Getting to Know the Behavioral Design Patterns

Esteban Herrera
JAVA ARCHITECT

@eh3rrera www.eherrera.net

Behavioral Patterns

Visitor

Strategy

Observer

Chain of Responsibility

Command

State

Mediator

Interpreter

Iterator

Memento

Template Method

Class Behavioral Patterns

Interpreter

Template Method

Object Behavioral Patterns

Visitor

Strategy

Observer

Chain of Responsibility

Command

State

Mediator

Iterator

Memento

Object Behavioral Patterns

Visitor

Strategy

Command

State

Iterator

Memento

Object Behavioral Patterns

Observer

Chain of Responsibility

Mediator

Behavioral Patterns You Should Know

Strategy and State

Command

Observer

Template method

The Strategy and State Patterns

A Strategy Is…

A plan

An approach

An algorithm

Multiples Strategies Have…

Same

- Inputs
- Outputs

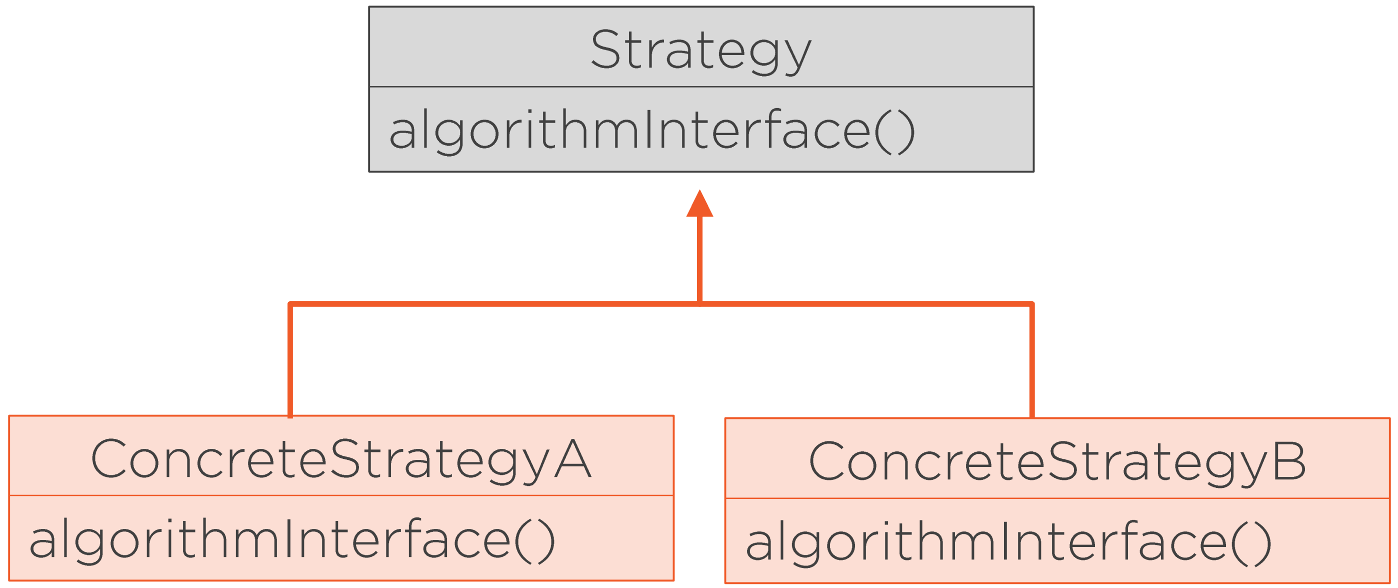
Different

- Implementations

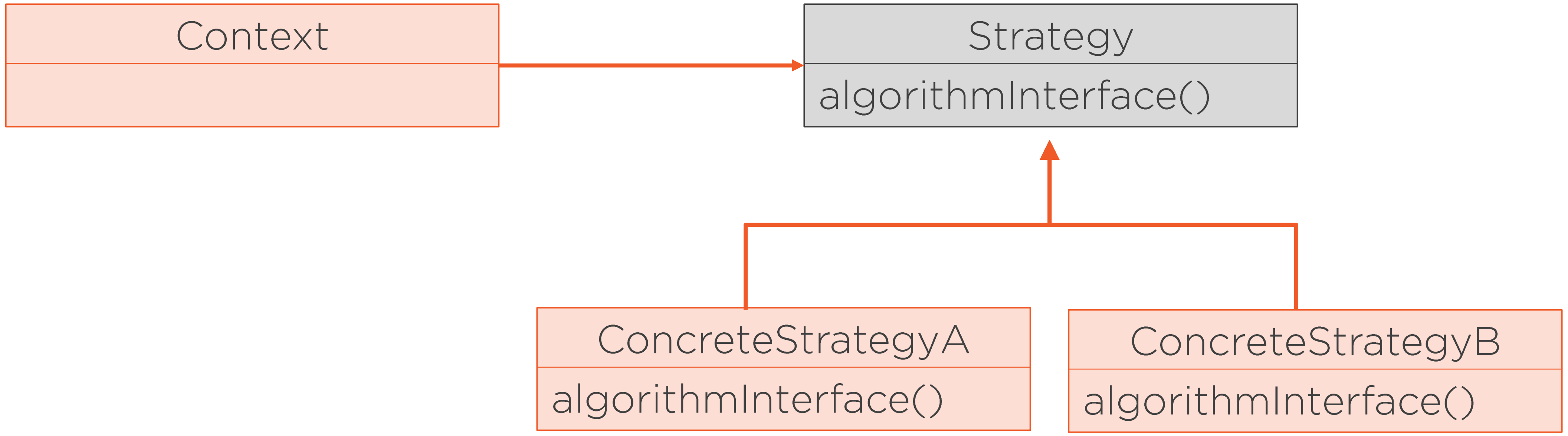
Different Strategies

```
if (isX) {
  // ...
} else if (isY) {
  // ...
} else {
  // ...
}
```

Encapsulate an Algorithm

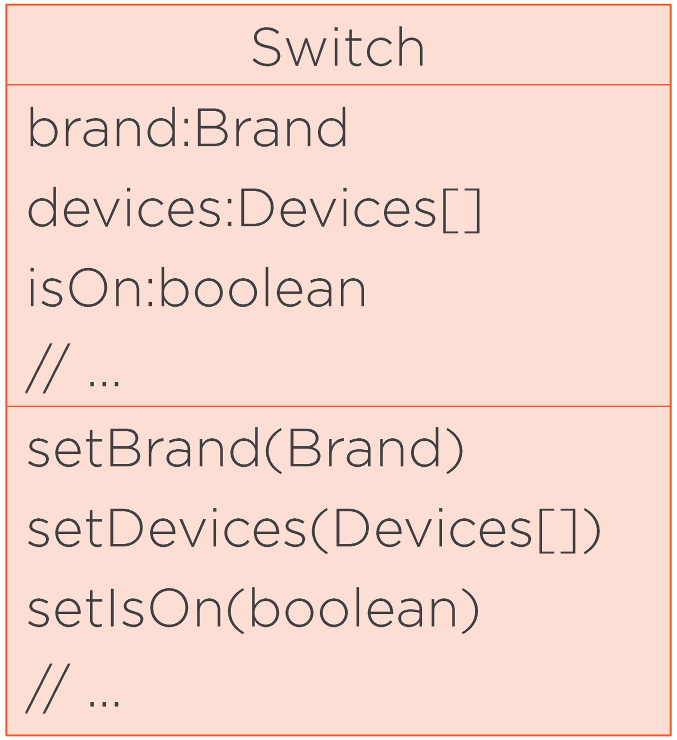


The Strategy Pattern



~~Strategy~~
State

An Object’s State



```
Switch
---------------------------
brand:Brand
devices:Devices[]
isOn:boolean
// ...
---------------------------
setBrand(Brand)
setDevices(Devices[])
setIsOn(boolean)
// ...
```

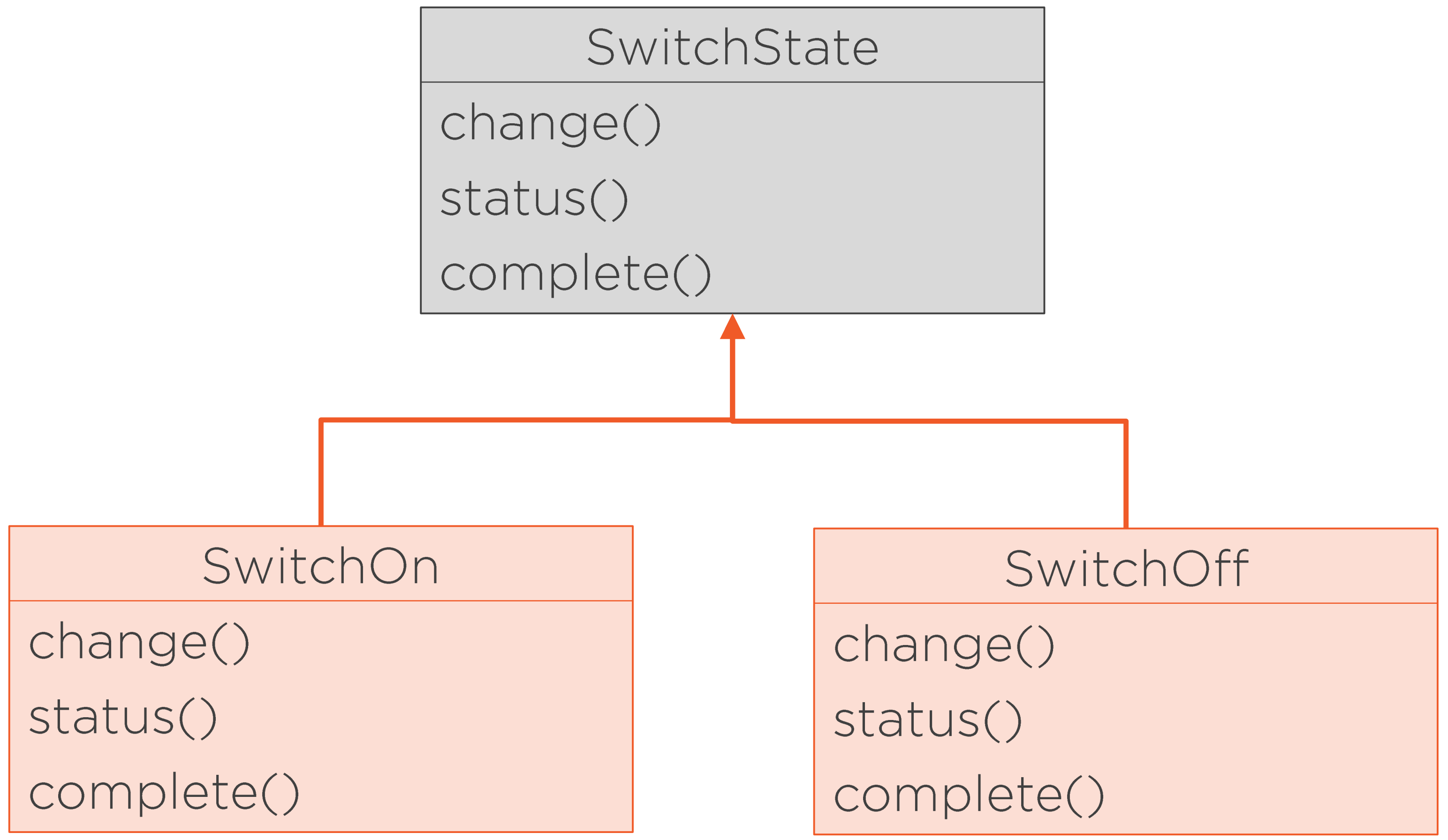
An Object's State

An Object's State

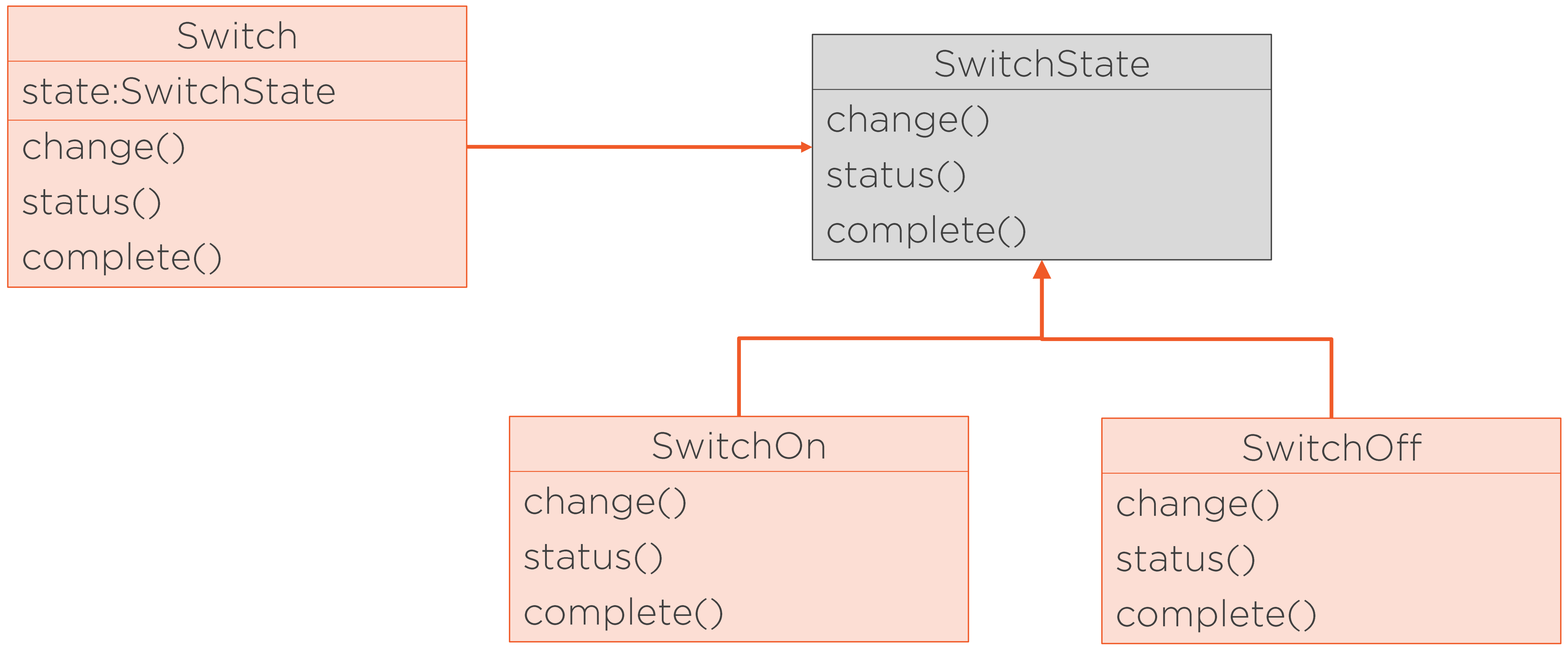
Different States

```
if (isStateX) {
  // ...
} else if (isStateY) {
  // ...
} else {
  // ...
}
```

Encapsulating State-specific Logic



The State Pattern



The Command Pattern

Calling a Method

```
anObject.myMethod(argument);
```

Calling a Method

```
menu.openFile("notes.txt");
```

Parameterizing a Method Call

```
menu.executeAction();
```

Parameterizing a Method Call

```
menu.executeAction(openFileCommand);
```

Parameterizing a Method Call

```
void executeAction(Command command) {
    command.execute();
}
```

Parameterizing a Method Call

```
menu.executeAction(new Command() {
   void execute() {
     openFile("notes.txt");
   }
});
```

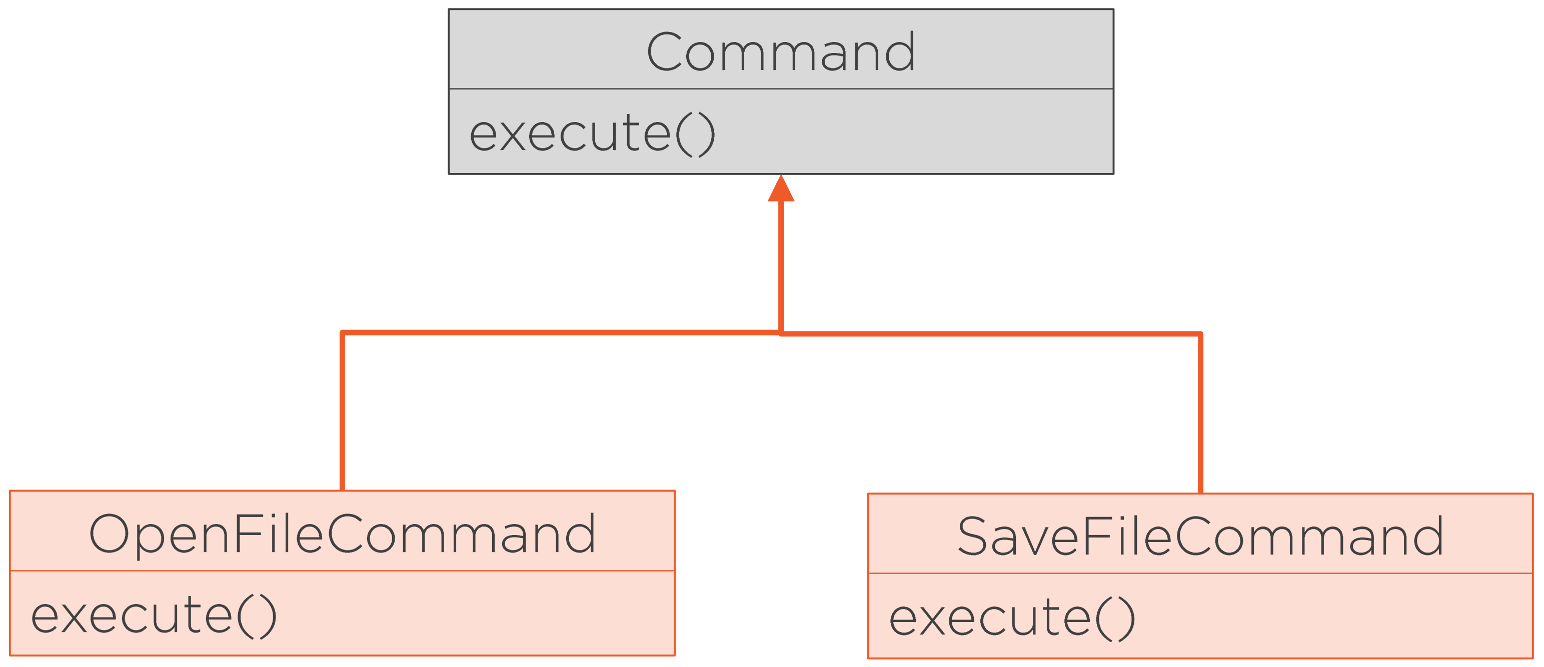
Parameterizing a Method Call

```
menu.executeAction(() -> openFile("notes.txt"));
```

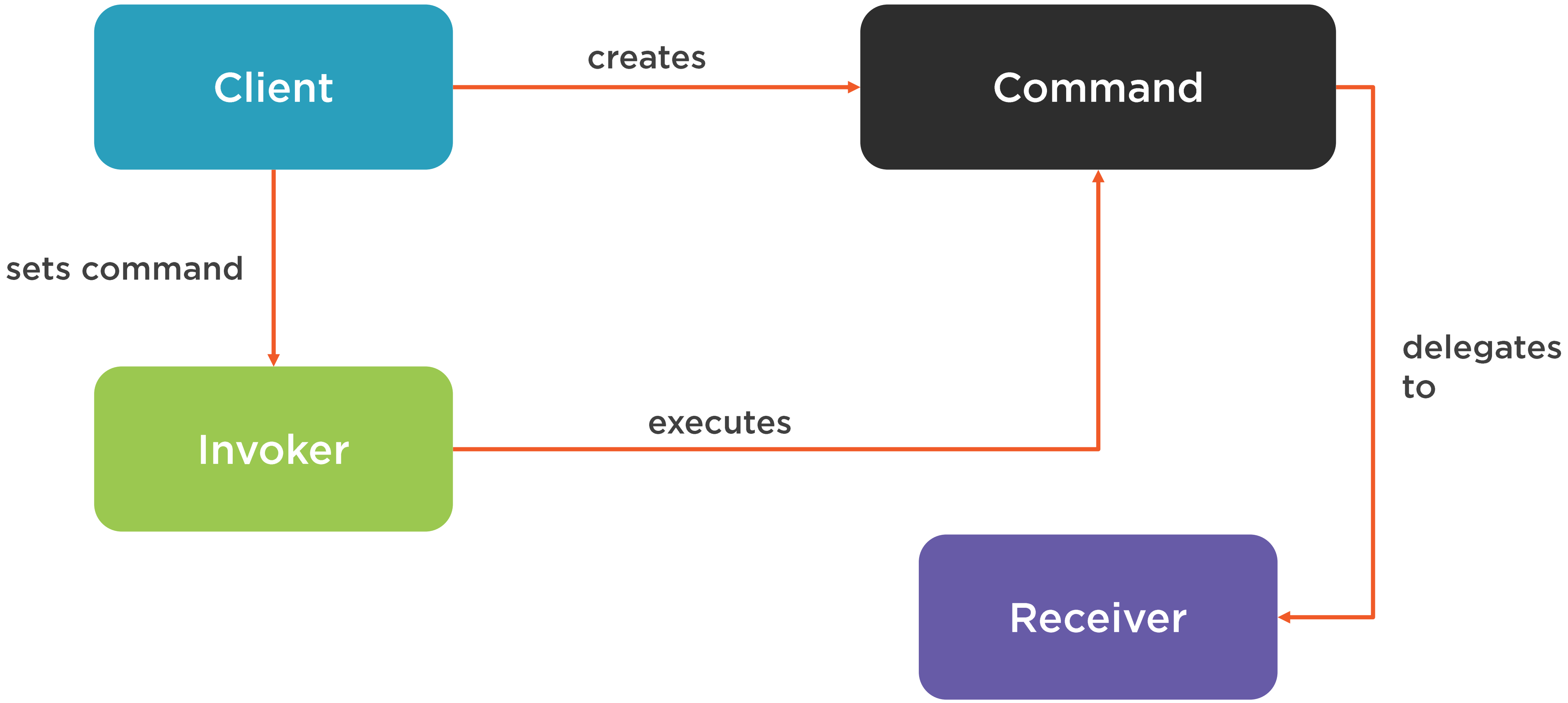
Parameterizing a Method Call

```
menu.executeAction(() -> openFile("notes.txt"));
menu.executeAction(() -> saveFile("notes.txt"));
```

Encapsulating Commands



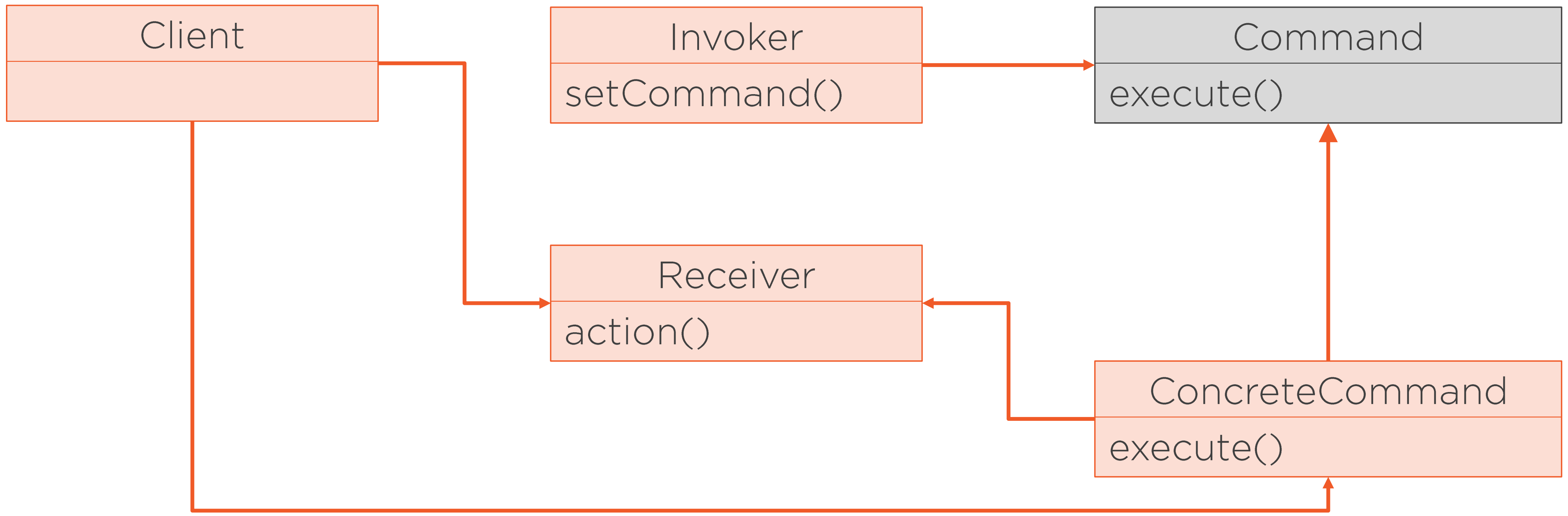
Object Relationships in the Command Pattern



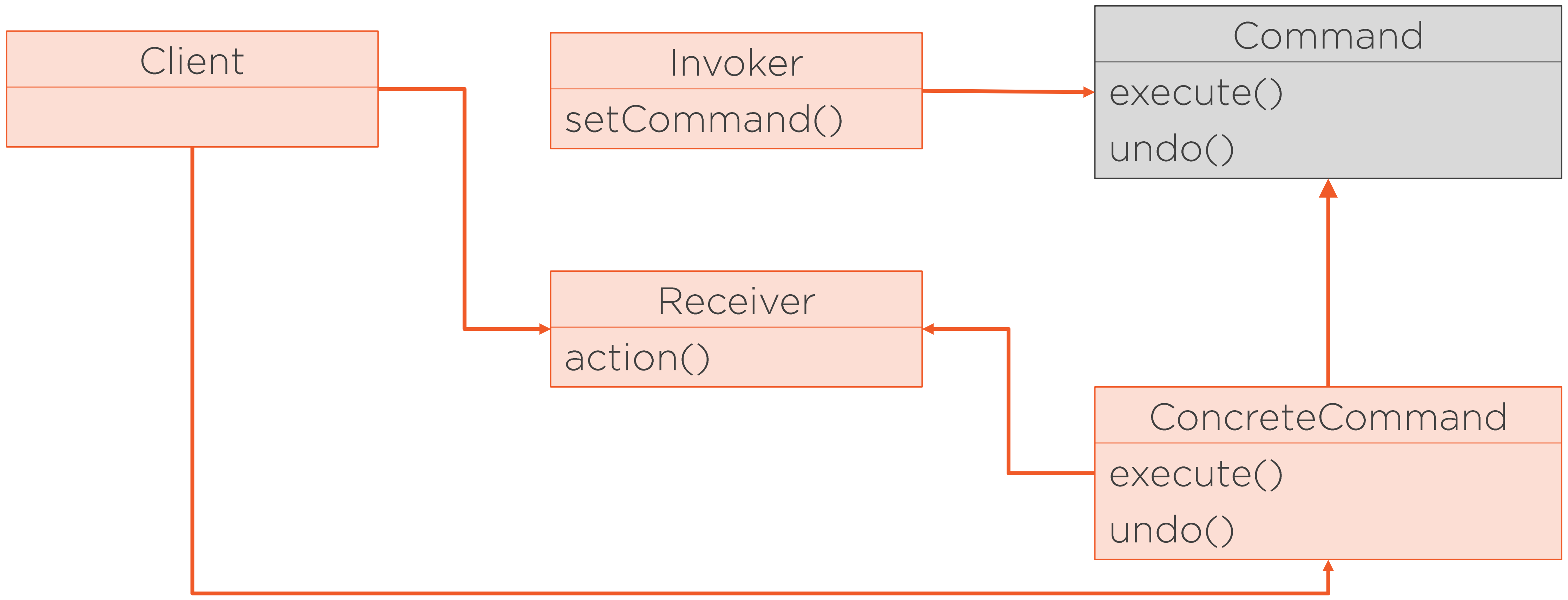
An Encapsulated Request

```
class SaveFileCommand implements Command {
    private File myFile; //The receiver

    public void execute() {
        myFile.save();
    }
}
```

The Command Pattern

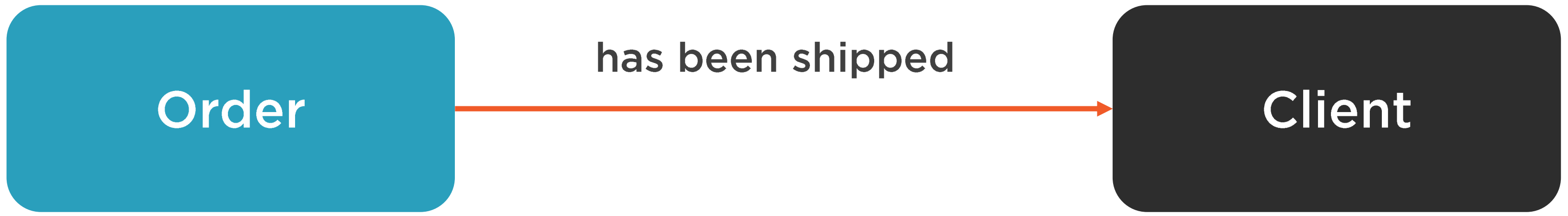


The Command Pattern



The Observer Pattern

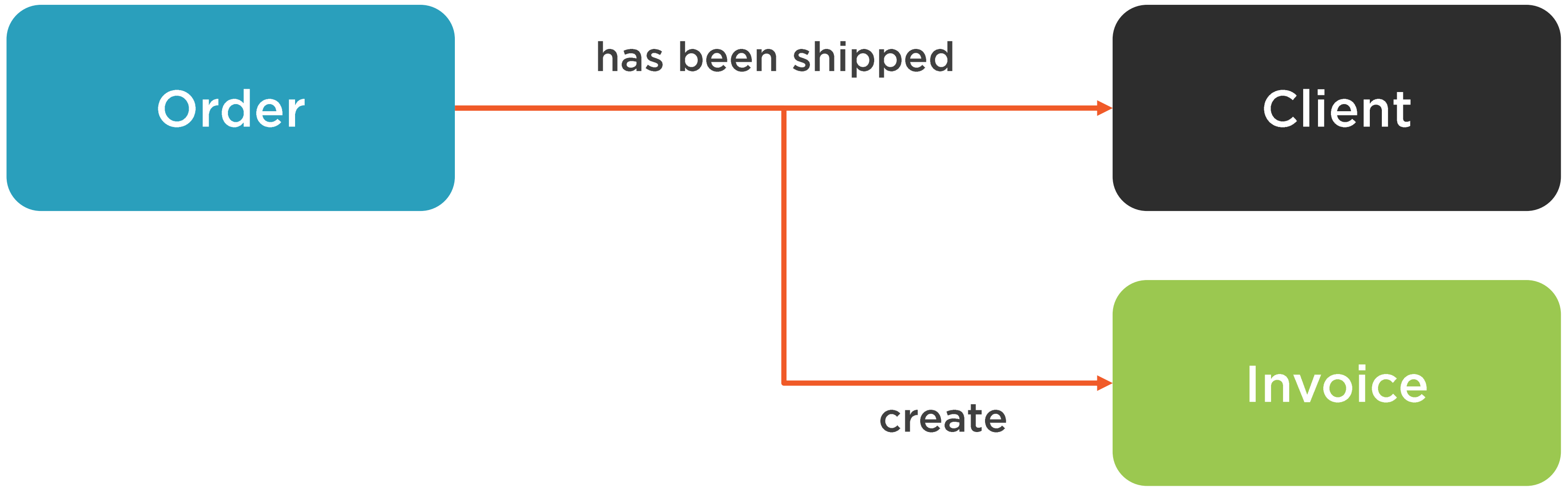
Notifying Events



Notifying Events

```
public class Order {
    // ...

    public void shipOrder() {
        // ...
        sendEmail(client);
    }
}
```

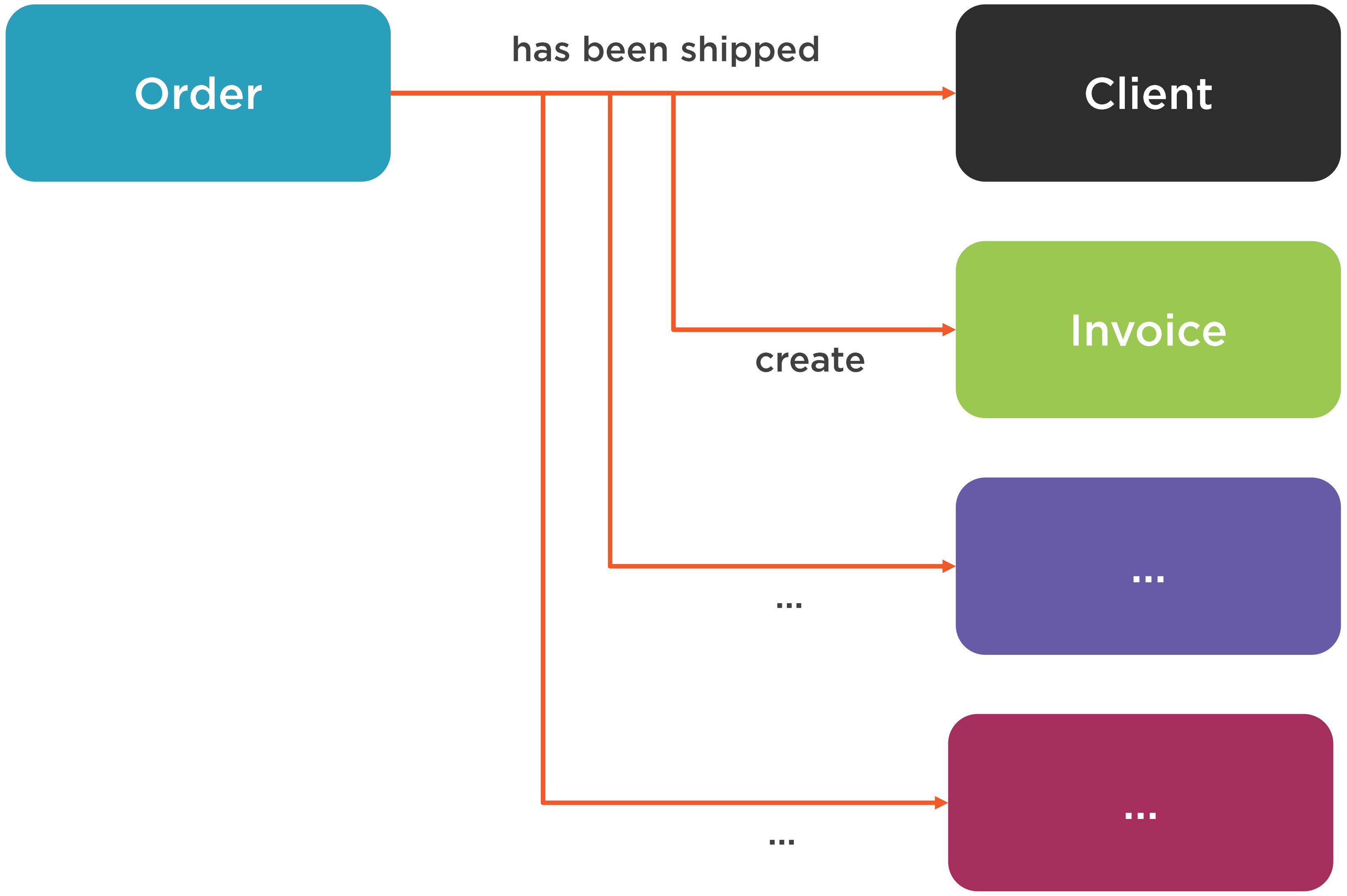
Notifying Events



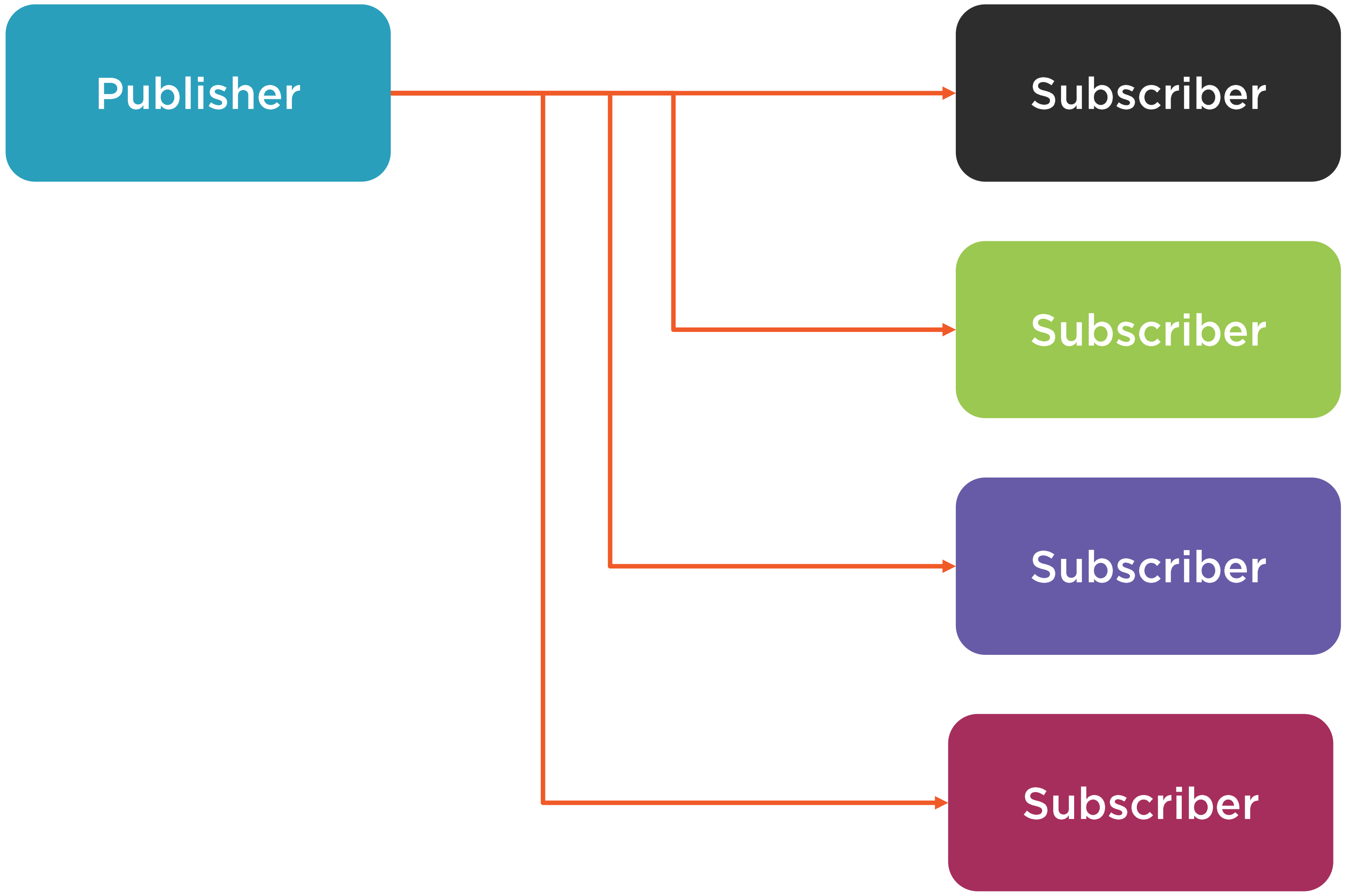
Notifying Events

```
public class Order {
    // ...

    public void shipOrder() {
        // ...
        sendEmail(client);
        invoice.create();
    }
}
```

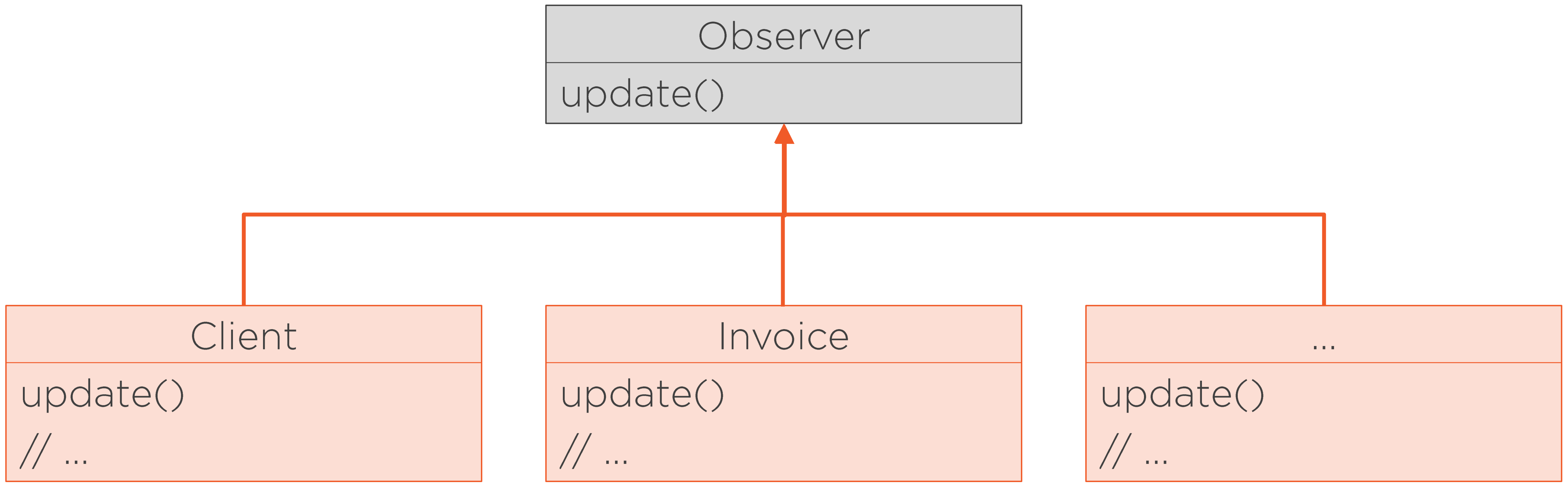
Notifying Events



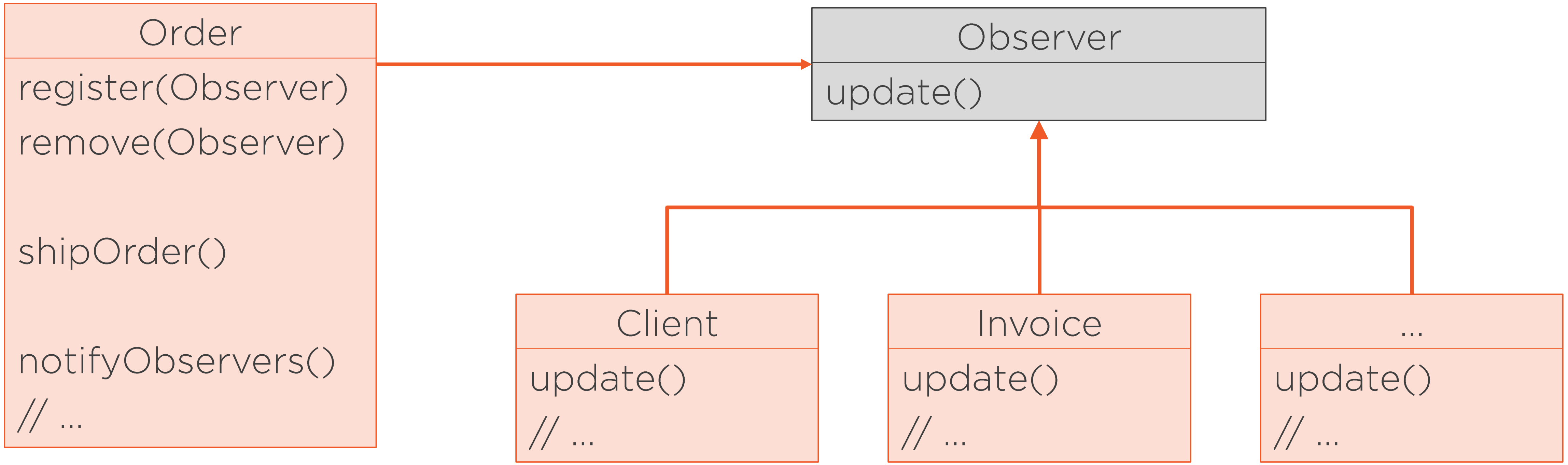
Notifying Events



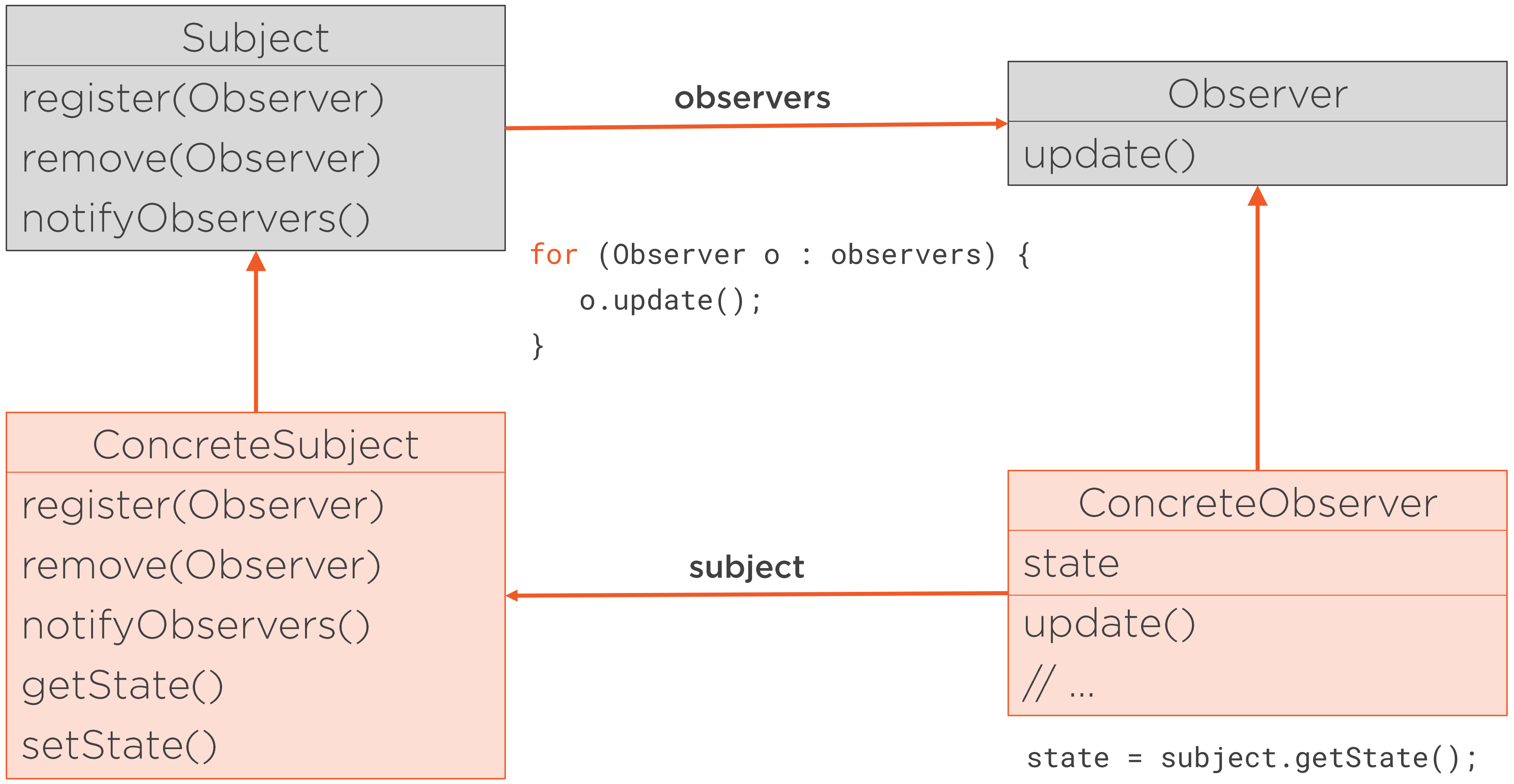
As Other Patterns Do…



A More Flexible Design



The Observer Pattern



```
for (Observer o : observers) {
   o.update();
}
```

```
state = subject.getState();
```

The Template Method Pattern

The Structure of a Method

```
myMethod() {
   // Fist statement/step
   // Second statement/step
   // ...
   // Last statement
}
```

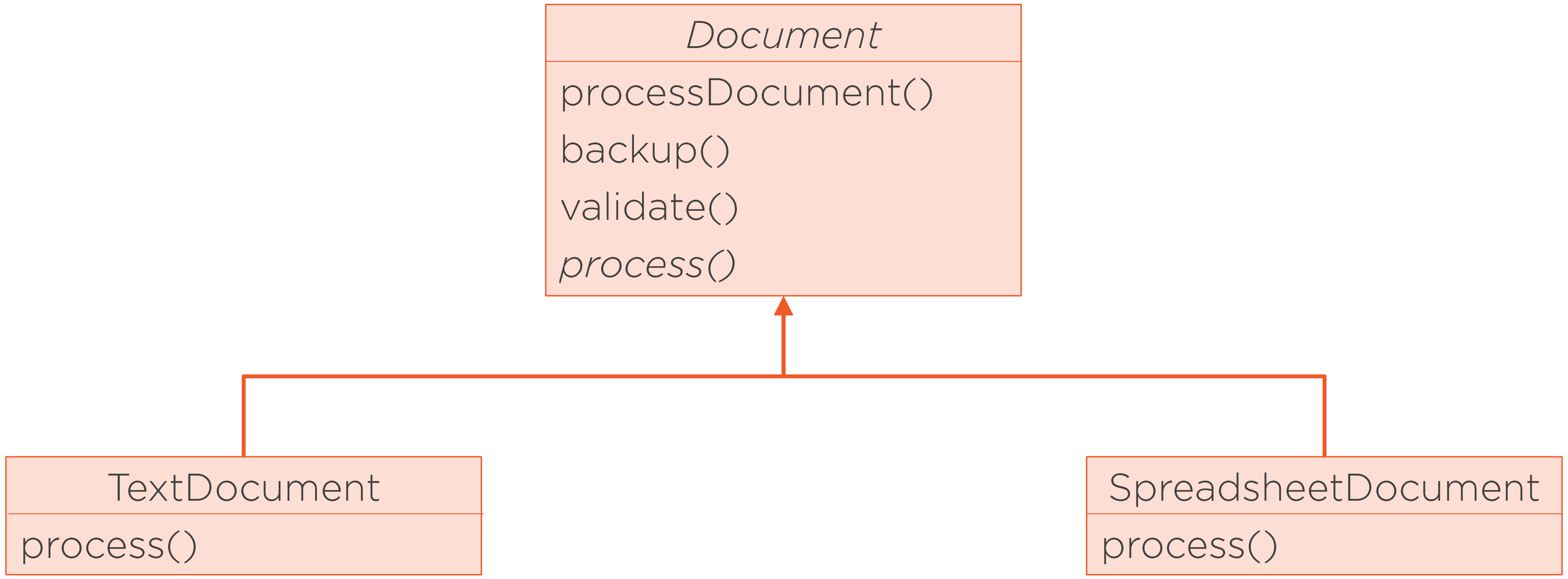
The Structure of a Method

```
myMethod() {
    // Fist statement/step
    ????
    // ...
    // Last statement
}
```

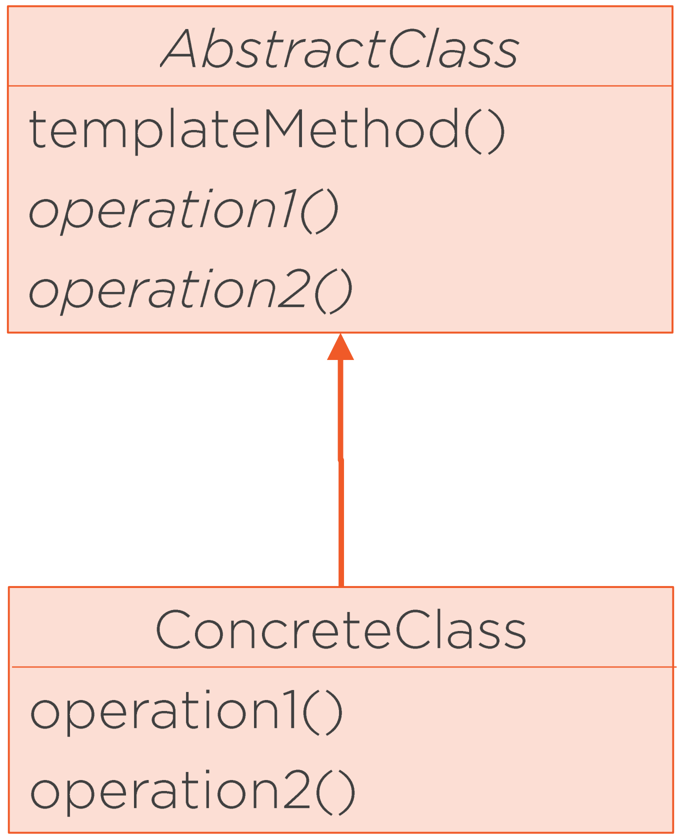
An Example

```
abstract class Document {
   void processDocument() {
      backup();
      process();
      validate();
   }
   void backup() { /* ... */ }

   void validate() { /* ... */ }

   abstract void process();
}
```

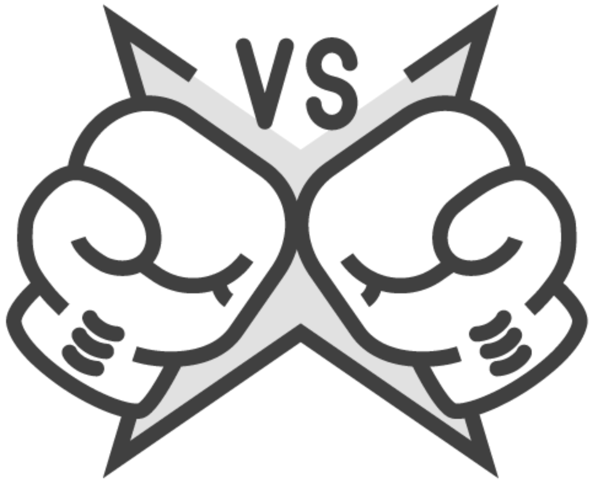
An Example


Document
processDocument()
backup()
validate()
process()
TextDocument
process()
SpreadsheetDocument
process()

Template Method Pattern



Abstract Methods and Hooks



Abstract Methods

- Required
- Steps that must be customized

Hooks

- Optional
- Abstract class may provide a default implementation

Things to Remember

Behavioral patterns

- How objects should communicate
- Their responsibilities

Things to Remember

Strategy

- Encapsulates strategies or algorithms

State

- Encapsulates states

Command

- Encapsulates requests in an object

Observer

- Notifies when the state of an object changes

Template method

- Allows to redefine steps of an algorithm by subclasses