The Swift Programming Language

Part 7 – Another Type Casting

iCodeWave Community

What We’re Going to Learn?

Type Checking

Optional Downcasting

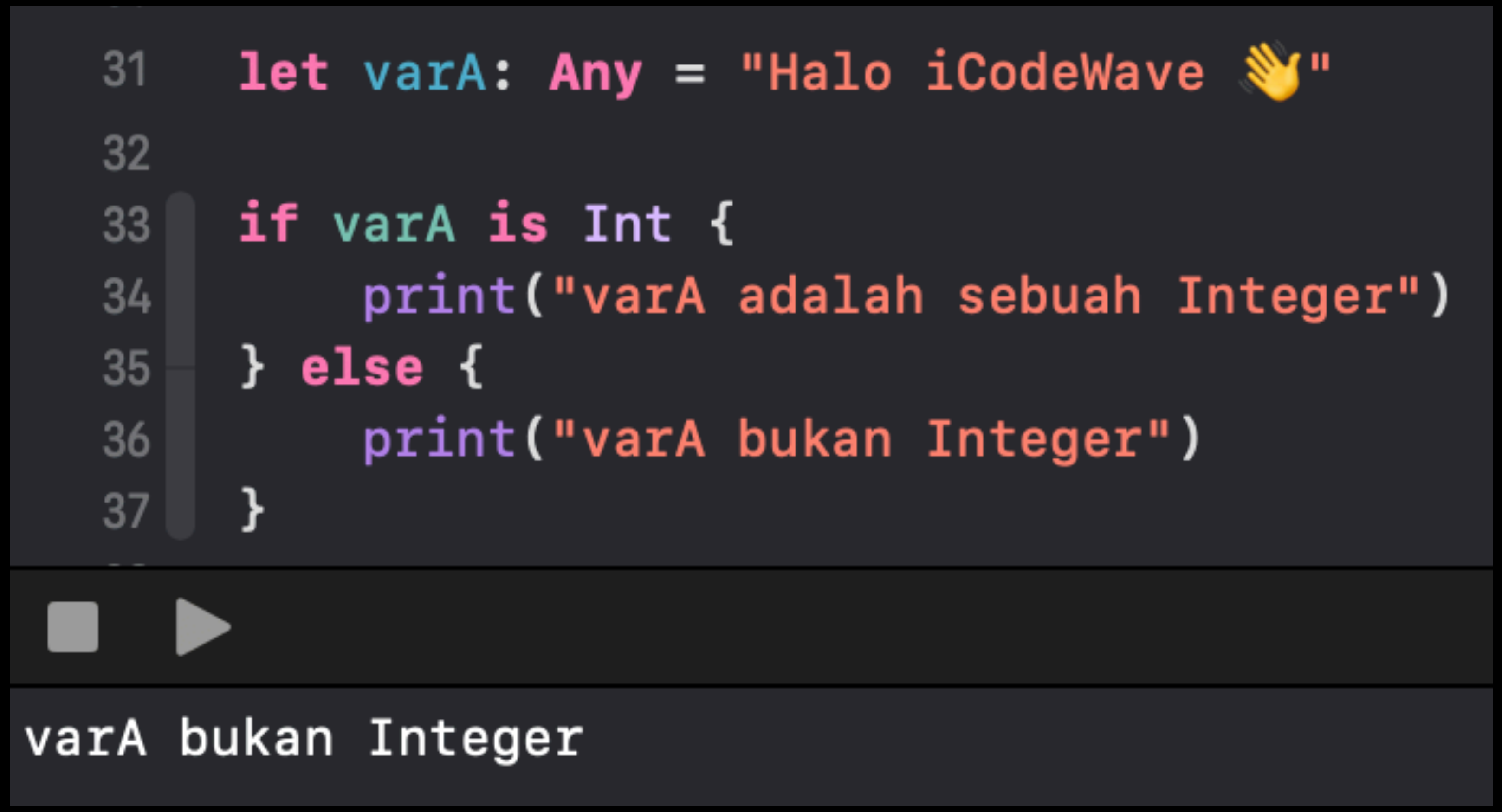
Force Downcasting

Upcasting

Type Checking

Points to pay attention & Example

- Type checking is the process of checking the data type of an instance
- In Swift, you can use the is operator to do type checking



```swift
let varA: Any = "Halo iCodeWave 👋"

if varA is Int {
    print("varA adalah sebuah Integer")
} else {
    print("varA bukan Integer")
}
```

```
varA bukan Integer
```

iCodeWave Community

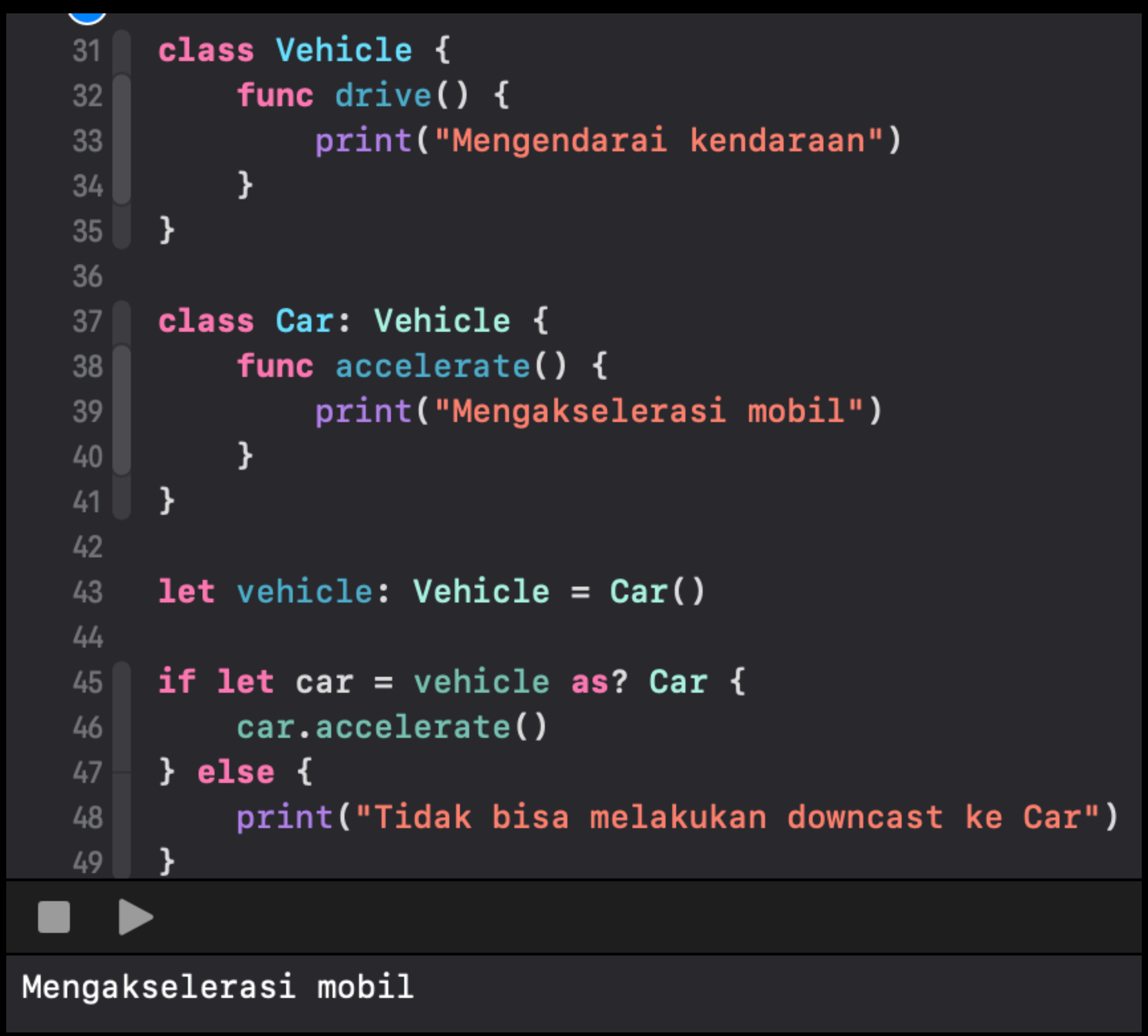
Optional Downcasting

Points to pay attention

- Optional downcasting is used when you want to change the type of an object to a more specific type, and return an optional value if the downcasting fails
- Converts an instance of a superclass to a subclass
- Use as? as a syntax for optional downcasting

Optional Downcasting

Example code #1 – Downcasting Superclass Vehicle() to Subclass Car()



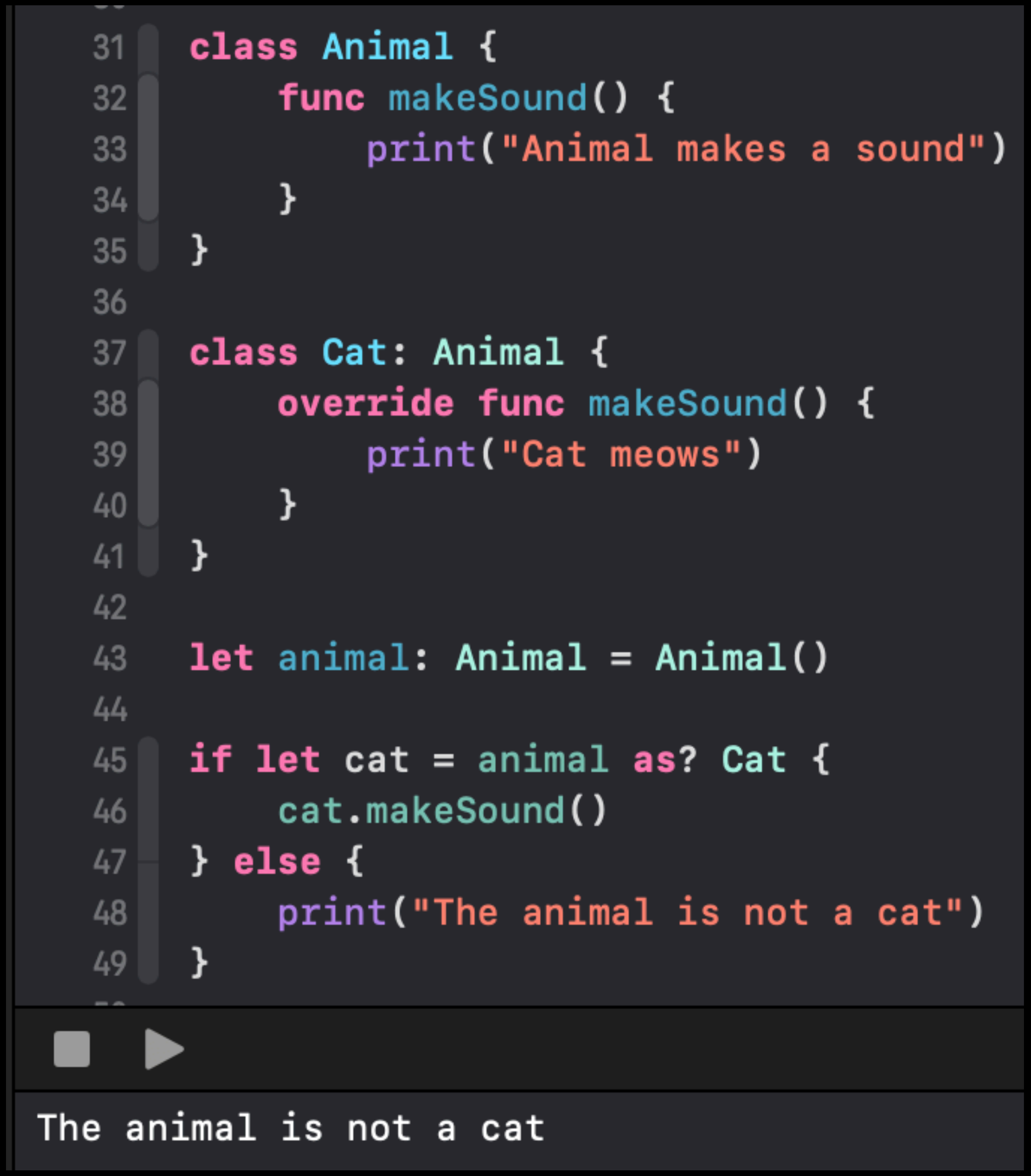
```swift
class Vehicle {
    func drive() {
        print("Mengendarai kendaraan")
    }
}

class Car: Vehicle {
    func accelerate() {
        print("Mengakselerasi mobil")
    }
}

let vehicle: Vehicle = Car()

if let car = vehicle as? Car {
    car.accelerate()
} else {
    print("Tidak bisa melakukan downcast ke Car")
}
```

```
Mengakselerasi mobil
```

iCodeWave Community

Optional Downcasting

Example code #2 – Else statement executed if downcasting failed



```swift
class Animal {
    func makeSound() {
        print("Animal makes a sound")
    }
}

class Cat: Animal {
    override func makeSound() {
        print("Cat meows")
    }
}

let animal: Animal = Animal()

if let cat = animal as? Cat {
    cat.makeSound()
} else {
    print("The animal is not a cat")
}
```

```
The animal is not a cat
```

iCodeWave Community

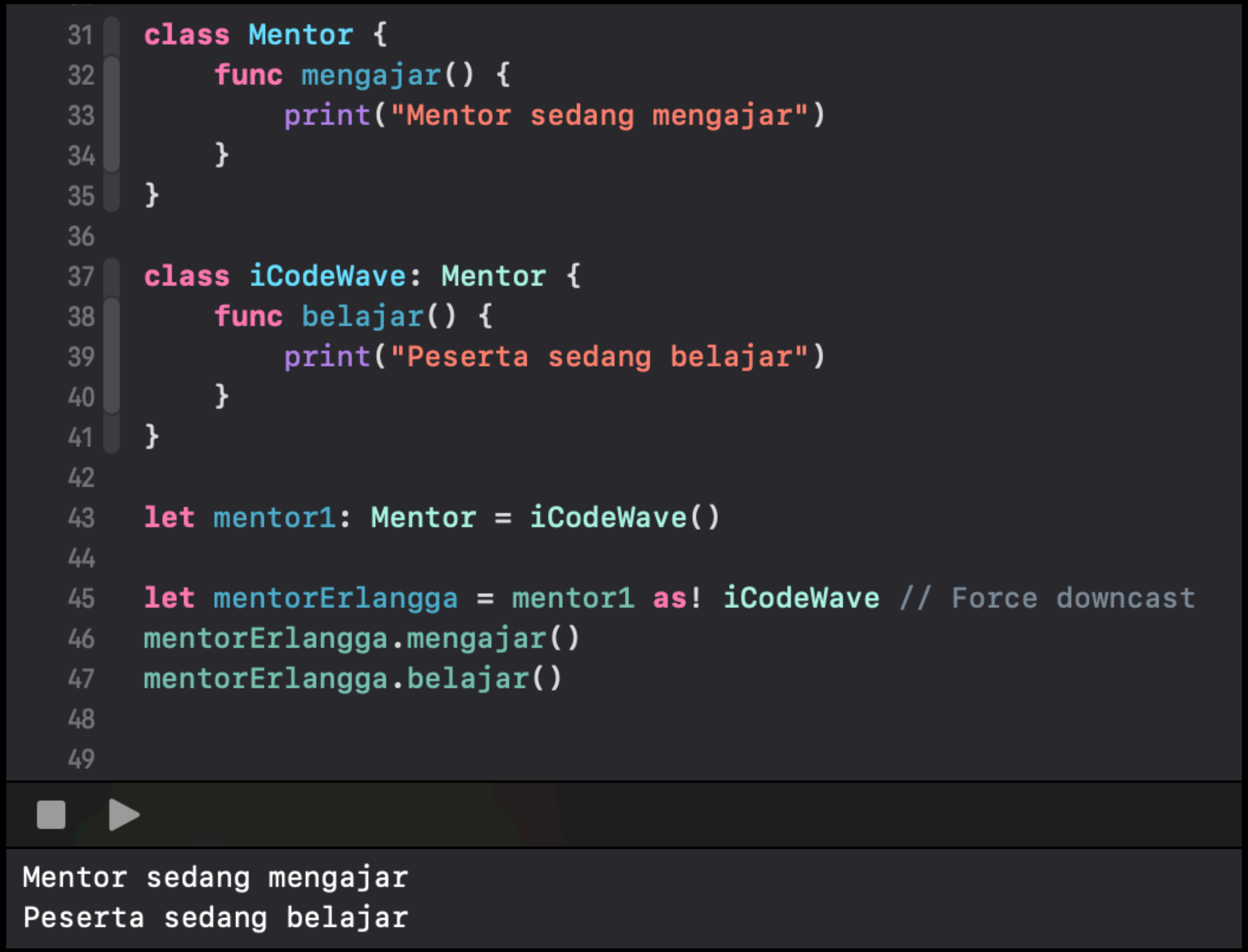
Force Downcasting

Points to pay attention

- Force downcasting is done when you believe that downcasting will be successful, and you want to take a specific type value without using options.
- Use the operator as! for force downcasting
- Keep in mind that the use of force downcasting can cause runtime errors if downcasting fails.

Force Downcasting

Example code #1 – Successful force downcasting



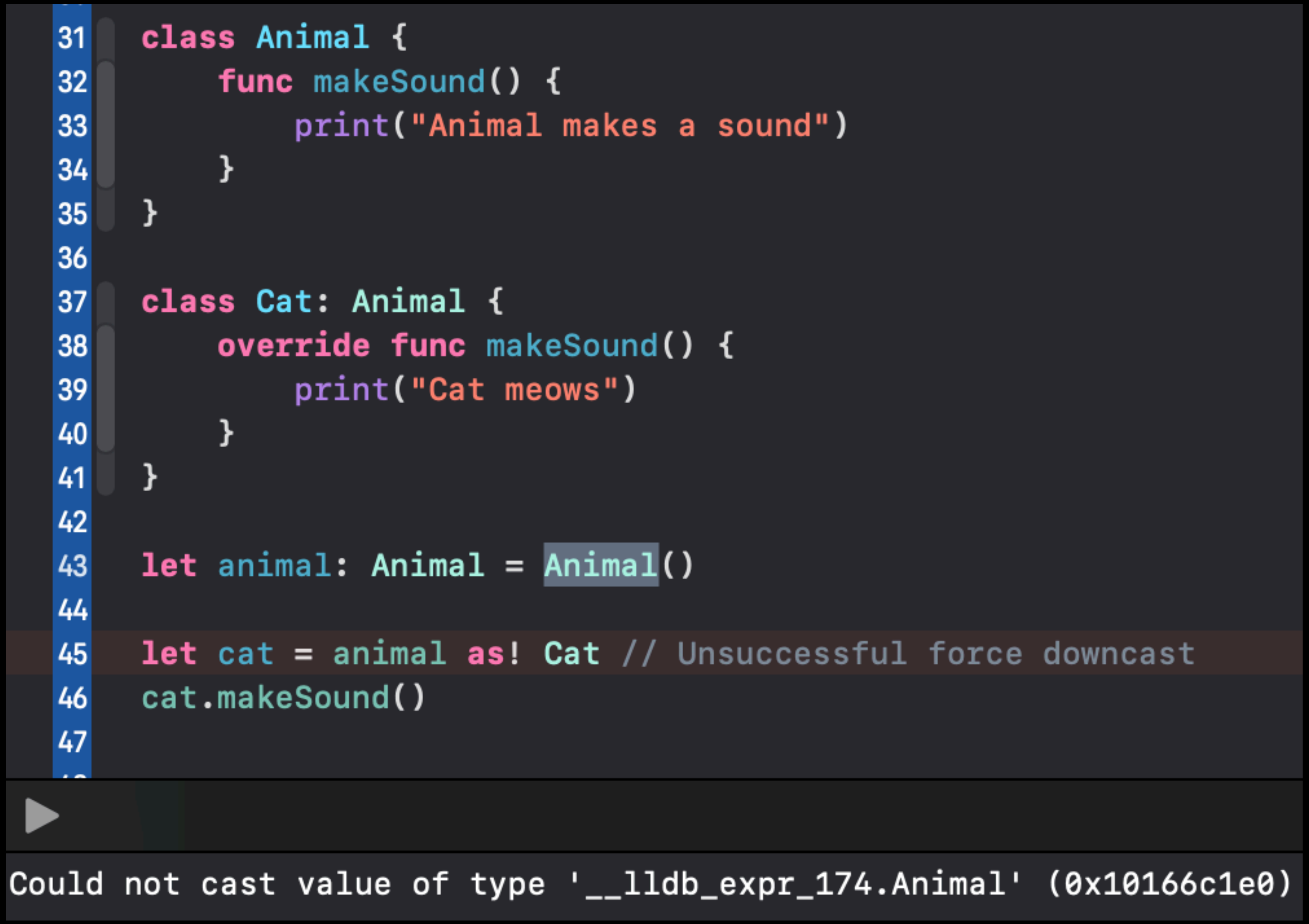
```swift
class Mentor {
    func mengajar() {
        print("Mentor sedang mengajar")
    }
}

class iCodeWave: Mentor {
    func belajar() {
        print("Peserta sedang belajar")
    }
}

let mentor1: Mentor = iCodeWave()

let mentorErlangga = mentor1 as! iCodeWave // Force downcast
mentorErlangga.mengajar()
mentorErlangga.belajar()
```

```
Mentor sedang mengajar
Peserta sedang belajar
```

iCodeWave Community

Force Downcasting

Example code #2 – What if the case is failed?



```swift
class Animal {
    func makeSound() {
        print("Animal makes a sound")
    }
}

class Cat: Animal {
    override func makeSound() {
        print("Cat meows")
    }
}

let animal: Animal = Animal()

let cat = animal as! Cat // Unsuccessful force downcast
cat.makeSound()
```

```
Could not cast value of type '__lldb_expr_174.Animal' (0x10166c1e0)
```

iCodeWave Community

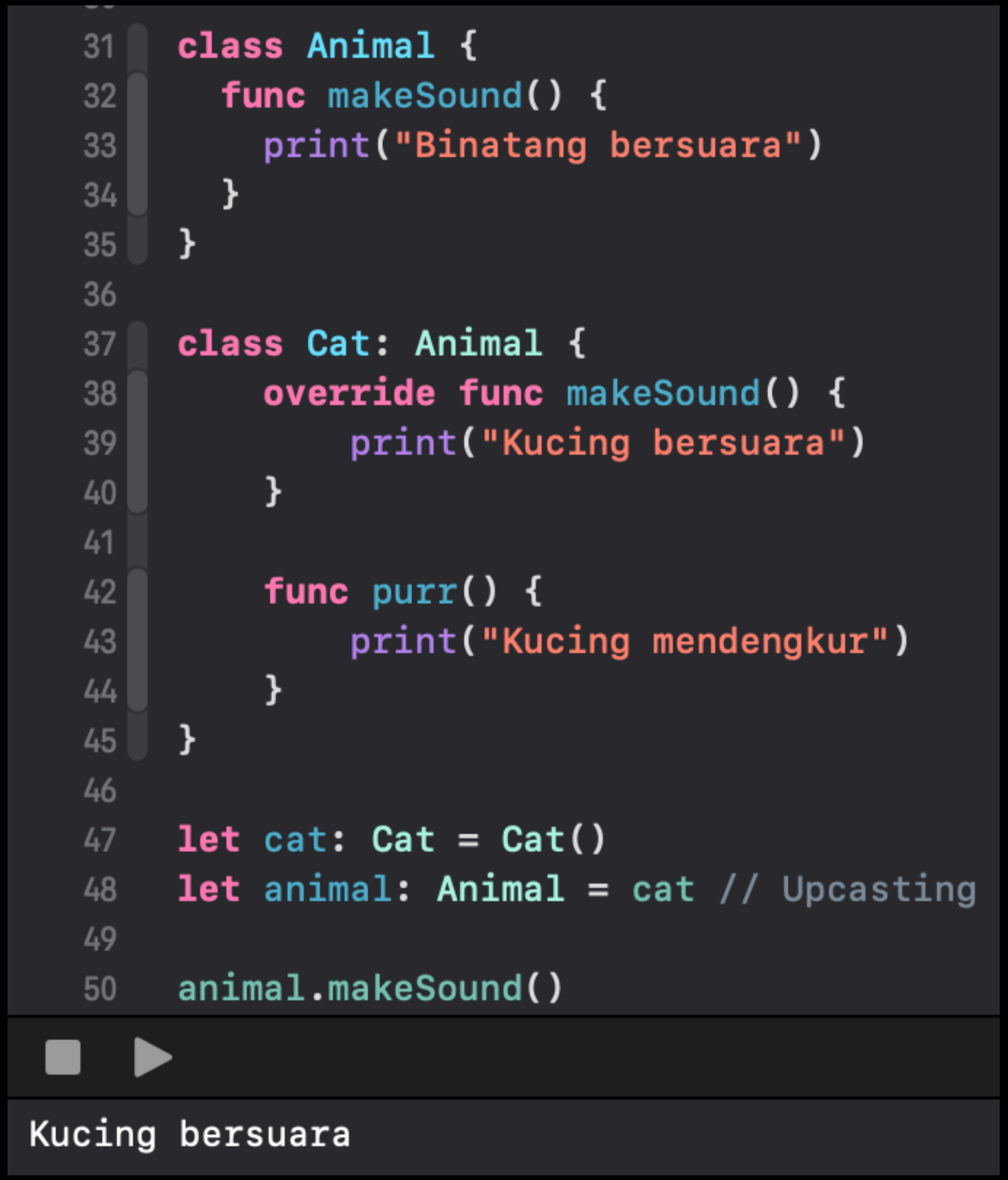
Upcasting

Points to pay attention

- Upcasting is the opposite of downcasting. This happens when you change the type of an object to its parent type
- Casts an instance of a subclass to its superclass
- Automatically, upcasting is safe and **does not** need to use a special operator
- There are **NO FAIL** in upcasting

Upcasting

Example code #1



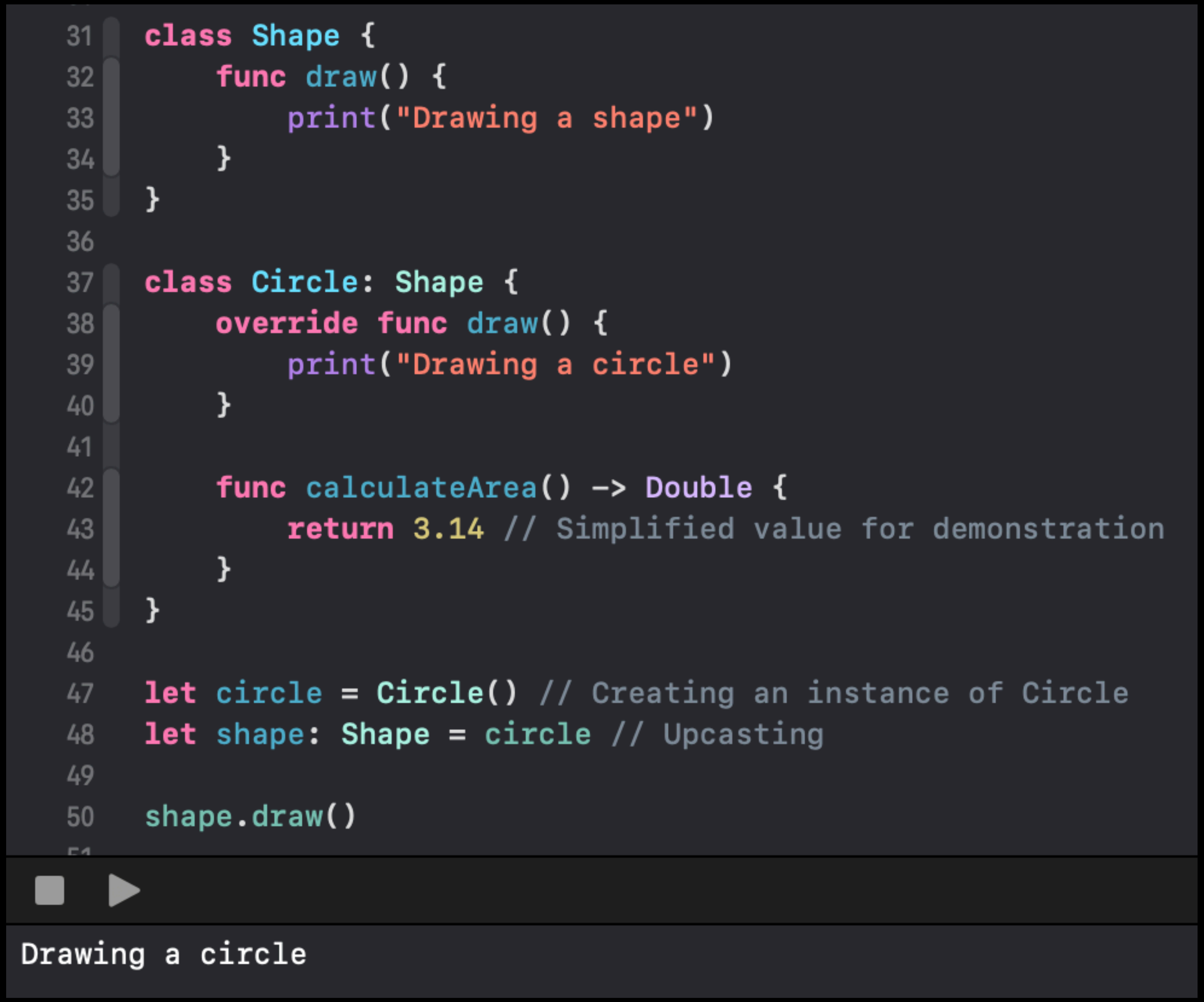
```swift
class Animal {
  func makeSound() {
    print("Binatang bersuara")
  }
}

class Cat: Animal {
    override func makeSound() {
        print("Kucing bersuara")
    }

    func purr() {
        print("Kucing mendengkur")
    }
}

let cat: Cat = Cat()
let animal: Animal = cat // Upcasting

animal.makeSound()
```

```
Kucing bersuara
```

iCodeWave Community

Upcasting

Example code #2



```swift
class Shape {
    func draw() {
        print("Drawing a shape")
    }
}

class Circle: Shape {
    override func draw() {
        print("Drawing a circle")
    }

    func calculateArea() -> Double {
        return 3.14 // Simplified value for demonstration
    }
}

let circle = Circle() // Creating an instance of Circle
let shape: Shape = circle // Upcasting

shape.draw()
```

```
Drawing a circle
```

iCodeWave Community

Thanks For Your Attendance Today!

iCodeWave Community