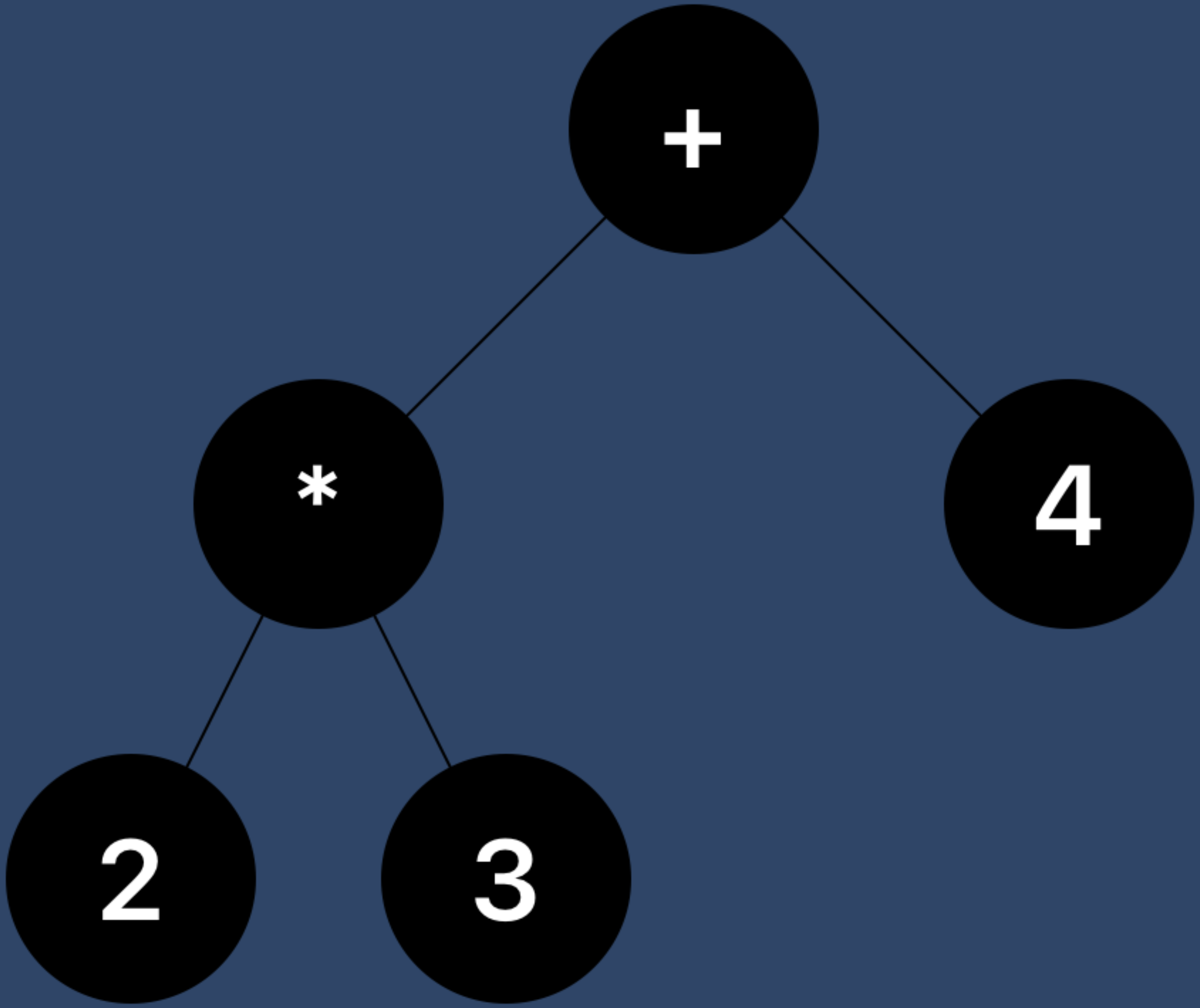
Polymorphism with Protocols

Good API Design is…

👀 🔨 🤷‍♀️


+
*
4
2
3

Naive Approach

```
class Node {
  var value: Double?
  var operation: String?
  var leftChild: Node?
  var rightChild: Node?

  func evaluate() -> Double
}
```

Values

```
class Node {
  var value: Double?
  var operation: String?
  var leftChild: Node?
  var rightChild: Node?

  func evaluate() -> Double
}
```

Operations

```
class Node {
  var value: Double?
  var operation: String?
  var leftChild: Node?
  var rightChild: Node?

  func evaluate() -> Double
}
```

Is this reasonable? 🤔

Optionals? Everywhere

```
class Node {
  var value: Double?
  var operation: String?
  var leftChild: Node?
  var rightChild: Node?

  func evaluate() -> Double
}
```

Which can be `nil`?

```swift
class Node {
  var value: Double?
  var operation: String?
  var leftChild: Node?
  var rightChild: Node?

  func evaluate() -> Double
}
```

Structs or Classes?

```
class Node {
  var value: Double?
  ...
}

struct Node {
  var value: Double?
  ...
}
```

What happens if we do this?

```
node.leftChild = nil
```

What we did

- Naive class implementation
- Introducing abstract classes
- Moving into protocols (value types)
- Implementing a third-party protocol (rendering) (protocol extensions)

Additional Readings

Inheritance, Polymorphism, & Testing
Misko Hevery

Everyone is an API designer
John Sundell

Playground Quicklook for Binary Trees
Swift Talk

Thank You

@matthewcheok