Warunki i opcje

Poprzednio przyjrzałeś się typom danych, stałym, zmiennym i operacjom. W tym momencie jesteś w stanie pisać proste programy przetwarzające litery i cyfry. Jednak programy nie zawsze działają po kolei. Często będziesz musiał wykonać różne instrukcje w zależności od warunku. Swift pozwala to zrobić za pomocą instrukcji warunkowych, a dowiesz się, jak z nich korzystać w tym rozdziale. Inną rzeczą, którą mogłeś zauważyć, jest to, że w ostatnim rozdziale każdej zmiennej lub stałej od razu przypisywana była wartość. Co zrobić, jeśli potrzebujesz zmiennej, której wartość może początkowo nie być obecna? Będziesz potrzebować sposobu na utworzenie zmiennej, która może mieć wartość lub nie. Swift pozwala to zrobić za pomocą opcji, o których również dowiesz się w tym rozdziale. Pod koniec powinieneś być w stanie pisać programy, które robią różne rzeczy w oparciu o różne warunki i obsługiwać zmienne, które mogą lub nie mają wartości. Omówione zostaną następujące tematy:

• Wprowadzenie warunków

• Przedstawiamy opcje i opcjonalne wiązanie

Przedstawiamy warunki

Czasami będziesz chciał wykonać różne bloki kodu na podstawie określonego warunku, na przykład w następujących scenariuszach:

• Wybór pomiędzy różnymi rodzajami pokoi w hotelu. Cena za większe pokoje byłaby wyższa.

• Przełączanie między różnymi metodami płatności w sklepie internetowym. Różne metody płatności miałyby różne procedury.

• Decydowanie o tym, co zamówić w restauracji typu fast-food. Procedury przygotowania dla każdego artykułu spożywczego byłyby inne.

Aby to zrobić, użyjesz warunków warunkowych. W Swift jest to zaimplementowane za pomocą instrukcji if (dla jednego warunku) i instrukcji switch (dla wielu warunków). Zobaczmy, jak instrukcje są używane do wykonywania różnych zadań w zależności od wartości warunku w następnej sekcji.

Korzystanie z instrukcji if

Instrukcja if wykonuje blok kodu, jeśli warunek jest prawdziwy, i opcjonalnie kolejny blok kodu, jeśli warunek jest fałszywy. Instrukcja if wygląda tak:

```
if condition {

code1

} else {

code2

}
```

Zaimplementujmy teraz instrukcję if, aby zobaczyć to w akcji. Wyobraź sobie, że programujesz aplikację dla restauracji. Aplikacja umożliwiłaby sprawdzenie, czy restauracja jest otwarta, wyszukanie restauracji i sprawdzenie, czy klient przekroczył limit wieku, w którym można pić. Wykonaj następujące kroki:

1. Aby sprawdzić, czy restauracja jest otwarta, dodaj następujący kod do swojego placu zabaw, aby utworzyć stałą i wykonać instrukcję, jeśli wartość stałej jest prawdziwa. Kliknij przycisk Odtwórz/Zatrzymaj, aby go uruchomić:

```
let isRestaurantOpen = true

if isRestaurantOpen {

print("Restaurant is open.")

}
```

Najpierw utworzyłeś stałą isRestaurantOpen i przypisałeś jej wartość true. Następnie masz instrukcję if, która sprawdza wartość przechowywaną w isRestaurantOpen. Ponieważ wartość jest true, instrukcja print() jest wykonywana, a Restauracja jest otwarta jest drukowana w obszarze Debug.

2. Spróbuj zmienić wartość isRestaurantOpen na false i ponownie uruchom kod. Ponieważ warunek jest teraz fałszywy, nic nie zostanie wydrukowane w obszarze debugowania.

3. Możesz również wykonać instrukcje, jeśli wartość jest fałszywa. Załóżmy, że klient wyszukał konkretną restaurację, której nie ma w bazie danych aplikacji, więc aplikacja powinna wyświetlić komunikat z informacją, że restauracja nie została znaleziona. Wpisz następujący kod, aby utworzyć stałą i wykonać instrukcję, jeśli wartość stałej jest fałszywa:

```
let isRestaurantFound = false

if isRestaurantFound == false {

print("Restaurant was not found")

}
```

Stała isRestaurantFound jest ustawiona na false. Następnie sprawdzana jest instrukcja if. Warunek isRestaurantFound == false zwraca prawdę, a restauracja nie została znaleziona jest drukowana w obszarze Debug.

4. Spróbuj zmienić wartość isRestaurantFound na true. Ponieważ warunek jest teraz fałszywy, nic nie zostanie wydrukowane w obszarze debugowania.

5. Aby wykonać jeden zestaw instrukcji, jeśli warunek jest prawdziwy, a drugi zestaw instrukcji, jeśli warunek jest fałszywy, użyj słowa kluczowego else. Wpisz poniższy kod, który sprawdzi, czy klient w barze przekroczył granicę wieku spożywania alkoholu:

```
let drinkingAgeLimit = 21

let customerAge = 23

if customerAge < drinkingAgeLimit {

print("Under age limit")

} else {

print("Over age limit")

}
```

W tym przypadku drinkingAgeLimit przypisywana jest wartość 21, a customerAge przypisywana jest wartość 23. W instrukcji if zaznaczona jest wartość customerAge < drinkingAgeLimit. Ponieważ 23 < 21 zwraca wartość false, instrukcja else jest wykonywana i w obszarze Debug jest wypisywany limit wieku. Jeśli zmienisz wartość customerAge na 19, customerAge < drinkingAgeLimit zwróci true, więc Under limit wieku zostanie wydrukowany w obszarze Debug. Do tej pory miałeś do czynienia tylko z pojedynczymi przypadkami. A jeśli jest wiele warunków? W tym miejscu pojawiają się instrukcje switch, o których dowiesz się w następnej sekcji.

Korzystanie z instrukcji switch

Aby zrozumieć instrukcje switch, zacznijmy od implementacji instrukcji if z wieloma warunkami. Wyobraź sobie, że programujesz sygnalizację świetlną. Istnieją trzy możliwe warunki sygnalizacji świetlnej - czerwony, żółty lub zielony — i chcesz, aby wydarzyło się coś innego w zależności od koloru światła. Aby to zrobić, możesz połączyć ze sobą wiele instrukcji if. Wykonaj następujące kroki:

1. Dodaj następujący kod do swojego placu zabaw, aby zaimplementować sygnalizację świetlną za pomocą wielu instrukcji if i kliknij przycisk Odtwórz/Zatrzymaj, aby go uruchomić:

```
var trafficLightColor = "Yellow"

if trafficLightColor == "Red" {

print("Stop")

} else if trafficLightColor == "Yellow" {

print("Caution")

} else if trafficLightColor == "Green" {

print("Go")

} else {

print("Invalid color")

}
```

Pierwszy warunek if, trafficLightColor == "Red", zwraca fałsz, więc

Instrukcja else jest wykonywana. Drugi warunek, trafficLightColor == "Żółty", zwraca prawdę, więc Ostrzeżenie jest drukowane w obszarze Debug i nie więcej, jeśli warunki są oceniane. Spróbuj zmienić wartość trafficLightColor, aby zobaczyć różne wyniki. Użyty tutaj kod działa, ale jest trochę trudny do odczytania. W takim przypadku instrukcja switch byłaby bardziej zwięzła i łatwiejsza do zrozumienia. Instrukcja switch wygląda tak:

```
switch value {

case firstValue:

code1

case secondValue:

code2

default:
```

code3

}

Wartość jest sprawdzana i dopasowywana do przypadku, a kod dla tego przypadku jest wykonywany. Jeśli żaden z przypadków nie pasuje, wykonywany jest kod w przypadku domyślnej.

2. Oto jak napisać instrukcję if pokazaną wcześniej jako instrukcję switch. Wpisz następujący kod:

trafficLightColor = "Yellow"

switch trafficLightColor {

case "Red":

print("Stop")

case "Yellow":

print("Caution")

case "Green":

print("Go")

default:

print("Invalid color")

}

Kod tutaj jest znacznie łatwiejszy do odczytania i zrozumienia w porównaniu z poprzednią wersją. Wartość trafficLightColor to „Yellow”, więc przypadek „Yellow” jest dopasowany, a w obszarze Debug wydrukowana jest uwaga. Spróbuj zmienić wartość trafficLightColor, aby zobaczyć różne wyniki. Należy pamiętać o dwóch rzeczach dotyczących instrukcji switch:

• instrukcje switch w języku Swift domyślnie nie przechodzą przez dół każdego przypadku i do następnego. W przykładzie pokazanym wcześniej, po dopasowaniu przypadku „Czerwony” przypadek „Żółty”:, przypadek „Zielony”: i domyślnie: nie zostanie wykonany.

• instrukcje switch muszą obejmować wszystkie możliwe przypadki. W powyższym przykładzie każda wartość trafficLightColor inna niż „Red”, „Yellow” lub „Green” zostanie dopasowana do wartości domyślnej: a Nieprawidłowy kolor zostanie wydrukowany w obszarze Debug.

Na tym kończy się sekcja dotycząca instrukcji if i switch. W następnej sekcji dowiesz się o opcjach, które umożliwiają tworzenie zmiennych bez wartości początkowych, oraz opcjonalnym wiązaniu, które umożliwia wykonanie instrukcji, jeśli opcja opcjonalna ma wartość.

Przedstawiamy opcje i opcjonalne wiązanie

Do tej pory za każdym razem, gdy deklarowałeś zmienną lub stałą, natychmiast przypisywałeś jej wartość. Ale co, jeśli chcesz najpierw zadeklarować zmienną, a później przypisać jej wartość? W takim przypadku użyjesz opcji. Nauczmy się tworzyć i używać opcji oraz zobaczmy, jak są one używane w programie. Wyobraź sobie, że piszesz program, w którym użytkownik musi wpisać imię małżonka. Oczywiście, jeśli użytkownik nie jest żonaty, nie będzie to miało żadnej wartości. W takim przypadku możesz użyć opcjonalnego do reprezentowania nazwiska małżonka. Opcjonalny może mieć jeden z dwóch możliwych stanów. Może zawierać wartość lub nie zawierać wartości. Jeśli opcjonalny zawiera

wartość, możesz uzyskać dostęp do wartości w nim zawartej. Proces uzyskiwania dostępu do wartości opcjonalnej jest znany jako rozpakowywanie opcjonalnego. Zobaczmy, jak to działa. Wykonaj następujące kroki:

1. Dodaj następujący kod do swojego placu zabaw, aby utworzyć zmienną i wydrukować jej zawartość:

var spouseName: String

print(spouseName)

2. Kliknij przycisk Odtwórz/Zatrzymaj, aby go uruchomić. Ponieważ Swift jest bezpieczny dla typu, wyświetli błąd, zmienna „spouseName" używana przed zainicjowaniem.

3. Aby rozwiązać ten problem, możesz przypisać pusty ciąg do spouseName. Zmodyfikuj swój kod, jak pokazano:

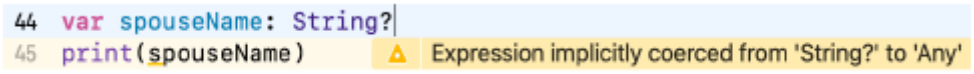
var spouseName: String = ""

To sprawia, że błąd znika, ale pusty ciąg nadal jest wartością, a współmałżonka nie powinna mieć wartości.

4. Ponieważ nazwa współmałżonka nie powinna mieć początkowo wartości, uczyńmy ją opcjonalną. Aby to zrobić, wpisz znak zapytania po adnotacji typu i usuń przypisanie pustego ciągu:

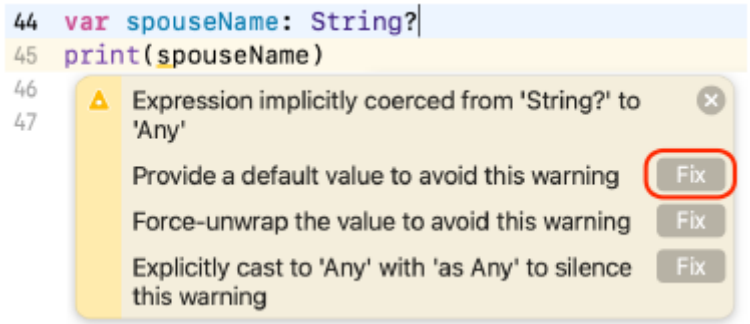
var spouseName: String?

Zobaczysz ostrzeżenie, ponieważ współmałżonka jest teraz opcjonalną zmienną łańcuchową zamiast zwykłej zmiennej łańcuchowej, a instrukcja print() oczekuje zwykłej zmiennej łańcuchowej:



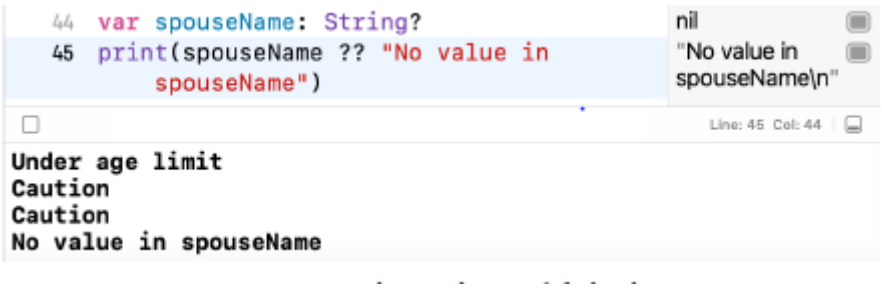
Kliknij przycisk Odtwórz/Zatrzymaj. Nawet jeśli pojawi się ostrzeżenie, program zostanie wykonany. Na razie zignoruj ostrzeżenie. Wartość spouseName jest wyświetlana jako "nil\n" w obszarze Wyniki, a zero jest drukowane w obszarze Debug. nil to specjalne słowo kluczowe, które oznacza, że opcjonalna zmienna spouseName nie ma wartości.

5. Pojawia się ostrzeżenie, ponieważ instrukcja print traktuje współmałżonka jako typu Any zamiast String?. Kliknij żółty trójkąt, aby wyświetlić możliwe poprawki, i wybierz pierwszą poprawkę:



Oświadczenie zmieni się na print(spouseName ?? default value). Zwróć uwagę na użycie ?? operator. To przypisuje wartość domyślną do współmałżonka, jeśli nie zawiera wartości.

6. Zastąp domyślny symbol zastępczy wartością „Brak wartości w współmałżonku”, jak pokazano. Ostrzeżenie zniknie. Uruchom program ponownie, a w obszarze Wyniki pojawi się komunikat „"No value in spouseName":



7. Przypiszmy wartość do współmałżonka. Zmodyfikuj kod, jak pokazano:

var spouseName: String?

spouseName = "Nia"

print(spouseName ?? "No value in spouseName")

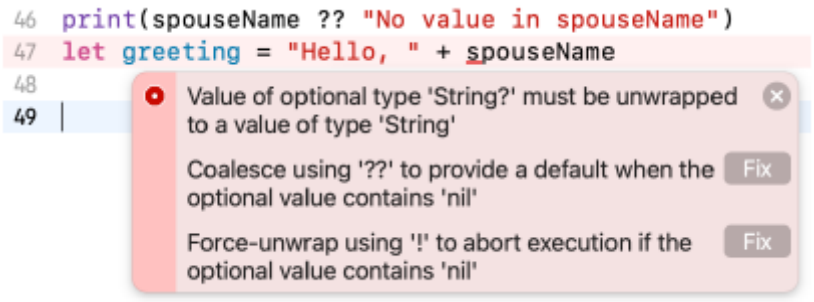
8. Dodaj jeszcze jeden wiersz kodu, aby połączyć współmałżonka z innym ciągiem, jak pokazano:

print(spouseName ?? "No value in spouseName")

let greeting = "Hello, " + spouseName

Otrzymasz błąd, a w obszarze Debugowanie wyświetlane są informacje o błędzie oraz miejsce wystąpienia błędu. Stało się tak, ponieważ nie możesz połączyć zwykłej zmiennej String z opcjonalną za pomocą operatora +. Aby użyć ciągu znaków w opcjonalnym, musisz go najpierw rozpakować.

9. Kliknij czerwone kółko, aby wyświetlić możliwe poprawki, a zobaczysz następujące informacje:



Druga poprawka zaleca wymuszenie rozpakowania w celu rozwiązania tego problemu. Forceunwrapping odpakowuje opcjonalne, niezależnie od tego, czy zawiera wartość, czy nie. Działa dobrze, jeśli współmałżonka ma wartość, ale jeśli współmałżonka ma wartość zero, program się zawiesi.

10. Kliknij drugą poprawkę, a zobaczysz wykrzyknik po nazwie współmałżonka w ostatnim wierszu kodu, co oznacza, że opcja opcjonalny jest wymuszona:

let greeting = "Hello, " + spouseName!

11. Po uruchomieniu programu, Witaj, Nia jest przypisywana do powitania, jak pokazano w obszarze Wyniki. Oznacza to, że pouseName zostało pomyślnie rozpakowane.

12. Aby zobaczyć efekt wymuszenia rozpakowania zmiennej zawierającej nil, ustaw współmałżonka na nil:

spouseName = nil

Twój program ulega awarii i możesz zobaczyć, co spowodowało awarię w obszarze Debug:

Ponieważ współmałżonka ma teraz wartość zero, program uległ awarii podczas próby wymuszenia odpakowania współmałżonka. Lepszym sposobem obsługi tego jest użycie opcjonalnego wiązania. W wiązaniu opcjonalnym próbujesz przypisać wartość w opcjonalnym zmiennej tymczasowej (możesz ją nazwać jak chcesz). Jeśli przypisanie się powiedzie, wykonywany jest blok kodu.

13. Aby zobaczyć efekt opcjonalnego powiązania, zmodyfikuj swój kod w następujący sposób:

spouseName = "Nia"

print(spouseName ?? "No value in spouseName")

if let spouseTempVar = spouseName {

let greeting = "Hello, " + spouseTempVar

print(greeting)

}

Witam, Nia pojawi się w obszarze Debug. Oto jak to działa. Jeśli spouseName ma wartość, zostanie rozpakowana i przypisana do zmiennej tymczasowej spouseTempVar, a instrukcja if zwróci true. Instrukcje znajdujące się w nawiasach klamrowych zostaną wykonane, a stałe powitanie otrzyma wartość Hello, Nia. Następnie Hello, Nia zostanie wydrukowana w obszarze Debug. Zauważ, że tymczasowa zmienna spouseTempVar nie jest opcjonalna.

Jeśli spouseName nie ma wartości, nie można przypisać żadnej wartości do spouseTempVar, a instrukcja if zwróci false. W takim przypadku wyrażenia w nawiasach klamrowych nie zostaną w ogóle wykonane.

14. Aby zobaczyć efekt opcjonalnego powiązania, gdy opcjonalne zawiera zero, przypisz zero jeszcze raz do współmałżonka:

spouseName = nil

Zauważysz, że nic nie pojawia się w obszarze Debug, a twój program już się nie zawiesza, nawet jeśli współmałżonka ma wartość zero. To kończy sekcję dotyczącą opcji i opcjonalnych wiązań, a teraz możesz tworzyć i używać zmiennych opcjonalnych. Niesamowite!

Podsumowanie

Świetnie ci idzie! Nauczyłeś się używać instrukcji if i switch, co oznacza, że możesz teraz pisać własne programy, które robią różne rzeczy w oparciu o różne warunki.

Dowiedziałeś się również o opcjach i opcjonalnym wiązaniu. Oznacza to, że możesz teraz reprezentować zmienne, które mogą mieć wartość lub nie, i wykonywać instrukcje tylko wtedy, gdy wartość zmiennej jest obecna. W następnym rozdziale dowiesz się, jak używać zakresu wartości zamiast pojedynczych wartości oraz jak powtarzać instrukcje programu za pomocą pętli.