class User {

var name: String

init(name: String) {

self.name = name

}

}

var user1 = User(name: "Alice")

var user2 = user1 // user2 is now a reference to the same User instance as user1

user2.name = "Bob"

print(user1.name) // This will print "Bob"

struct Point {

var x: Int

var y: Int

}

var point1 = Point(x: 1, y: 2)

var point2 = point1 // point2 is now a copy of point1

point2.x = 3

print(point1.x) // This will print 1, not 3

class MyClass { }

var reference1: MyClass? = MyClass() // ARC sets reference count to 1

var reference2: reference1 // ARC increases count to 2

reference1 = nil // ARC decreases count to 1

reference2 = nil // ARC decreases count to 0,

// and deallocates the MyClass instance

class MyClass {

var property: () -> () = {}

func configureClosure() {

property = { [weak self] in

print(self)

}

}

deinit {

print("MyClass is being deinitialized")

}

}

class Parent {

var child: Child?

}

class Child {

var parent: Parent?

}

var parent = Parent()

var child = Child()

parent.child = child

child.parent = parent

class Child {

weak var parent: Parent?

}

class MyClass {

deinit {

// Perform clean-up

}

}

{ [unowned self, weak delegate = self.delegate] (parameter) -> ReturnType in

// Closure body

}

networkRequest.fetchData { [weak self] result in

switch result {

case .success(let data):

self?.updateUI(with: data)

case .failure(let error):

self?.showError(error)

}

}