

UNIVERSITÄT HEIDELBERG

SOMMERSEMESTER 2014

Softwareentwicklung für iOS mit Objective-C und Xcode

BEISPIELLÖSUNGEN DER ÜBUNGSAUFGABEN

NILS FISCHER

Aktualisiert am 11. Mai 2014
Begleitende Dokumente auf der Vorlesungsseite:
<http://ios-dev-kurs.github.io>

Inhaltsverzeichnis

1	Fibonacci	2
2	Primzahlen	2
3	Scientists	3
4	Emails	5
5	Scientists 2	7
6	Simple UI	8
6.1	Counter App	8
6.2	BMI App	9
6.3	RGB App	10

1 Fibonacci

```

1  int a=1; // erstes Folgenglied
2  int b=2; // zweites Folgenglied
3  int di=0; // Abstand zur letzten geraden Zahl
4  while ((a+b)<1000) { // die Schleife wird ausgeführt, bis die nächste
    Fibonaccizahl zu groß wird
5      int c = a+b;
6      NSLog(@"%i",c);
7      a = b;
8      b = c;
9      di++;
10     if (c%2==0) { // Ausdruck ergibt YES für gerade Zahlen
11         NSLog(@"Zahl gerade, Abstand zur vorigen geraden Fibonaccizahl: %i", di)
12         ;
13         di = 0;
14     }
15 }
```



2 Primzahlen

```

1  // prime numbers
2  for (int i=1; i<1000; i++) { // loop through all numbers from 1 to 1000
3      BOOL isPrime = YES;
4      for (int j=2; j<i-1; j++) { // check the division rest with every number
        from 2 up to (excluding) the current number and break when a divisor
        was found
5          if (i%j==0) {
6              isPrime = NO;
7              break;
8          }
9      }
10     if (isPrime) { // if isPrime is still YES here, no divisor was found
```

```
11     NSLog(@"Prime number found: %i", i);
12 }
13 }
```

3 Scientists

1. In Xcode könnt ihr eine neue Klasse mit  +  erstellen. Nennt die Klasse `Scientist` und wählt die Superklasse `Person`. Die Main- und Header-Datei erscheinen nach dem Speichern im Project Navigator.
2. Im Interface in der Header-Datei wird die neue Methode zunächst definiert:

```
1 #import "Person.h"
2
3 @interface Scientist : Person
4
5 - (void)sayPrimeNumbersUpTo:(int)number;
6
7 @end
```

Anschließend können wir sie in der Main-Datei implementieren:

```
1 #import "Scientist.h"
2
3 @implementation Scientist
4
5 - (void)sayPrimeNumbersUpTo:(int)number {
6     // Algorithmus aus Aufgabe 'Primzahlen'
7 }
8
9 @end
```

In der nun schon häufiger verwendeten `application:didFinishLaunchingWithOptions` : Methode in der `AppDelegate.m`-Datei, wollen wir die neue Klasse ausprobieren:

```
1 #import "AppDelegate.h"
2
3 #import "Scientist.h"
4
5 @implementation AppDelegate
6
7 - (BOOL)application:(UIApplication *)application
8     didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
9
10     Scientist *aScientist = [[Scientist alloc] init];
11     [aScientist sayPrimeNumbersUpTo:100];
12
13     return YES;
14 }
15 @end
```

Führen wir die App aus, werden alle Primzahlen bis 100 in der Konsole ausgegeben.

3. Mit Objekten des Typs `Scientist` kann genauso verfahren werden wie mit Objekten des Typs `Person`, da sie voneinander abstammen und daher alle Attribute und Methoden erben. Schreibt also in der `AppDelegate.m`-Datei:

```

1  #import "AppDelegate.h"
2
3  #import "Scientist.h"
4
5  @implementation AppDelegate
6
7  - (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
8
9      Scientist *aScientist = [[Scientist alloc] init];
10     aScientist.name = @"Alice";
11     [aScientist sayHello];
12
13     return YES;
14 }
15
16 @end

```

Es wird wieder ausgegeben:

```

1  Hello World! My name is Alice.

```

4. Die `Scientist`-Klasse besitzt bereits die Methode `sayHello`, da sie in ihrer Superklasse `Person` definiert und implementiert wird. Um sie zu überschreiben, müssen wir sie nur in der `Main`-Datei erneut implementieren:

```

1  #import "Scientist.h"
2
3  @implementation Scientist
4
5  - (void)sayPrimeNumbersUpTo:(int)number {
6      // Primzahlen-Algorithmus
7  }
8
9  - (void)sayHello {
10     NSLog(@"Hello World! My name is %@. I know all Prime Numbers.", self.
        name);
11 }
12
13 @end

```

Nun wird statt der Methodenimplementierung in der Superklasse 'Person' diese Neuimplementierung verwendet und der zusätzliche Text wird ausgegeben, wenn die Methode `sayHello` von einem Objekt der Klasse 'Scientist' aufgerufen wird:

```

1  Hello World! My name is Alice. I am a Scientist.

```

Da wir der Implementierung der Superklasse im Prinzip nur etwas hinzufügen wollen, ist es eleganter, stattdessen erst diese aufzurufen und dann den zusätzlichen Code auszuführen:

```

1 - (void)sayHello {
2     [super sayHello]; // Die Implementierung Superklasse wird aufgerufen
3     NSLog(@"I know all Prime Numbers.");
4 }

```

Ausgabe:

```

1 Hello World! My name is Alice.
2 I know all Prime Numbers.

```

4 Emails

Email.h

```

1 @interface Email : NSObject
2
3 @property (strong, nonatomic) NSMutableArray *allRecipients;
4
5 - (void)sendTo:(NSArray *)recipients;
6
7 @end

```

Email.m

```

1 #import "Email.h"
2 #import "Person.h"
3
4 @implementation Email
5
6 - (void)sendTo:(NSArray *)recipients {
7     for (Person *person in recipients) {
8         // treat email as spam and skip if it has already been sent to this
8         // person before
9         if ([self.allRecipients containsObject:person]) continue;
10        // add person to list of all recipients
11        if (![self.allRecipients self.allRecipients = [[NSMutableArray alloc]
12            init];
13            [self.allRecipients addObject:person];
14            // deliver email
15            [person receiveEmail:self];
16        }
17    }
18 @end

```

Person.h

```

1 @class Email; // Forward Declaration
2
3 @interface Person : NSObject
4
5 @property (strong, nonatomic) NSString *name;
6

```

```

7 - (void)makeFriendsWith:(Person *)person;
8
9 - (void)sendEmail;
10 - (void)receiveEmail:(Email *)email;
11
12 @end

```

Person.m

```

1 #import "Person.h"
2 #import "Email.h"
3
4 @interface Person () // private interface
5
6 @property (strong, nonatomic) NSMutableArray *friends;
7
8 @end
9
10 @implementation Person
11
12 - (void)makeFriendsWith:(Person *)person {
13     // make sure to skip connection if already existent or redundant
14     if (!person || person == self || [self.friends containsObject:person]) return;
15     // create friends array if not existent yet and add person
16     if (!self.friends) self.friends = [[NSMutableArray alloc] init];
17     [self.friends addObject:person];
18     // trigger reverse connection
19     [person makeFriendsWith:self];
20     NSLog(@"%@ <-> %@", self.name, person.name);
21 }
22
23 - (void)sendEmail {
24     Email *newEmail = [[Email alloc] init];
25     [newEmail sendTo:self.friends];
26 }
27
28 - (void)receiveEmail:(Email *)email {
29     NSLog(@"%@ received an Email.", self.name);
30     [email sendTo:self.friends];
31 }
32
33 @end

```

AppDelegate.m

```

1 #import "AppDelegate.h"
2 #import "Person.h"
3
4 @implementation AppDelegate
5
6 - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions
    :(NSDictionary *)launchOptions {
7
8     Person *me = [[Person alloc] init];
9     me.name = @"Nils";
10
11     NSMutableArray *persons = [[NSMutableArray alloc] init];

```

```

12
13     NSArray *names = @[@"Alice", @"Bob", @"Cindy", @"Bruce", @"Chris", @"Bill "
14         , @"Susan"];
15     for (NSString *name in names) {
16         Person *newPerson = [[Person alloc] init];
17         newPerson.name = name;
18
19         [persons addObject:newPerson];
20         [me makeFriendsWith:newPerson];
21     }
22     // befriend persons with same initial letter
23     for (Person *person in persons) {
24         for (Person *other in persons) {
25             if ([person.name characterAtIndex:0]==[other.name characterAtIndex
26                 :0]) {
27                 [person makeFriendsWith:other];
28             }
29         }
30     }
31     // start simulation
32     [me sendEmail];
33
34     return YES;
35 }
36
37 @end

```

5 Scientists 2

Die Methode `sayHello` muss nicht mehr implementiert werden, denn eine Überschreibung der Superklassenimplementierung ist nicht mehr notwendig. Diese ruft nämlich nur die Methode `helloString` auf und gibt den Rückgabewert in der Konsole aus. Stattdessen müssen wir also `helloString` überschreiben, damit der Zusatz *I know all Prime Numbers.* ausgegeben wird:

```

1 - (NSString *)helloString {
2     NSString *greeting = [super helloString];
3     greeting = [greeting stringByAppendingString:@" I am a Scientist."];
4     return greeting;
5 }

```

Oder in Kurzform:

```

1 - (NSString *)helloString {
2     return [[super helloString] stringByAppendingString:@" I am a Scientist."
3         ];
3 }

```

6 Simple UI

6.1 Counter App

1. Erstellen wir zunächst das User Interface. Wählt eine der Storyboard-Dateien im Project Navigator, entsprechend des Geräts, für das ihr das UI konfigurieren möchtet.
2. Zieht zwei Labels und drei Buttons aus der Object Library unten im Inspektor in die erste und einzige Scene und konfiguriert sie mit dem Inspektor (s. S. 8, Abb. 1).

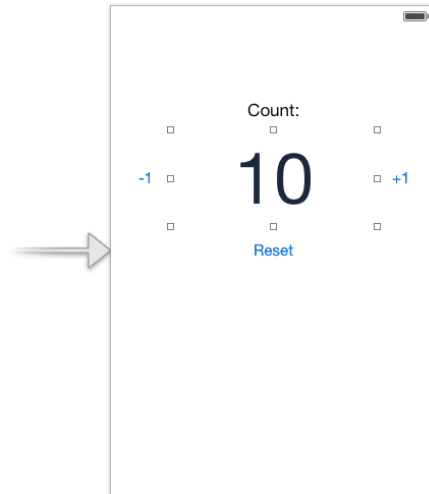


Abbildung 1: Einige Labels und Buttons sind für diese einfache App ausreichend

3. Wechselt in den Assistant-Editor, sodass die Datei *ViewController.m* rechts angezeigt wird. Wählt diese wenn nötig in der Jump bar des Assistant-Editors unter **Automatic** aus.
4. Definiert eine Property im privaten Interface, die zur Laufzeit der App die Referenz zum Label hält. Zusätzlich kann hier direkt die Property `int` `count` definiert werden, da diese ebenfalls nicht öffentlich sein muss. Definiert außerdem die benötigten Methoden:

```

1  @interface ViewController ()
2
3  @property (nonatomic) int count;
4
5  @property (strong, nonatomic) IBOutlet UILabel *countLabel;
6
7  - (void)updateLabel;
8
9  - (IBAction)increaseButtonPressed:(id)sender;
10 - (IBAction)decreaseButtonPressed:(id)sender;
11 - (IBAction)resetButtonPressed:(id)sender;
12
13 @end

```


5. Zieht mit gedrückter `ctrl`-Taste eine Verbindung zwischen Label und Property, sowie zwischen Buttons und Methoden.
6. Implementiert die Methoden in der Main-Datei:

```

1 - (void)updateLabel {
2     self.countLabel.text = [NSString stringWithFormat:@"%i", self.count];
3 }
4
5 - (IBAction)increaseButtonPressed:(id)sender {
6     self.count++;
7     [self updateLabel];
8 }
9 - (IBAction)decreaseButtonPressed:(id)sender {
10    self.count--;
11    [self updateLabel];
12 }
13 - (void)resetButtonPressed:(id)sender {
14     self.count = 0;
15     [self updateLabel];
16 }

```

7. Build & Run !

6.2 BMI App

Analog zur Counter App erstellen wir ein neues Projekt und konfigurieren das User Interface im Storyboard (s. S. 9, Abb. 2).

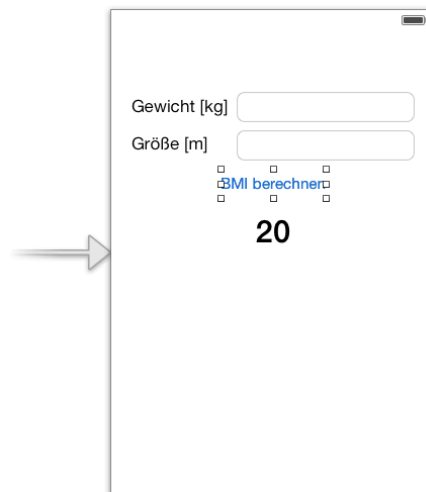


Abbildung 2: Achtet bei der UI Gestaltung darauf, dass die Interfaceelemente nicht von der erscheinenden Tastatur verdeckt werden

ViewController.m

```

1 #import "ViewController.h"
2

```

```

3  @interface ViewController ()
4
5  @property (strong, nonatomic) IBOutlet UITextField *gewichtTextfield;
6  @property (strong, nonatomic) IBOutlet UITextField *groesseTextfield;
7  @property (strong, nonatomic) IBOutlet UILabel *bmiLabel;
8
9  - (IBAction)berechneButtonPressed:(id)sender;
10
11 @end
12
13 @implementation ViewController
14
15 - (IBAction)berechneButtonPressed:(id)sender {
16     float m = [self.gewichtTextfield.text floatValue];
17     float l = [self.groesseTextfield.text floatValue];
18     float bmi = m/l/l;
19     self.bmiLabel.text = [NSString stringWithFormat:@"%0.1f", bmi];
20 }
21
22 @end

```

6.3 RGB App

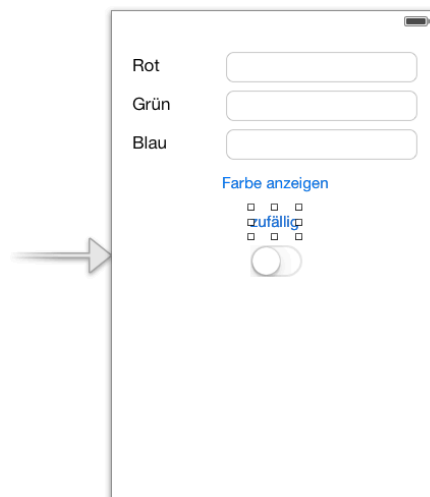


Abbildung 3: Die RGB App zeigt Eingabefelder für die Farbkomponenten der Hintergrundfarbe

ViewController.m

```

1  #import "ViewController.h"
2
3  @interface ViewController ()
4
5  @property (strong, nonatomic) IBOutlet UITextField *rTextfield;
6  @property (strong, nonatomic) IBOutlet UITextField *gTextfield;
7  @property (strong, nonatomic) IBOutlet UITextField *bTextfield;
8

```

```
9  @property (strong, nonatomic) NSTimer *randomTimer;
10
11  - (IBAction)farbeAnzeigenButtonPressed:(id)sender;
12  - (IBAction)randomButtonPressed:(id)sender;
13  - (IBAction)switchValueChanged:(UISwitch *)sender;
14
15  @end
16
17  @implementation ViewController
18
19  - (IBAction)farbeAnzeigenButtonPressed:(id)sender {
20      float r = [self.rTextfield.text floatValue];
21      float g = [self.gTextfield.text floatValue];
22      float b = [self.bTextfield.text floatValue];
23      self.view.backgroundColor = [UIColor colorWithRed:r/255. green:g/255. blue:
24          b/255. alpha:1.];
25
26  - (IBAction)randomButtonPressed:(id)sender {
27      self.rTextfield.text = [NSString stringWithFormat:@"%i", arc4random_uniform
28          (256)];
29      self.gTextfield.text = [NSString stringWithFormat:@"%i", arc4random_uniform
30          (256)];
31      self.bTextfield.text = [NSString stringWithFormat:@"%i", arc4random_uniform
32          (256)];
33      [self farbeAnzeigenButtonPressed:sender];
34
35  - (IBAction)switchValueChanged:(UISwitch *)sender {
36      if (sender.isOn) {
37          self.randomTimer = [NSTimer scheduledTimerWithTimeInterval:0.1 target:
38              self selector:@selector(randomButtonPressed:) userInfo:nil repeats
39              :YES];
40      } else {
41          [self.randomTimer invalidate];
42          self.randomTimer = nil;
43      }
44  }
```