(e)

```
class Fruits {
    var fruitName: String
    init(fruitName: String) {
        self.fruitName = fruitName
    }
    func display() {
        print("I buy some \(fruitName)s")
    }
    deinit {
        print("Executing deinitializer")
    }
}

var oneObject2: Fruits = Fruits(fruitName: "Kiwi")
oneObject2.display()
oneObject2 = nil
```


13 CHAPTER

自動參考計數

自動參考計數 (Automatic Reference Counting, ARC) 我實在太愛妳了。因為有撰寫過 iPhone App 的人都有受過記憶體不足而當機的經驗。在未有 ARC 時，程式設計師主要的任務之一，就是要負責所有物件參考計數的問題，如今這個惡夢已結束，取而代之的是系統自動化的來幫你處理參考計數的問題，所以現在撰寫 iPhone 的 App 真是幸福多了，所以您還在猶豫什麼，動手吧！

當類別出現參考循環時，會出現無法釋還的結果，因此必須採取一些機制來解決。這些機制是本章將探討的主題。我們將配合示意圖來加以解說。

13.1 自動參考計數如何運作

為了能清楚知道自動參考計數是如何運作，我們以一範例來說明，程式如下所示：

範例程式

```
01 // Automatic Reference Count(ARC)
02 class Book {
03     let author: String
04     let bookName: String
05     init(author: String, bookName: String) {
06         self.author = author
07         self.bookName = bookName
```

(e)

```
class Fruits {
    var fruitName: String
    init(fruitName: String) {
        self.fruitName = fruitName
    }
    func display() {
        print("I buy some \(fruitName)s")
    }
    deinit {
        print("Executing deinitializer")
    }
}

var oneObject2: Fruits = Fruits(fruitName: "Kiwi")
oneObject2.display()
oneObject2 = nil
```


13 CHAPTER

自動參考計數

自動參考計數 (Automatic Reference Counting, ARC) 我實在太愛妳了。因為有撰寫過 iPhone App 的人都有受過記憶體不足而當機的經驗。在未有 ARC 時，程式設計師主要的任務之一，就是要負責所有物件參考計數的問題，如今這個惡夢已結束，取而代之的是系統自動化的來幫你處理參考計數的問題，所以現在撰寫 iPhone 的 App 真是幸福多了，所以您還在猶豫什麼，動手吧！

當類別出現參考循環時，會出現無法釋還的結果，因此必須採取一些機制來解決。這些機制是本章將探討的主題。我們將配合示意圖來加以解說。

13.1 自動參考計數如何運作

為了能清楚知道自動參考計數是如何運作，我們以一範例來說明，程式如下所示：

範例程式

```
01 // Automatic Reference Count(ARC)
02 class Book {
03     let author: String
04     let bookName: String
05     init(author: String, bookName: String) {
06         self.author = author
07         self.bookName = bookName
```