Softwareentwicklung für iOS mit Objective-C und Xcode

Beispiellösungen der Übungsaufgaben

Universität Heidelberg

Sommersemester 2014

Nils Fischer

Aktualisiert am 5. 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**

1 Fibonacci

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

2 Primzahlen

```
// prime numbers
for (int i=1; i<1000; i++) { // loop through all numbers from 1 to 1000
   BOOL isPrime = YES;
   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
      if (i%j==0) {
         isPrime = NO;
         break;
      }
   }
   if (isPrime) { // if isPrime is still YES here, no divisor was found
      NSLog(@"Prime number found: %i", i);
   }
}
```

3 Scientists

1. In Xcode könnt ihr eine neue Klasse mit ⌘+N 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:

```
#import "Person.h"

@interface Scientist : Person

- (void)sayPrimeNumbersUpTo:(int)number;

@end
```

   Anschließend können wir sie in der Main-Datei implementieren:

```
#import "Scientist.h"

@implementation Scientist

- (void)sayPrimeNumbersUpTo:(int)number {
   // Algorithmus aus Aufgabe 'Primzahlen'
}

@end
```

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

```
#import "AppDelegate.h"

#import "Scientist.h"

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application
      didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

   Scientist *aScientist = [[Scientist alloc] init];
   [aScientist sayPrimeNumbersUpTo:100];

   return YES;
}

@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:

```
#import "AppDelegate.h"

#import "Scientist.h"

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    Scientist *aScientist = [[Scientist alloc] init];
    aScientist.name = @"Alice";
    [aScientist sayHello];

    return YES;
}

@end
```

Es wird wieder ausgegeben:

```
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:

```
#import "Scientist.h"

@implementation Scientist

- (void)sayPrimeNumbersUpTo:(int)number {
    // Primzahlen-Algorithmus
}

- (void)sayHello {
    NSLog(@"Hello World! My name is %@. I know all Prime Numbers.", self.
        name);
}

@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:

```
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:

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

Ausgabe:

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

4 Emails

Email.h

```
@interface Email : NSObject

@property (strong, nonatomic) NSMutableArray *allRecipients;

- (void)sendTo:(NSArray *)recipients;

@end
```

Email.m

```
#import "Email.h"
#import "Person.h"

@implementation Email

- (void)sendTo:(NSArray *)recipients {
    for (Person *person in recipients) {
        // treat email as spam and skip if it has already been sent to this
            person before
        if ([self.allRecipients containsObject:person]) continue;
        // add person to list of all recipients
        if (!self.allRecipients) self.allRecipients = [[NSMutableArray alloc]
            init];
        [self.allRecipients addObject:person];
        // deliver email
        [person receiveEmail:self];
    }
}

@end
```

Person.h

```
@class Email; // Forward Declaration

@interface Person : NSObject

@property (strong, nonatomic) NSString *name;

- (void)makeFriendsWith:(Person *)person;

- (void)sendEmail;
- (void)receiveEmail:(Email *)email;

@end
```

Person.m

```objc
#import "Person.h"
#import "Email.h"

@interface Person () // private interface

@property (strong, nonatomic) NSMutableArray *friends;

@end

@implementation Person

- (void)makeFriendsWith:(Person *)person {
    // make sure to skip connection if already existent or redundant
    if (!person||person==self||[self.friends containsObject:person]) return;
    // create friends array if not existent yet and add person
    if (!self.friends) self.friends = [[NSMutableArray alloc] init];
    [self.friends addObject:person];
    // trigger reverse connection
    [person makeFriendsWith:self];
    NSLog(@"%@ <-> %@", self.name, person.name);
}

- (void)sendEmail {
    Email *newEmail = [[Email alloc] init];
    [newEmail sendTo:self.friends];
}

- (void)receiveEmail:(Email *)email {
    NSLog(@"%@ received an Email.", self.name);
    [email sendTo:self.friends];
}

@end
```

AppDelegate.m

```objc
#import "AppDelegate.h"
#import "Person.h"

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions
    :(NSDictionary *)launchOptions {

    Person *me = [[Person alloc] init];
    me.name = @"Nils";

    NSMutableArray *persons = [[NSMutableArray alloc] init];

    NSArray *names = @[@"Alice", @"Bob", @"Cindy", @"Bruce", @"Chris", @"Bill "
        , @"Susan"];
    for (NSString *name in names) {
        Person *newPerson = [[Person alloc] init];
        newPerson.name = name;

        [persons addObject:newPerson];
```

```
19          [me makeFriendsWith:newPerson];
20      }
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
                        :0]) {
26                  [person makeFriendsWith:other];
27              }
28          }
29      }
30
31      // start simulation
32      [me sendEmail];
33
34      return YES;
35  }
36
37  @end
```