Why Are Design Patterns Important?

Esteban Herrera
JAVA ARCHITECT

@eh3rrera www.eherrera.net

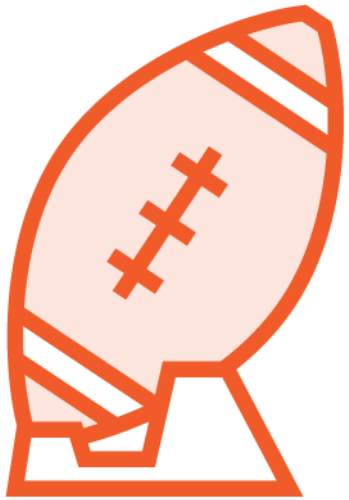
Overview

The importance of patterns

From inheritance to composition

The Importance of Patterns

Why Should You Learn Patterns?



Game rules are not enough

Knowledge

In football:

- Playbook
- Coach

Patterns capture expert knowledge

Well-designed Software Is:

- Flexible
- Easy to maintain
- Reusable

Design Pattern Benefits

Patterns are about reusability

Find the appropriate design

- Find classes and interfaces
- Determining object granularity

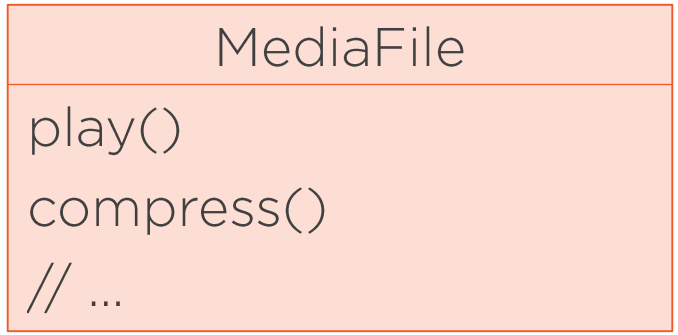
Communication and documentation

- Shared vocabulary
- Precise and complete

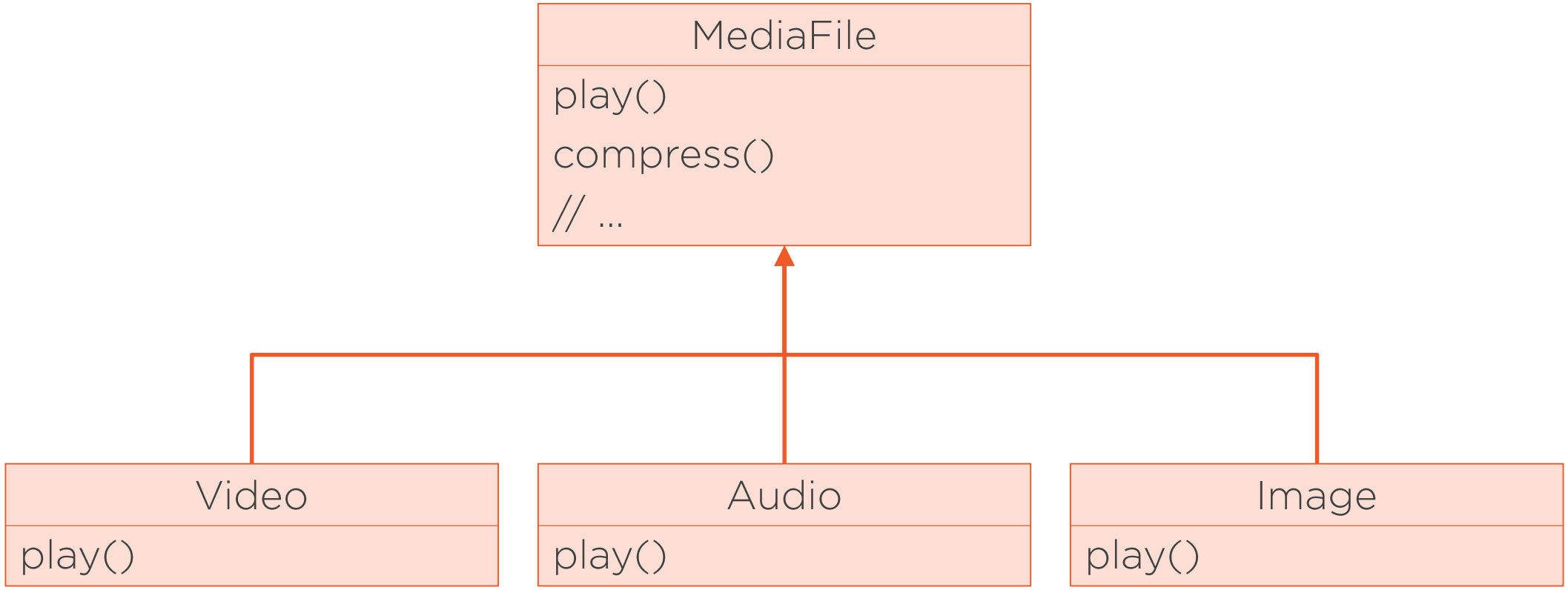
“I encapsulated this behavior in an interface, implementing each variation in a class that I will inject in this other class to delegate the behavior.”

“I used the Strategy pattern.”

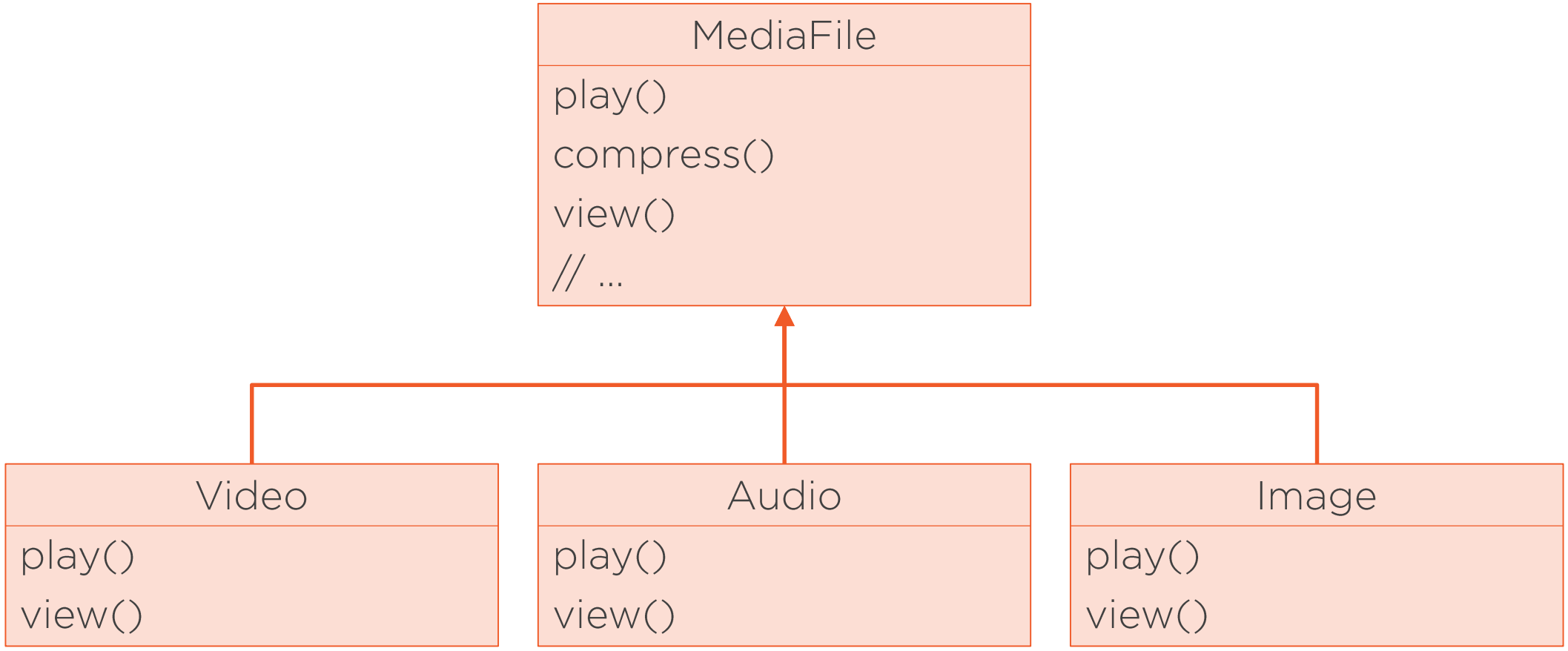
From Inheritance to Composition with the Strategy Pattern




```
if (isVideo) {
  // ...
} else if (isAudio) {
  // ...
} else {
  // ...
}
```



It's more like viewing instead of playing

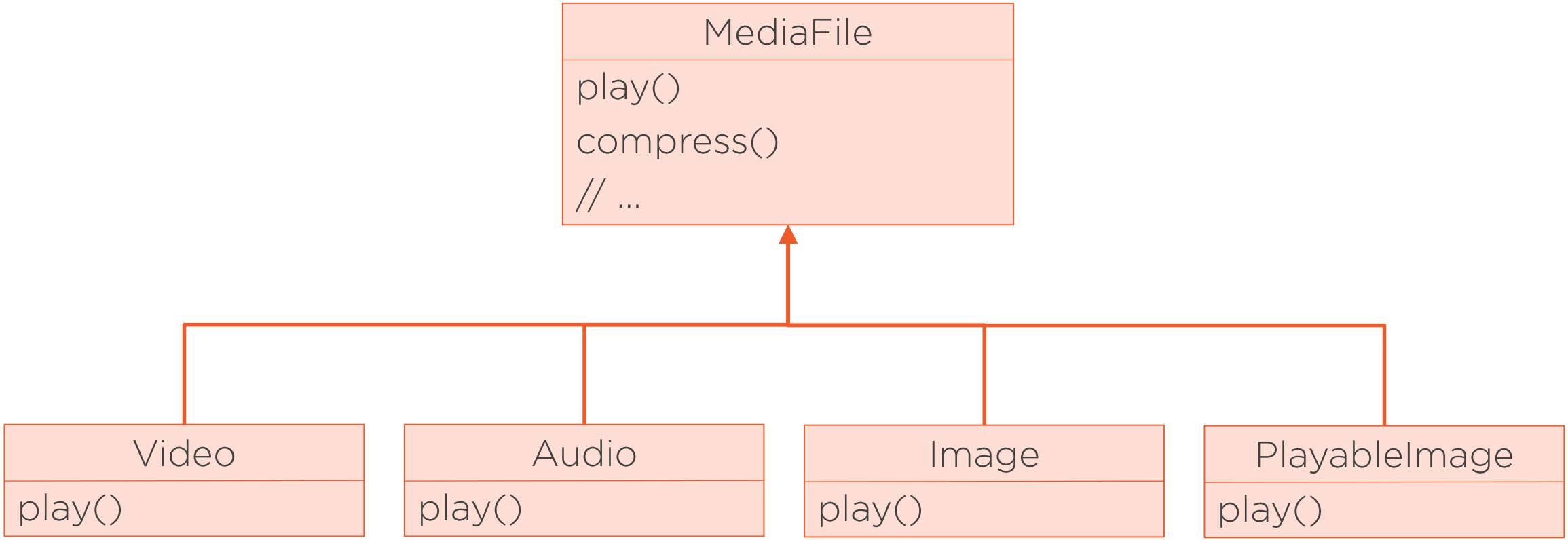

MediaFile
play()
compress()
view()
// ...
Video
play()
view()
Audio
play()
view()
Image
play()
view()

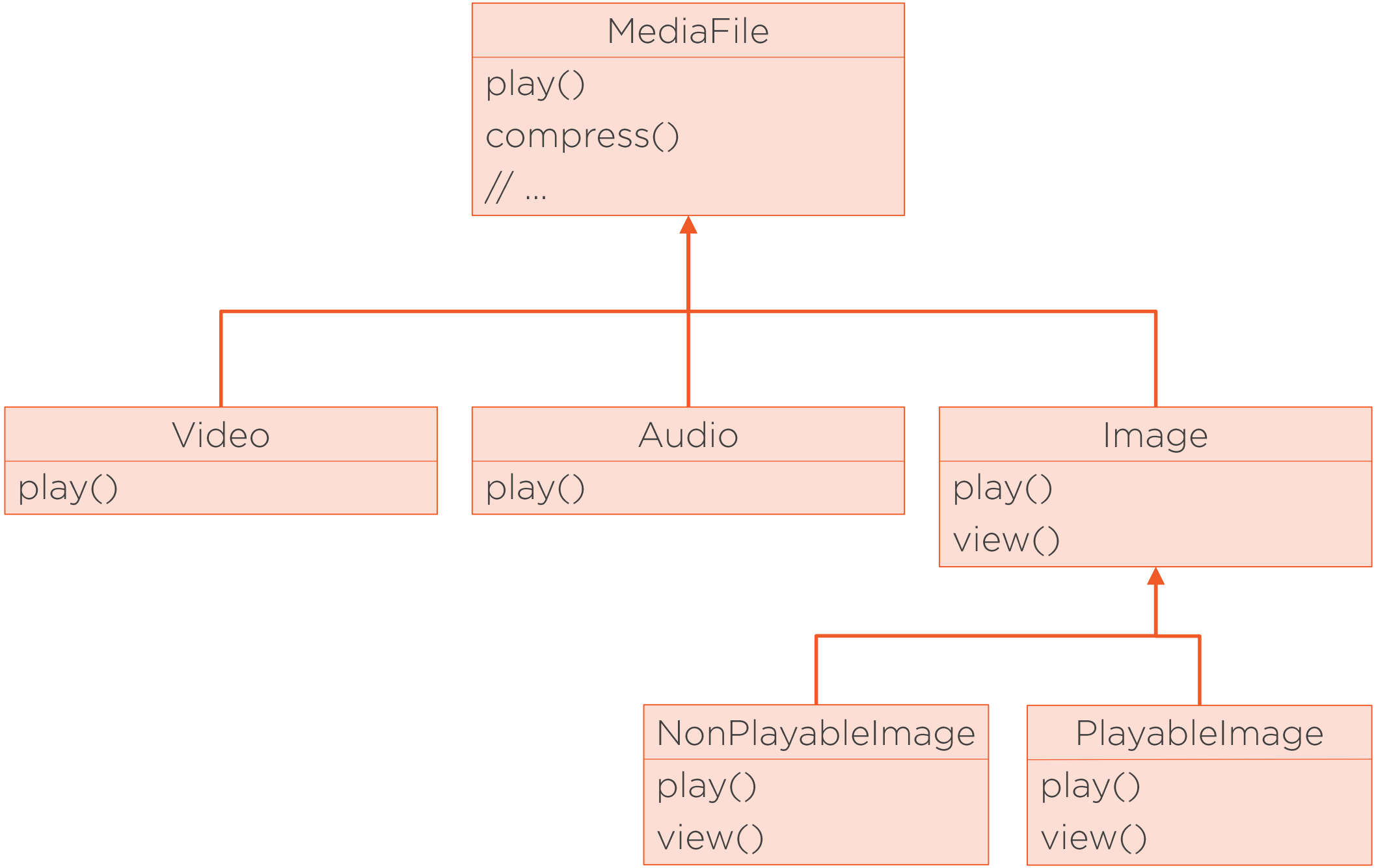
Problems with This Design

Violates the Liskov substitution principle

It's not flexible

- And if we need to support animated GIFs?


MediaFile
play()
compress()
// ...
Video
play()
Audio
play()
Image
play()
PlayableImage
play()


MediaFile
play()
compress()
// ...
Video
play()
Audio
play()
Image
play()
view()
NonPlayableImage
play()
view()
PlayableImage
play()
view()

Strategy Pattern

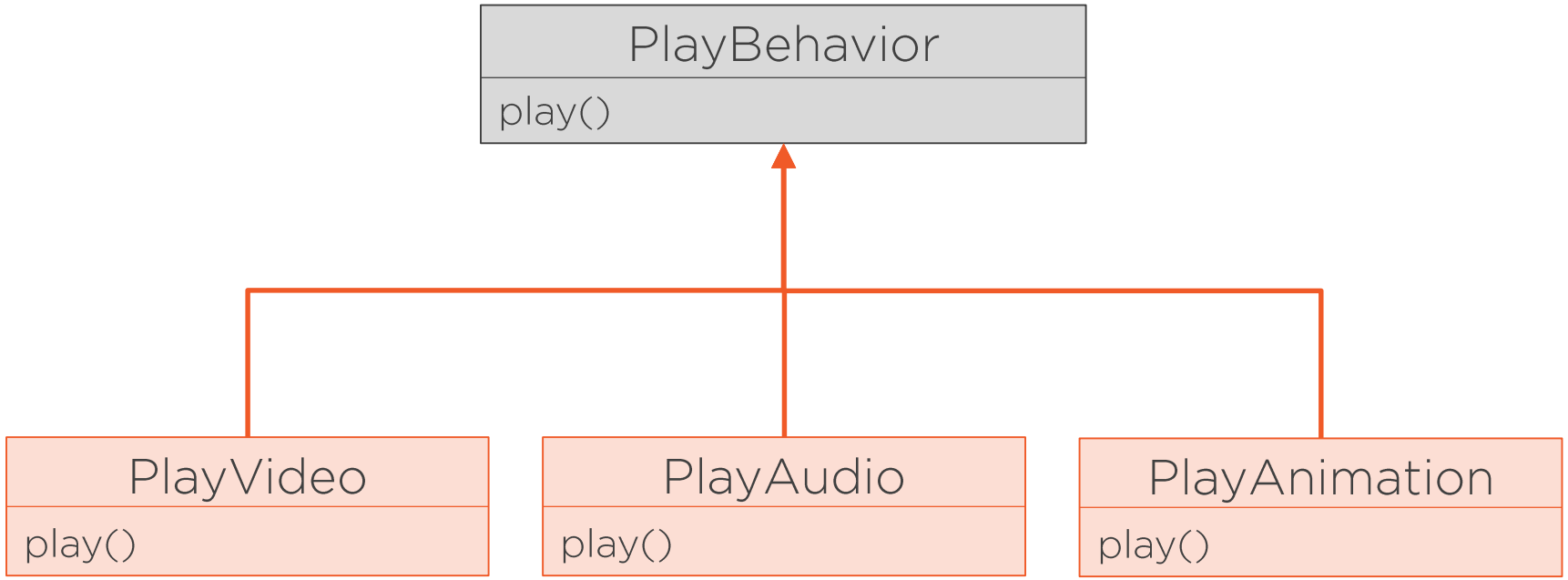
Defines a family of algorithms, encapsulates each one, and makes them interchangeable.

Strategy lets the algorithm vary independently from clients that use it.

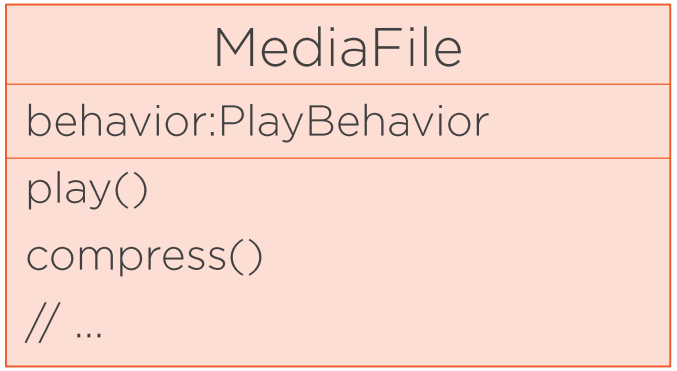
Strategy Pattern

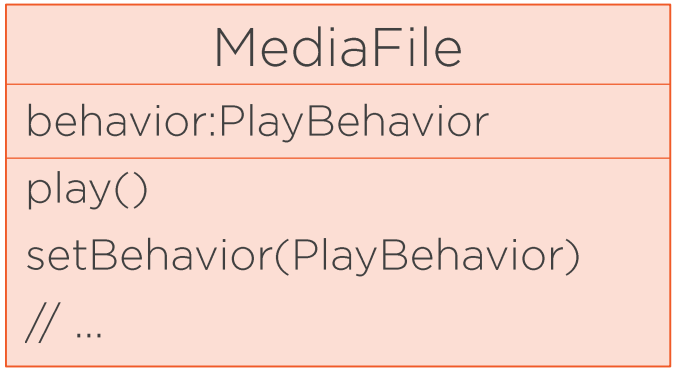
Change a part of a system independently of all other parts

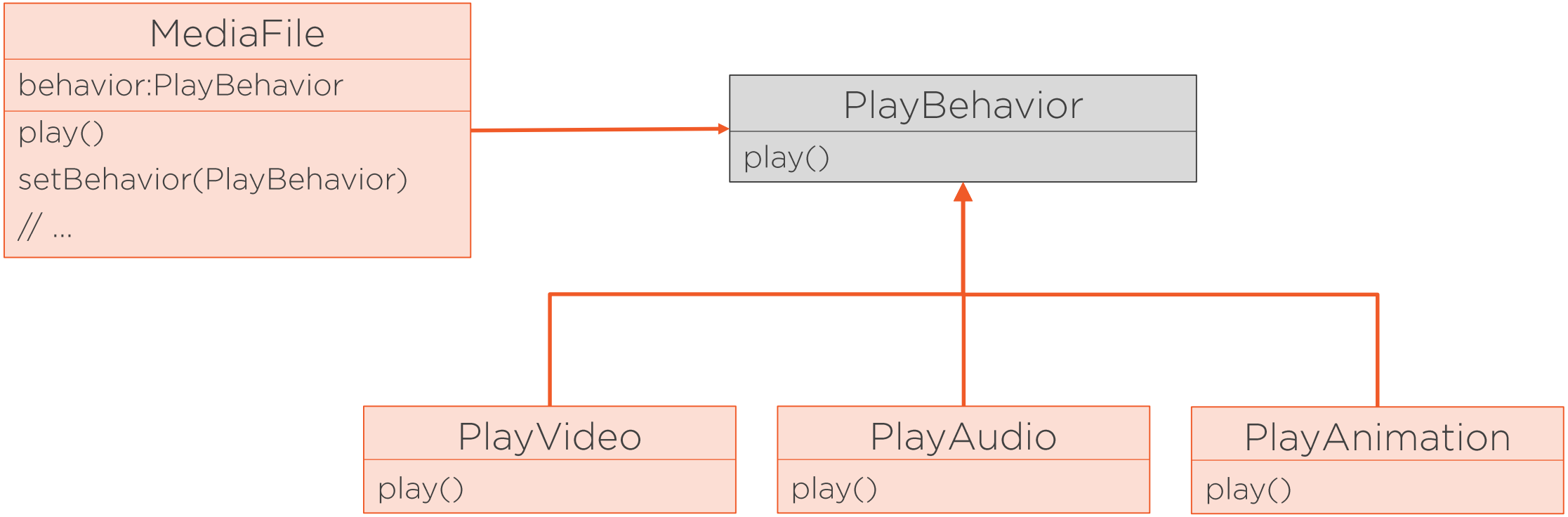
Swap out behavior at run-time


PlayBehavior
play()
PlayVideo
play()
PlayAudio
play()
PlayAnimation
play()


MediaFile
play()
compress()
// ...


MediaFile
behavior:PlayBehavior
play()
compress()
// ...


MediaFile
behavior:PlayBehavior
play()
setBehavior(PlayBehavior)
// ...


MediaFile
behavior:PlayBehavior
play()
setBehavior(PlayBehavior)
// ...
PlayBehavior
play()
PlayVideo
play()
PlayAudio
play()
PlayAnimation
play()

```
public class MediaFile {
    private PlayBehavior behavior;

    public MediaFile(PlayBehavior behavior) {
        this.behavior = behavior;
    }

    public void play() {
        (behavior != null)  ? behavior.play()
            : System.out.println("Play behavior not supported");
    }

    public void setBehavior(PlayBehavior behavior) {
        this.behavior = behavior;
    }
}
```

```
MediaFile file = new MediaFile(new PlayVideo());

file.play() // Play as video

file.setBehavior(new PlayAudio());

file.play() // Play as audio

file.setBehavior(null);

file.play() // No play behavior (for images)
```

Isn’t this just polymorphism?

Isn’t this just polymorphism?

No

The Strategy Pattern Is a Mix Of:

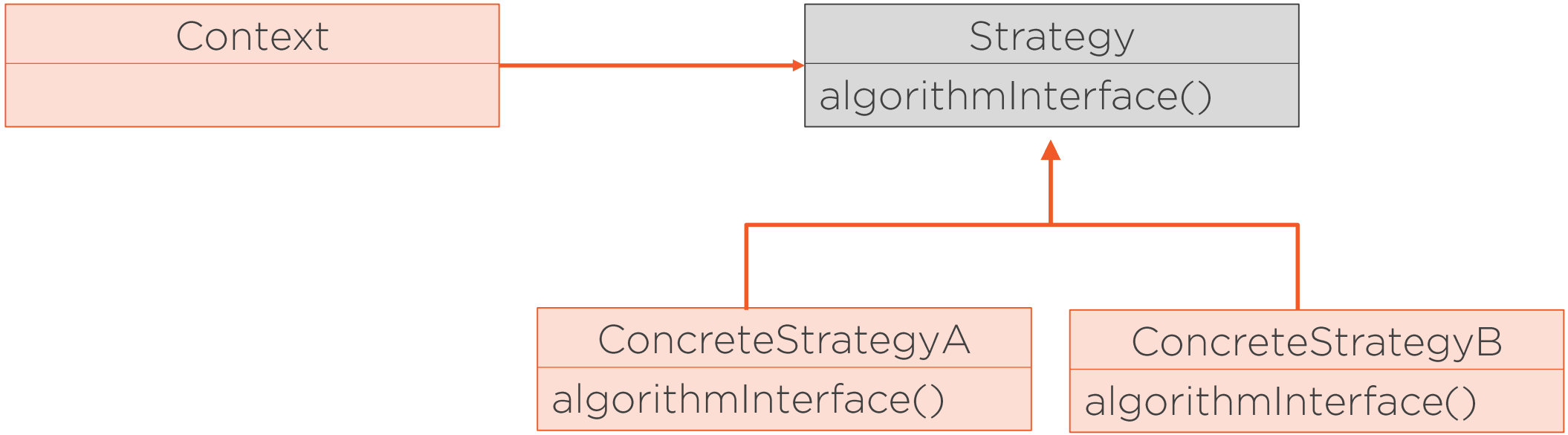
Encapsulating what changes

Favoring composition over inheritance

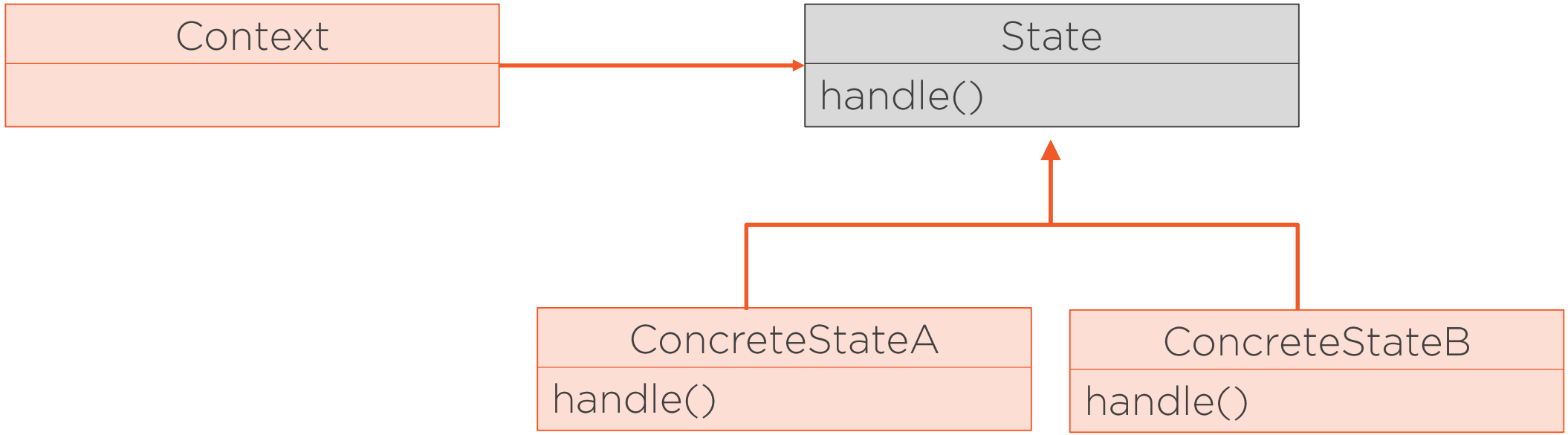
Open-close principle

Programming to interfaces

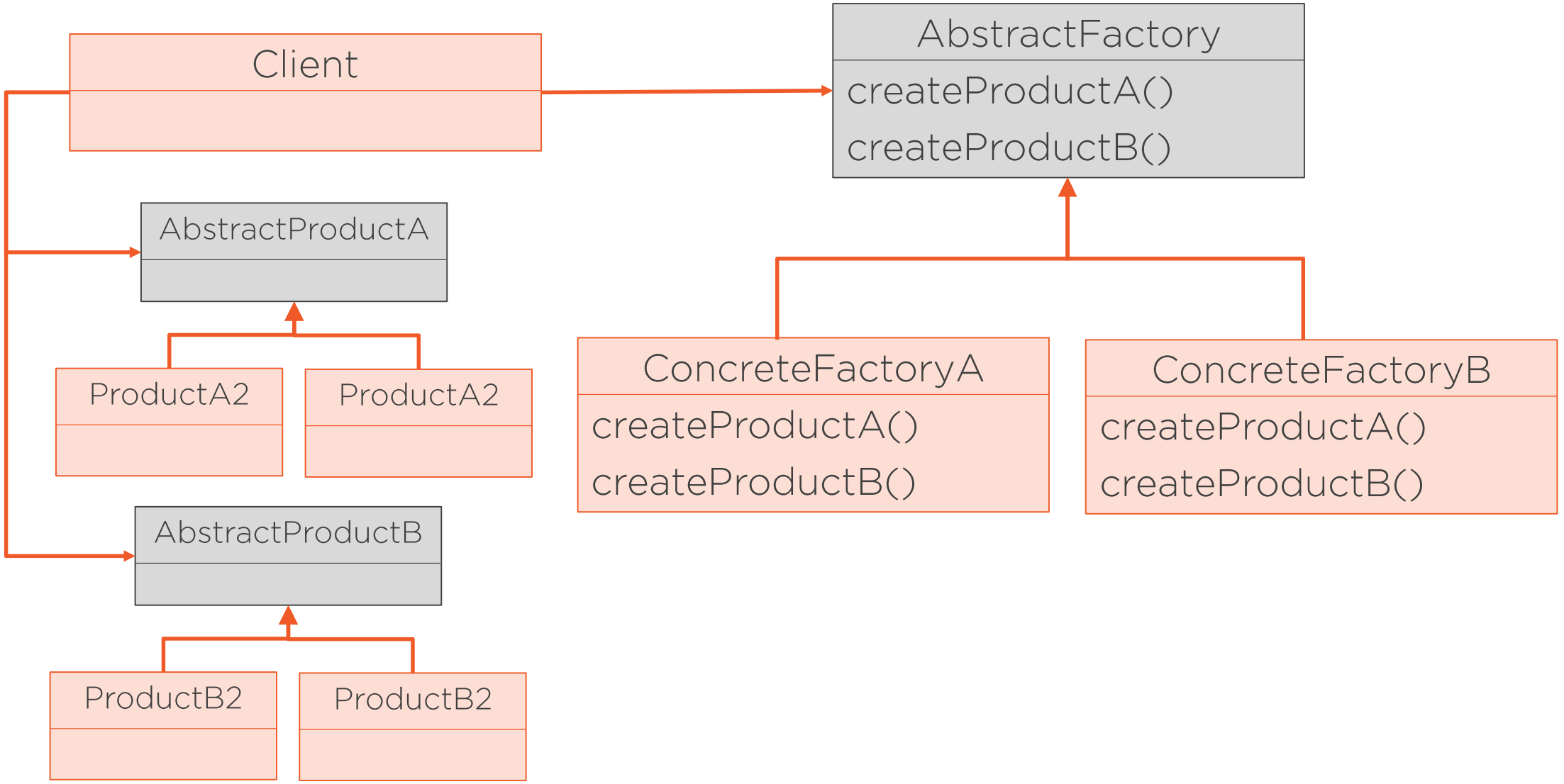
Canonical Diagram of the Strategy Pattern



Canonical Diagram of the State Pattern



Canonical Diagram of the Abstract Factory Pattern



Things to Remember

Patterns capture expert knowledge

- Reuse that knowledge
- Find the appropriate design
- Shared and precise vocabulary

Inheritance is not always the best solution

- Composition can offer a more flexible alternative
- Patterns can guide you

Study and know principles and patterns well