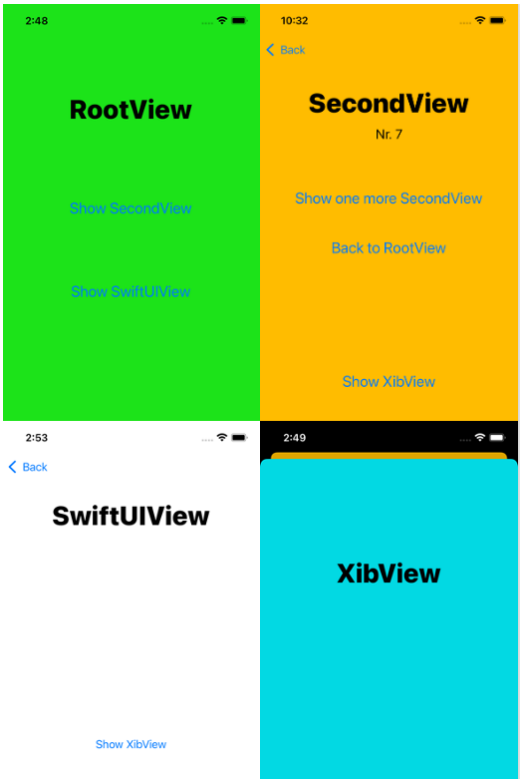
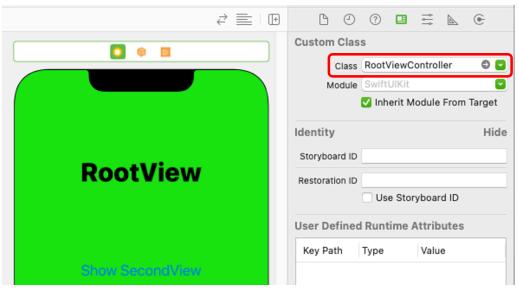


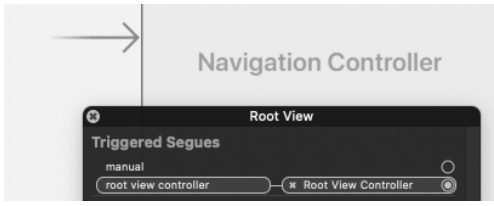
Übung 5: UIKit: Storyboards, XIBs, ViewControllers & SwiftUI-Interoperabilität

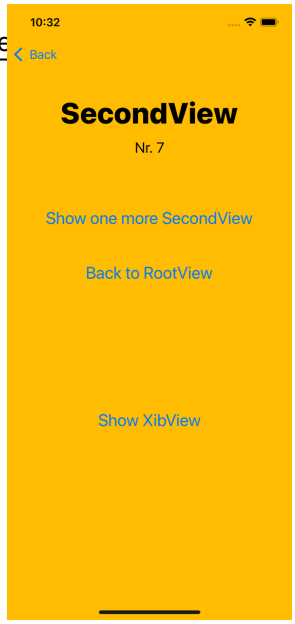
Die Übung beschäftigt sich vertieft mit dem UIKit-Framework, der "alten" UI-Technologie von iOS. Es geht um den Interface Builder, Storyboard- und xib-Dateien, die UIKit-Klasse `UIViewController` und einige Subklassen davon, sowie die Interoperabilität mit der neuen UI-Technologie SwiftUI.

1. **Neues Projekt "SwiftUIKit" & RootView**
   Erstellen sie in Xcode ein neues iOS-Projekt, wählen sie wie gehabt eine App für iOS, Produktname "SwiftUIKit", keine Verwendung von Core Data, und fürs UI ein Storyboard. Die initiale View soll einen grünen Hintergrund haben, mit einem grossen Titel (= UILabel) RootView und zwei Knöpfen "Show SecondView" und "Show SwiftUIView", siehe Screenshot rechts oben. Wie die App aufgebaut ist und die Navigation innerhalb der App funktionieren soll, ist in den aktuellen Folien dargelegt, studieren sie diese entsprechend.



2. **UINavigationController und RootViewController**
   Die ViewController-Klasse hinter der grünen RootView soll neu `RootViewController` heissen. Bennen sie dazu die existierende Klasse `ViewController` um in `RootViewController`. Und stellen sie sicher, dass bei der RootView im "Identity Inspector" diese neue Klasse angeben ist, siehe Screenshot rechts. Da diese App eine hierarchische Navigation verwenden soll, brauchen wir einen `UINavigationController`. Ziehen sie dazu im Storyboard wie in der Vorlesung gezeigt einen `UINavigationController` aus der "Object Library" und setzen Sie diesem also root-view-controller den `RootViewController`. Und setzen sie im Storyboard ebenfalls diesen neuen NavigationController als initiale Scene, siehe dazu das Beispiel aus den Folien. Das soll beim am Schluss im Storyboard aussehen wie auf dem Screenshot rechts.



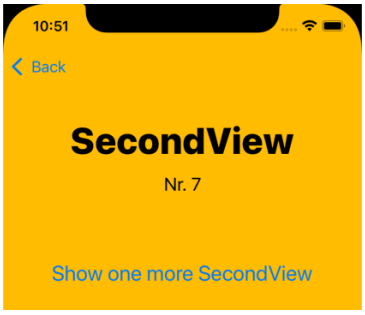


3. **SecondViewController**
   Durch Klicken auf "Show SecondView" soll neu eine andere Szene dargestellt werden. Ziehen Sie dazu im Storyboard aus der "Object Library" einen `UIViewController` neben die RootView. Gestalten sie die SecondView gemäss dem Screenshot rechts. Verknüpfen sie den Knopf "Show SecondView" von der RootView im Storyboard mittels ctrl-Maus mit der SecondView, und zwar als "Action Segue – Show", konsultieren sie ggf. die Folien dazu. Danach sollte in der App durch Drücken auf den entsprechenden Knopf auf der RootView die SecondView erscheinen, inkl. automatisch hinzugefügtem Back-Button links oben.

4. **Weitere SecondViewControllers & zurück zur RootView**
   Durch Drücken von "Show one more SecondView" sollen weitere `SecondViewControllers` dargestellt und auf den Stack vom NavigationController gedrückt werden. Verbinden sie im Storyboard dazu den entsprechenden Knopf mit dem gelben Symbol für den SecondViewController, siehe dazu den roten Pfeil im Screenshot rechts oben. Durch Drücken von "Back to RootView" soll jederzeit direkt auf die RootView zurückgesprungen werden können. Implementieren sie dazu in der Klasse `SecondViewController` eine IBAction-Methode mit folgender Signatur:


```
@IBAction func backToRootViewButtonPressed(_ sender: UIButton)
```

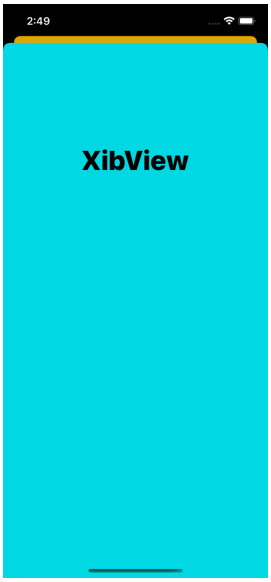
   Oder noch besser: Lassen Sie diese Funktion in der "Assistant View" erzeugen durch ctrl-Maus-Drag vom Storyboard-Knopf in den Code der Klasse `SecondViewController`, konsultieren sie dazu die Folien. In dieser Folie soll auf dem (geerbten!) Property `navigationController` die Methode `popToRootViewController(…)` aufgerufen warden. Und damit kann jederzeit und unabhängig von der Anzahl gezeigten SecondViews auf die RootView zurück gesprungen warden.



5. **Zähler für SecondView**
   Auf der SecondView soll angezeigt werden, die wievielte Instanz diese auf dem Stack vom NavigationController ist. Ziehen sie dazu im Storyboard ein UILabel auf die SecondView Erstellen sie dazu wie in der Vorlesung gesehen in der Klasse `SecondViewController` ein `IBOutlet` auf ein entsprechendes `UILabel` und ergänzen Sie die Klasse um ein Property `number` wie folgt:

```
@IBOutlet weak var numberLabel: UILabel!
var number = 1
```

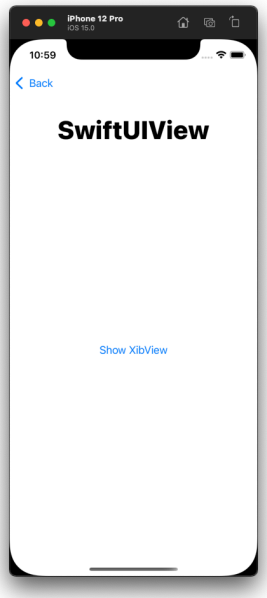
   Implementieren (d.h. überschreiben) sie in der Klasse `SecondViewController` dann die Methode `performSegueWithIdentifier:sender:` so, dass auf der neue Ziel-Instanz (`segue.destination`), die eine Instanz von `SecondViewController` ist, das Property `number` 1 höher gesetzt wird, als dessen Wert in der aktuellen Instanz. Stellen sie sicher, dass der aktuelle Wert von number auf der SecondView auch angezeigt wird. Setzen sie diesen Wert dazu in der (überschriebenen) Methode `viewDidLoad()` der Klasse `SecondViewController`.



6. **Eine weitere Szene: XibView modal anzeigen**
   Erzeugen sie eine neue Klasse `XibViewController` als Subklasse von `UIViewController` und inkl. xib-Datei mit gleichem Name. Gestalten sie in XibViewController.xib die XibView gemäss Screenshot rechts (also eine Hintergrundfarbe setzen und einen grossen Titel setzen). Diese XibView soll neu angezeigt werden durch Drücken von "Show XibView" auf der SecondView. Erstellen und verwenden Sie dazu in der Klasse `SecondViewController` eine Methode mit folgender Signatur:

   ```
   @IBAction func showXibViewButtonPressed(_ sender: UIButton)
   ```

   In dieser soll ein XibViewController instantiiert und mittels der Methode present(…) von UIViewcontroller modal angezeigt werden.



7. **Interop-SwiftUI: SwiftUI@UIKit und UIKit@SwiftUI anzeigen**
   Zum Schluss sollen SwiftUI-Views in einem ViewController und ein ViewController in SwiftUI-Views angezeigt werden. Erzeugen Sie dazu im Projekt eine Datei `SwiftUIContentView.swift` in welcher der ganze SwiftUI-Code hinkommt (vergessen sie dazu nicht, SwiftUI zu importieren). Diese SwiftUIView soll am Schluss aussehen wie im Screenshot rechts. Durch Drücken von "Show XibView" auf der dieser View soll modal eine Instanz vom oben erstellen `XibViewController` angezeigt werden. Verwenden sie dazu wie auf den Folien (und in der Demo) gezeigt das Protokoll `UIViewControllerRepresentable`. Und diese SwiftUIView soll durch Drücken von "Show SwiftUIView" auf der RootView in einem `UIHoistingController` mittels `UIViewController.show(…)` angezeigt werden. Damit haben sie in dieser App diverse UI-Technologien kennen gelernt und zusammen verwendet. ☺