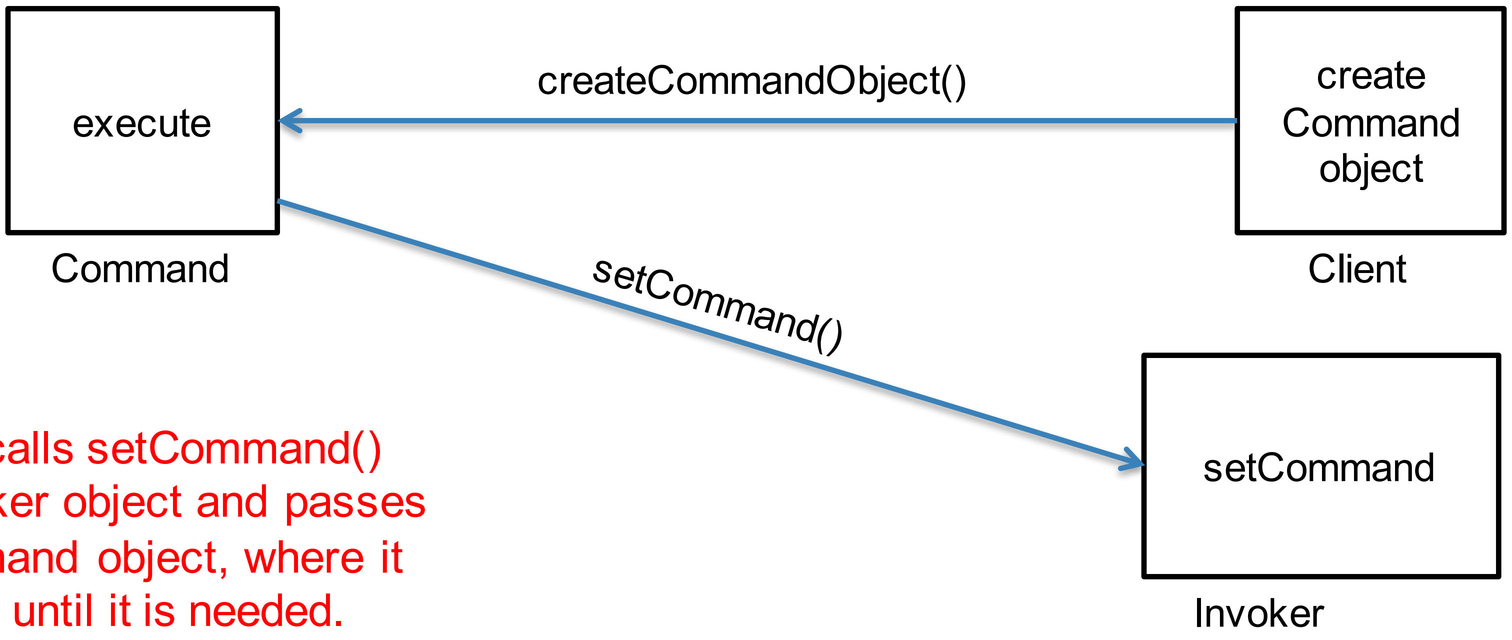
The actions and the Receiver are bound together in the Command object

From Diner to Command Pattern



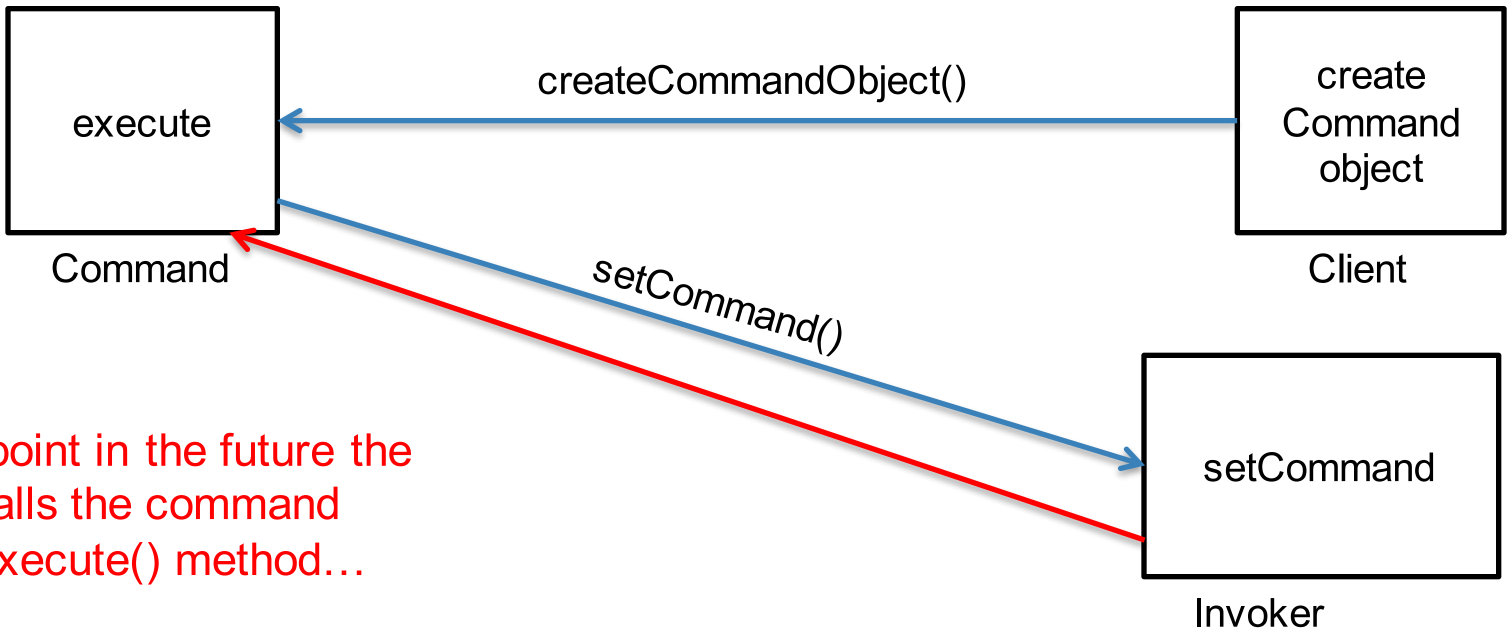
The Command object provides one method, execute(), that encapsulates the actions and can be called to invoke the actions on the Receiver.

From Diner to Command Pattern



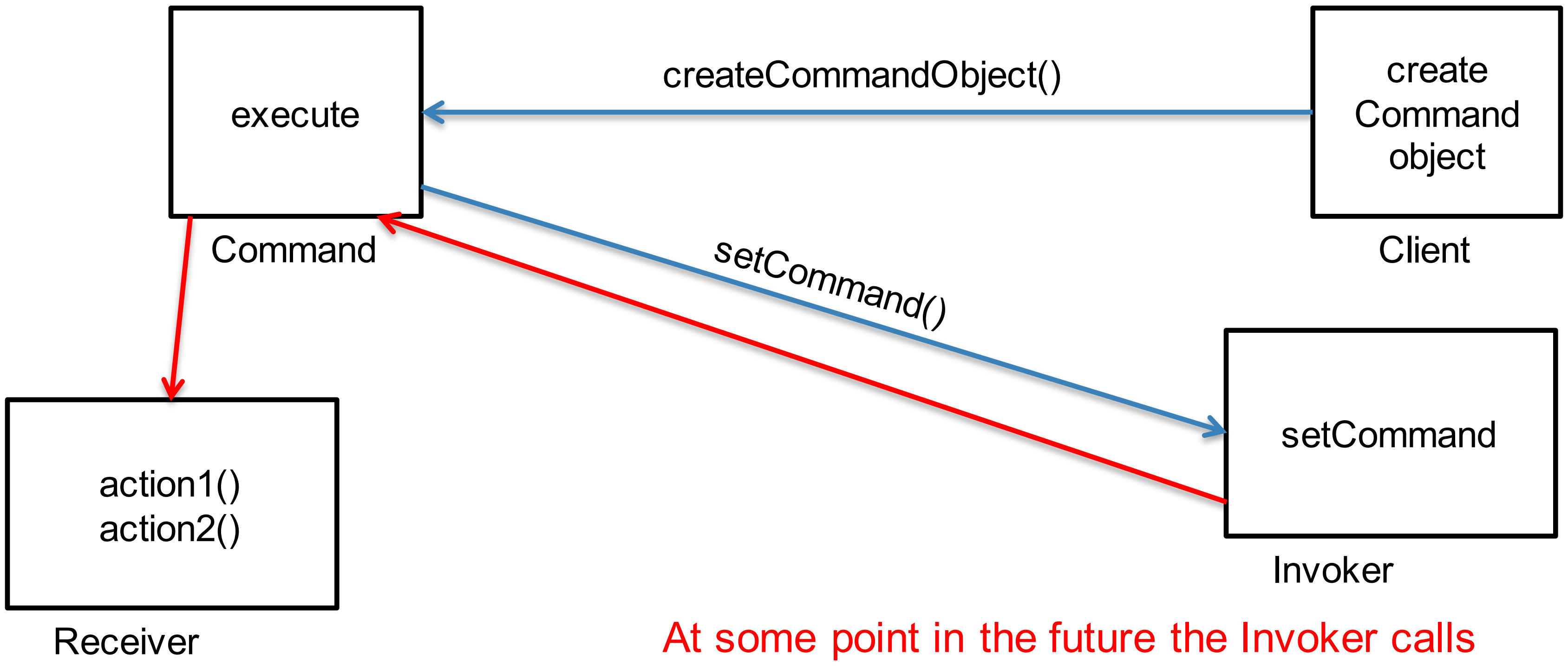
The client calls setCommand() on an Invoker object and passes it the command object, where it gets stored until it is needed.

From Diner to Command Pattern



At some point in the future the Invoker calls the command object's execute() method…

From Diner to Command Pattern

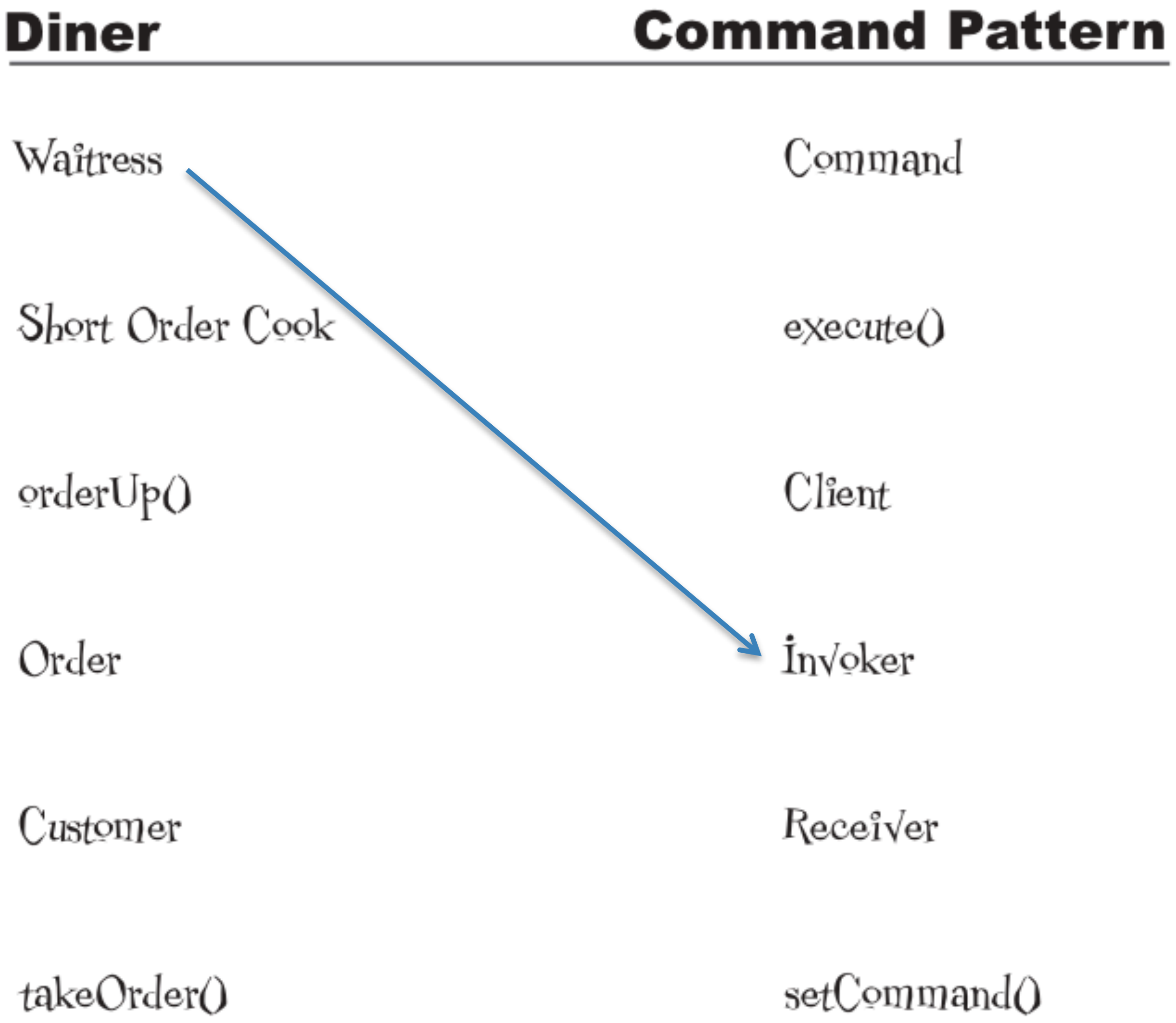


At some point in the future the Invoker calls the command object's execute() method…

What Does What?

Diner	Command Pattern
Waitress	Command
Short Order Cook	execute()
orderUp()	Client
Order	Invoker
Customer	Receiver
takeOrder()	setCommand()

What Does What?

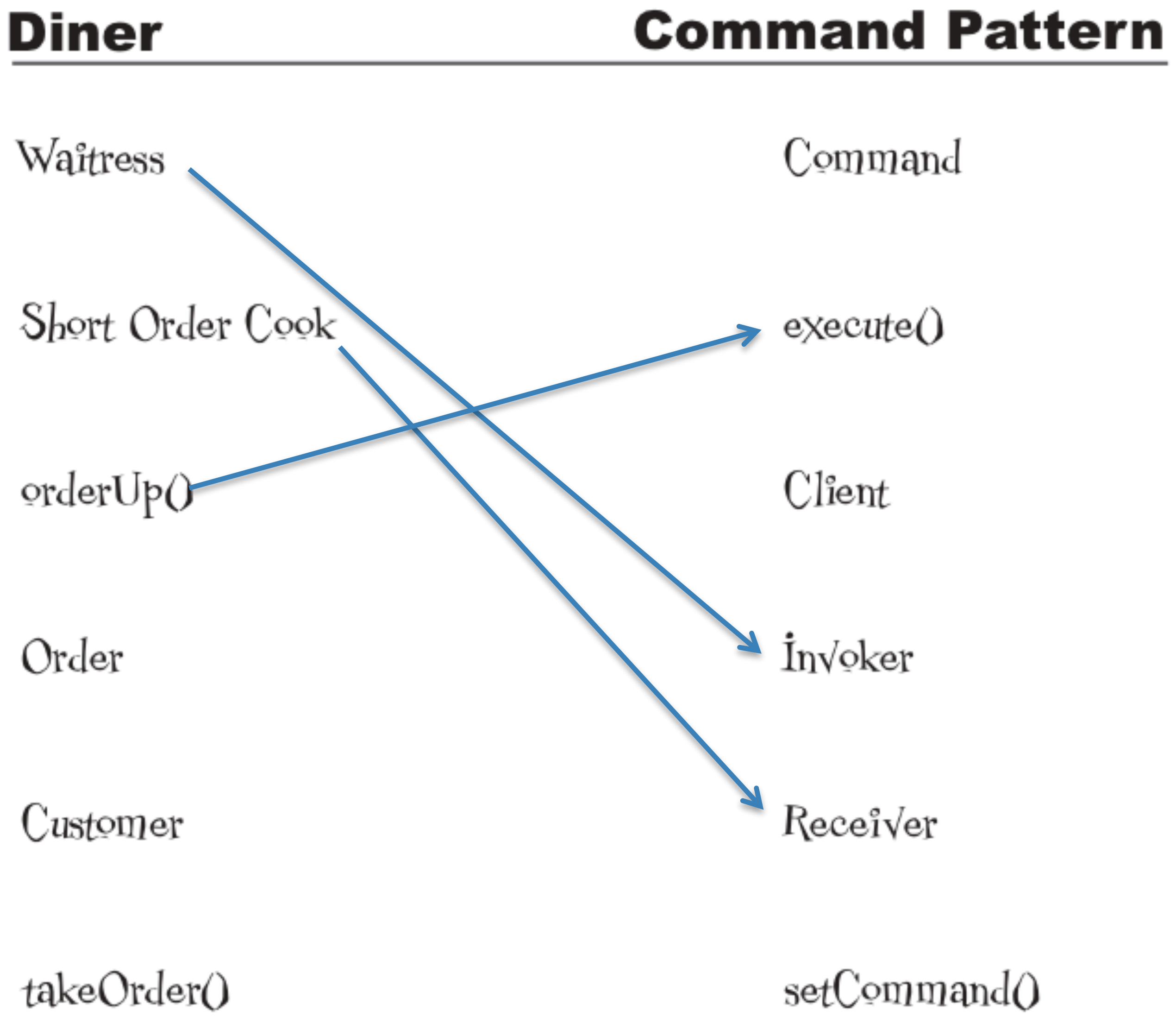


Diner	Command Pattern
Waitress	Command
Short Order Cook	execute()
orderUp()	Client
Order	Invoker
Customer	Receiver
takeOrder()	setCommand()

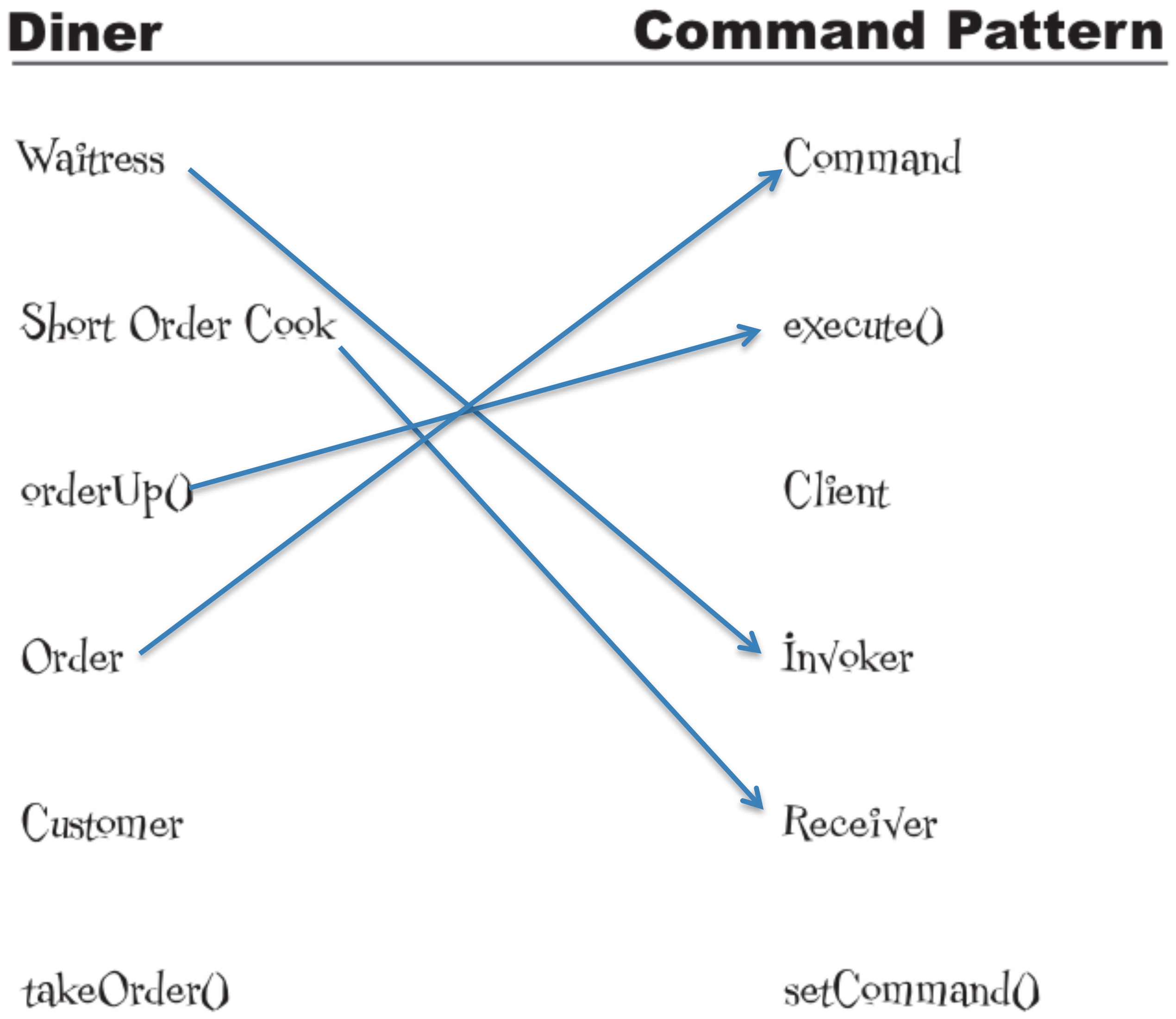
What Does What?

$$\begin{tikzcd}
\textbf{Diner} & \textbf{Command Pattern} \\
\text{Waitress} \arrow[dddr] & \text{Command} \\
\text{Short Order Cook} \arrow[dddr] & \text{execute()} \\
\text{orderUp()} & \text{Client} \\
\text{Order} & \text{Invoker} \\
\text{Customer} & \text{Receiver} \\
\text{takeOrder()} & \text{setCommand()}
\end{tikzcd}$$

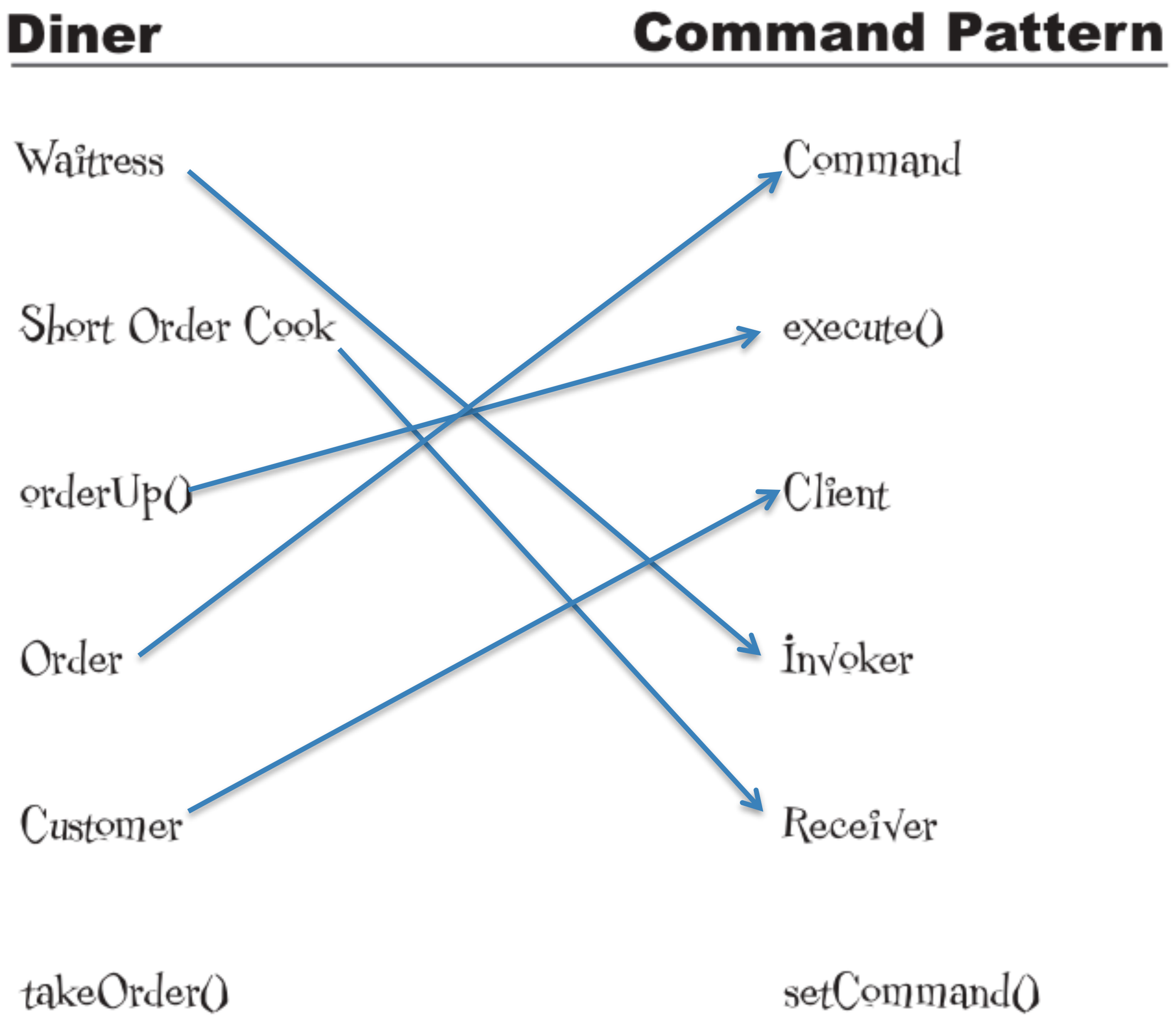
What Does What?



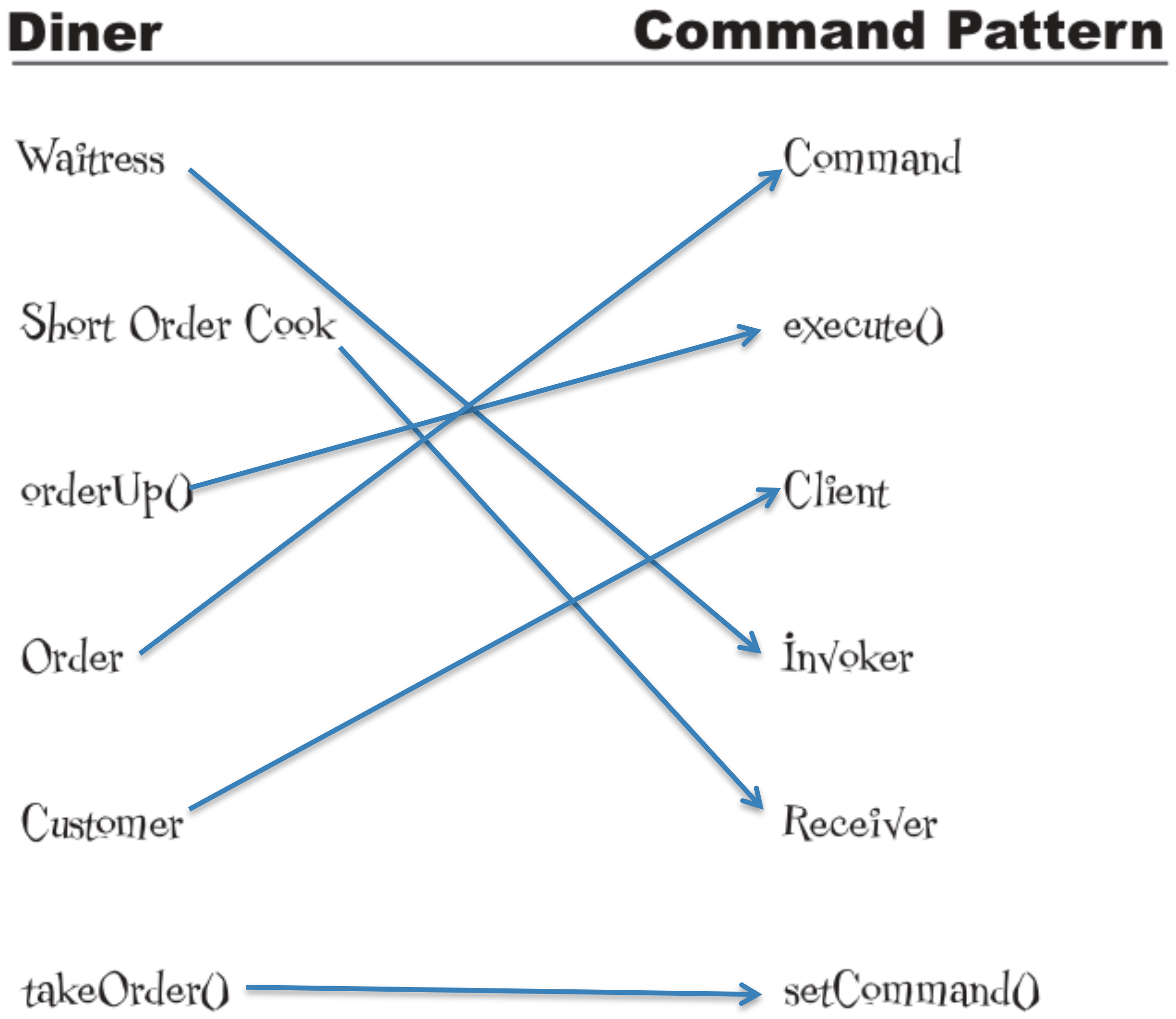
What Does What?



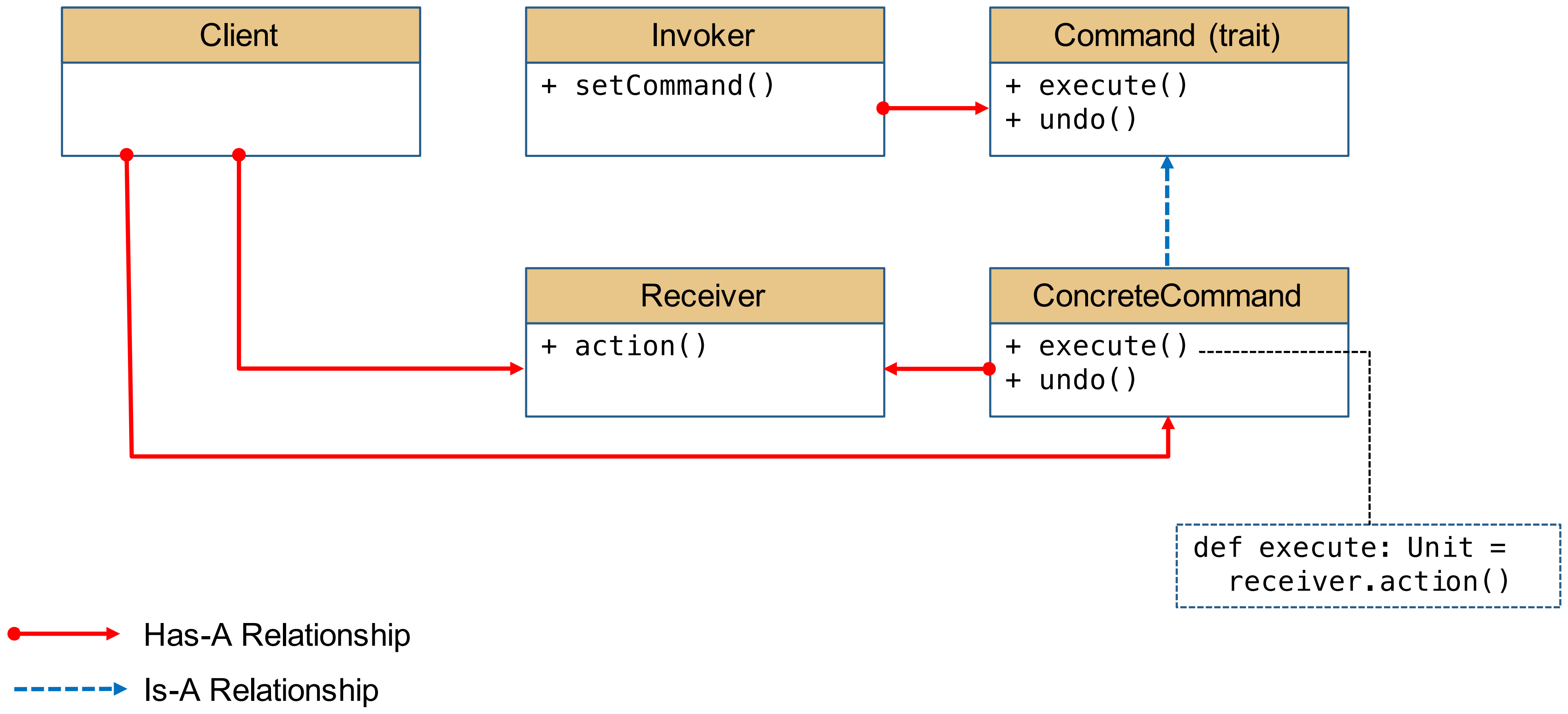
What Does What?



What Does What?



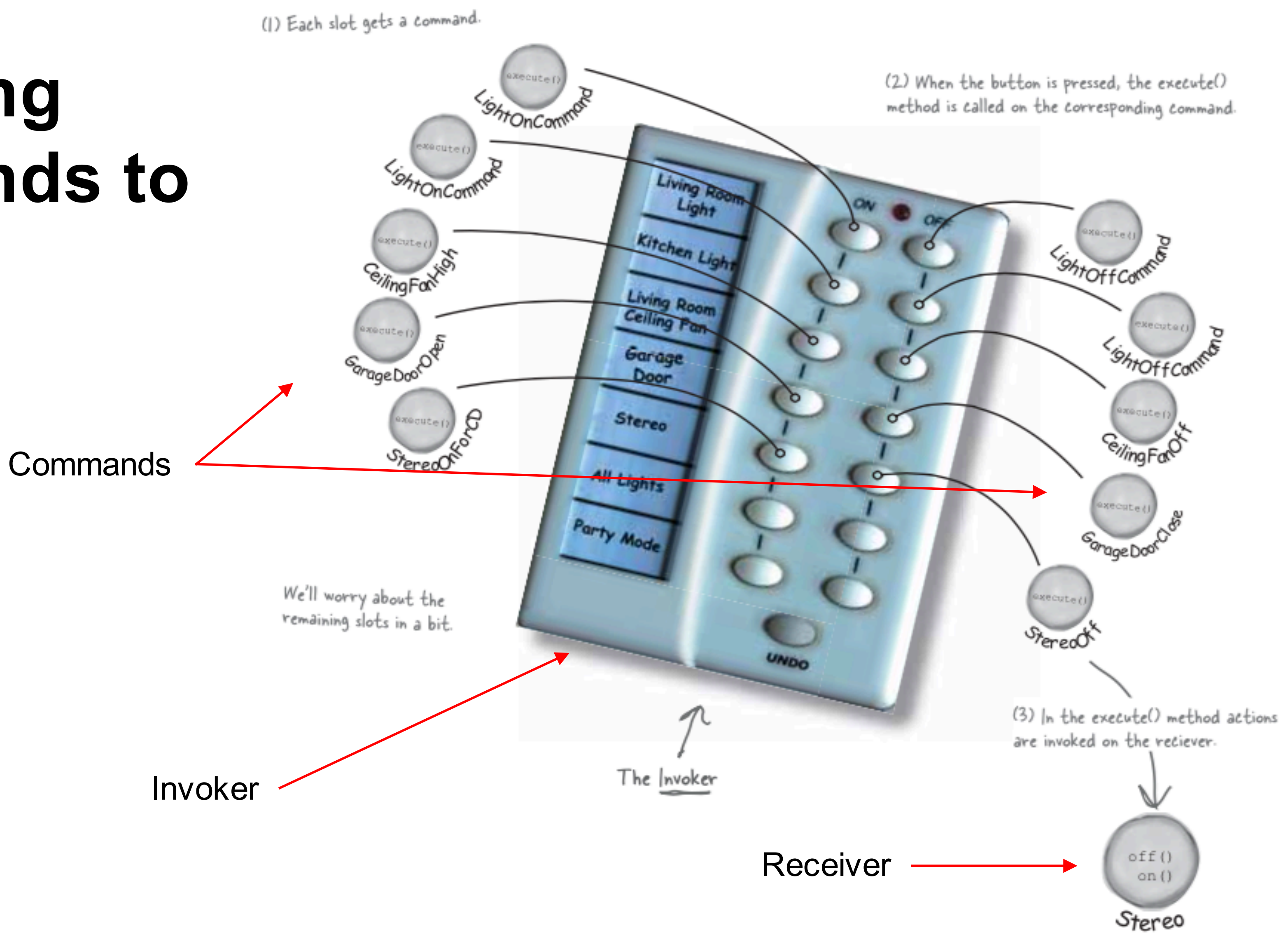
Command Pattern Class Diagram



Command Pattern Defined

The Command Pattern encapsulates a request as an object, thereby letting you parameterize other objects with different requests, queue or log requests, and support undoable operations.

Assigning Commands to Slots…



Example

- Simple
- Garage
- General
- Fan
- Undo
- Macro