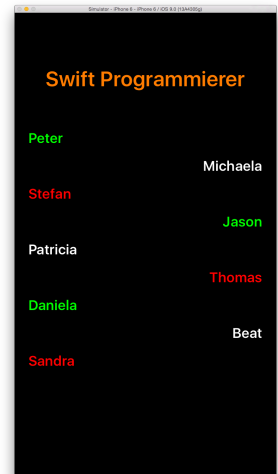
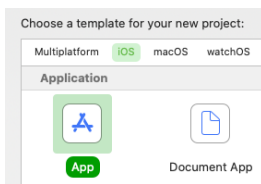


# Übung 1: Ein erstes einfaches iOS-Programm mit Swift

Das Ziel dieser Übung ist es, Swift und Xcode kennen zu lernen. Dazu wird eine erste einfache iPhone-Applikation programmiert. Das Programm soll eine Liste mit den Namen der Kursteilnehmer darstellen. Verwendet wird dazu ein Projekt-Template von Xcode sowie das grafische Element UILabel. Ebenfalls wird eine erste eigene Swift-Klasse definiert und verwendet. Das erste Programm soll analog zur ersten Übung ungefähr wie der Screenshot rechts aussehen.

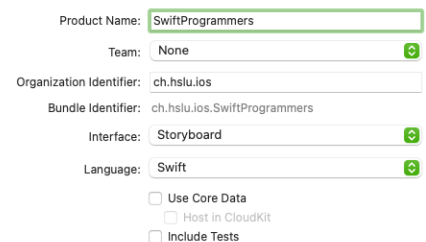


## 1. Neues Projekt "SwiftProgrammers"



Starten sie Xcode und erstellen sie ein neues Projekt für die Plattform "iOS", wählen sie als Application "App", siehe Screenshot links. Nennen sie das Produkt SwiftProgrammers. Optionen auf der nächsten Seite wie im Screenshot rechts, wählen sie hier insbesondere als

Sprache "Swift" und bei Interface "Storyboard". Ob sie ein lokales Git-Repo für ihr Projekt wollen, dürfen sie auf dem nächsten Bildschirm selber entscheiden. Xcode generiert damit automatisch ein Projekt mit diversen vorgegebenen Dateien. Starten sie das Programm im Simulator (Apfel – r). Da wir noch nichts konfiguriert oder programmiert haben, erscheint das weisse Bild einer Default-App.



## 2. Applikationstitel zeichnen

Unter anderem wurde von Xcode eine Swift-Klasse `ViewController` generiert. In dieser Klasse gibt es eine Methode `viewDidLoad()`. Schreiben sie ihren Code in diese Methode. Zuerst sollen sie einen Titel auf den Bildschirm zeichnen, z.B. "Swift-Programmierer". Verwenden sie dazu ein `UILabel` und setzen sie diesem die gewünschten Properties (text, font, textAlignment, ...). UILabels werden in Swift wie folgt instanziiert:

```
let label = UILabel(frame: CGRect(x: ?, y: ?, width: ?, height: ?))
```

Hier gibt (x, y) die Koordinaten links oben an und (width, height) Breite und Höhe des Labels. Hier die Bildschirmgrößen in (programmatischen, mehr dazu später) Pixeln: iPhone 5 (320x568), iPhone 6-8 (375x667), iPhone 6-8 Plus (540x960), iPhone X (375x812). Zum Bildschirm hinzugefügt wird ein UILabel label mittels `self.view.addSubview(label)`

## 3. Daten vom "DataProvider"

Schreiben sie in einer ersten Version eine neue Klasse `DataProvider`, welche als Property `memberNames` vom Typ `Array` hat. Dieser Array soll für jeden Kursteilnehmer je ein `String`-Objekt mit dessen Vornamen enthalten. Der Array soll in der `init`-Methode von `DataProvider` mit Vornamen (`String`-Objekten) gefüllt werden. Array-Literale können in Swift wie folgt angegeben werden:

```
[obj1, obj2, obj3, ...]
```

#### 4. Label pro Teilnehmer zeichnen

Instanzieren sie in `viewDidLoad()` von `ViewController` einen `DataProvider` und zeichnen sie für jeden Eintrag im Array `memberNames` ein Label. Fügen sie dazu der Klasse `ViewController` eine neue Methode mit folgender Signatur hinzu:

```
func addLabel(name: String, position: Int)
```

Diese Methode wird mit jedem Element vom Array `memberNames` aus dem `DataProvider` aufgerufen und zeichnet ein entsprechendes Label hin. Berechnen sie also anhand des Arguments `position` (von 0..n) die Koordinaten, an welche das Label hin kommt. Setzen sie mittels einer switch-case-Anweisung drei verschiedene Farben für die Labels, abhängig vom Argument `position`. Damit haben sie ein erstes iPhone-Programm geschrieben, welches den Bildschirm mit einem Titel und farbigen Namen erzeugt, analog zum Screenshot rechts oben auf der ersten Seite von dieser Aufgabenstellung. ☺

#### 5. Neue Klasse Person

Anstelle von Namen (`String`-Objekten) soll der `DataProvider` neu zusätzlich ebenfalls `Person`-Objekte verwalten. Implementieren sie dazu den notwendigen Rest zu diesem Code-Fragment einer Klasse `Person`:

```
class Person {
    var firstName : String
    var lastName : String
    var plz : Int

    init(firstName: String, lastName: String, plz: Int)
```

#### 6. Person verwenden im DataProvider

Erweitern sie die Klasse `DataProvider` so, dass sie neu zusätzlich ein Property `memberPersons` hat, einen Array mit Referenzen auf `Person`-Objekte. (Das Property `memberNames` von `DataProvider` mit `String`-Objekten bleibt daneben weiterhin bestehen!) Die Klasse `ViewController` soll neu diese `Person`-Objekte auch entsprechend ausgeben, siehe Bild rechts. Implementieren sie dazu in `ViewController` eine neue Methode mit folgender Signatur:

```
func addLabel(person: Person, position: Int)
```

Ob die `memberPersons`- oder die `memberNames`-Objekte angezeigt werden, soll in der Klasse `ViewController` mittels `BOOL`'scher Konstante `usePersonData` festgelegt werden können.

#### 7. DataProvider als Singleton

Vom `DataProvider` soll es in unserer Applikation immer nur eine Instanz geben, `DataProvider` soll also ein Singleton sein. Implementieren sie die dazu notwendige Funktionalität, so dass mittels `DataProvider.sharedInstance` auf diese (einzige!) Singleton-Instanz der Klasse `DataProvider` zugegriffen werden kann. Passen sie danach den Code in `ViewController` so an, dass dieser Singleton verwendet wird.

