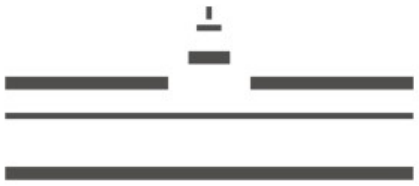


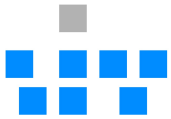
Westfälische
Wilhelms-Universität
Münster

Programmierung für mobile Endgeräte

Einleitung und Grundlagen

wissen.leben
WWU Münster

Name: Patrick Förster



ZENTRUM FÜR
INFORMATIONS
VERARBEITUNG

Organisatorisches

- Dozent:
  - Patrick Förster
  - patfoerster@uni-muenster.de
  - Einsteinstraße 60, Raum 104
  - 0251 83 31835

- Vorlesung ist nicht in den „Allgemeinen Studien“ angemeldet
- Keine Klausur, keine Creditpoints
- Evtl. Anrechnen der Vorlesung bitte mit dem jeweiligen Prüfungsamt absprechen
- Anwesenheitsschein, falls maximal nicht mehr als zwei mal gefehlt

Ziel

- Entwicklung einer Applikation für ein mobiles Endgerät der iOS-Familie
- Dazu gibt es eine Einführung in
  - die objektorientierte Programmiersprache Objective-C
  - die Entwicklungsumgebung Xcode
  - das Cocoa-Framework

- Die Vorlesung setzt keinerlei Kenntnisse in den obigen Themen voraus
- Des weiteren ist auch der Besitz eines iOS-Gerätes kein Muss
- Grundlagen in objektorientierter Programmierung werden allerdings vorausgesetzt (Klassen, Methoden, Polymorphie, ...)

Literatur

- Cocoa Programming for Mac OS X – Aaron Hillegass – Addison Wesley – ISBN: 978-0321-50361-9 [aaronh]
- Objective-C und Cocoa: Praxiseinstieg – Holger Hinzberg – mitp – ISBN: 978-3826-69085-3 [holgerh]
- iPhone-Apps entwickeln – Dr. Dirk Koller – Franzis – ISBN: 978-3645-60001-9 [dirkk]

- Apple iOS API: https://developer.apple.com/library/ios/navigation/index.html

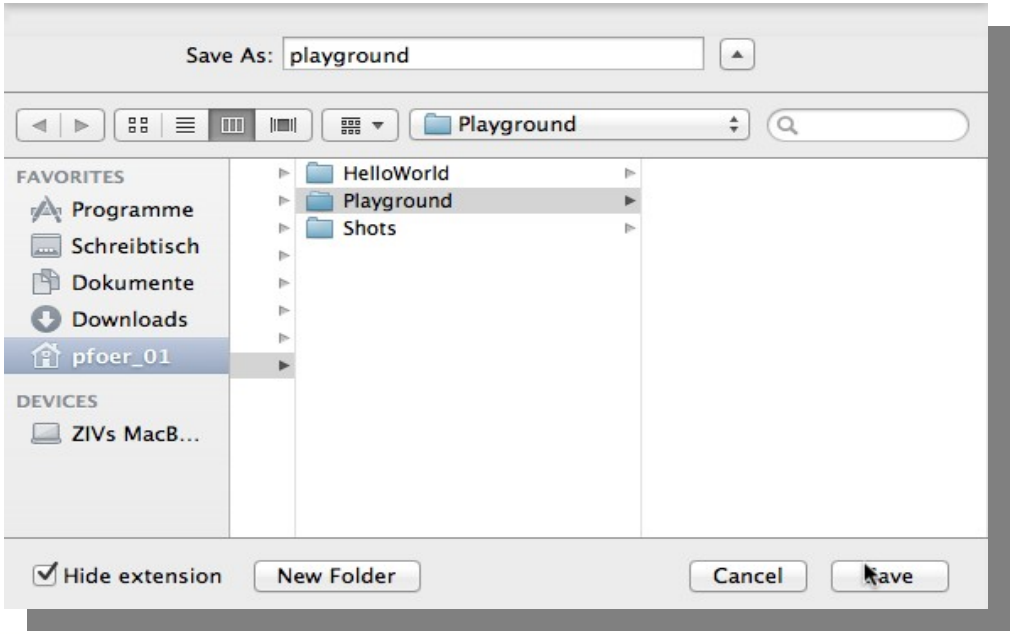
Xcode: Installation

- Xcode 4.3.x aus dem AppStore herunterladen: http://itunes.apple.com/us/app/xcode/id497799835?ls=1&mt=12
- AppleID notwendig
- „Lion“ notwendig
- Kostenlos

- Unter http://connect.apple.com können ältere Versionen geladen werden
- Developer License erforderlich:
  - https://developer.apple.com/programs/register/
  - Kostenlos, allerdings
    - iOS-Entwicklung nur auf Simulatoren
    - Kein Ausführen und Test auf „echten“ Geräten

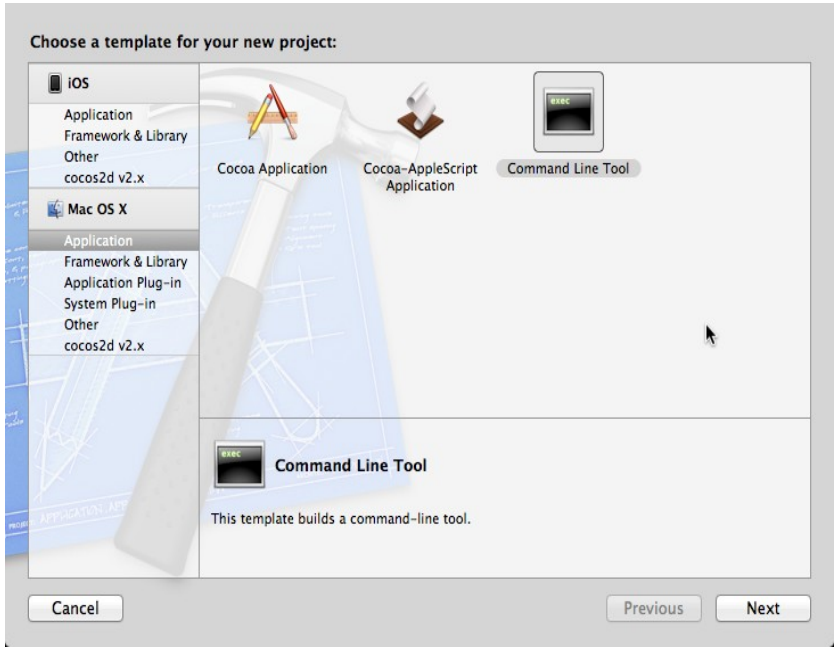
Xcode: Erste Schritte

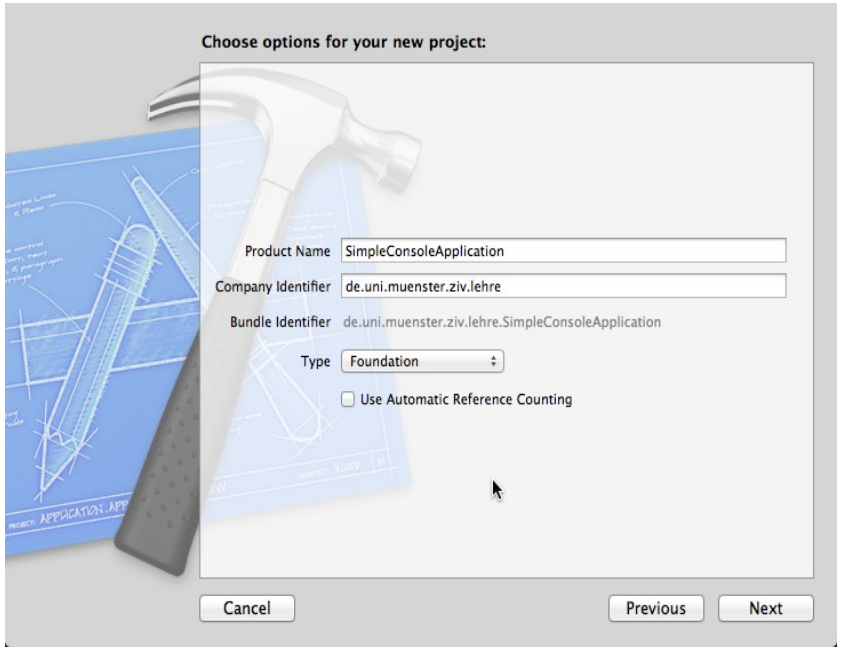
- Das „Welcome to Xcode“ für das erste ignorieren und per File→New→New Workspace einen neuen Arbeitsbereich erstellen
- In einem Arbeitsbereich können mehrere Projekte zusammengefasst und weitere projektbezogene Ressourcen verwaltet werden



Xcode: Erste Schritte

- Als nächstes wird für die Einführung in Objective-C ein „Command Line Tool“ Projekt erstellt
- Ein „Command Line Tool“ ist eine Mac OS- Anwendung auf Konsolen-Ebene

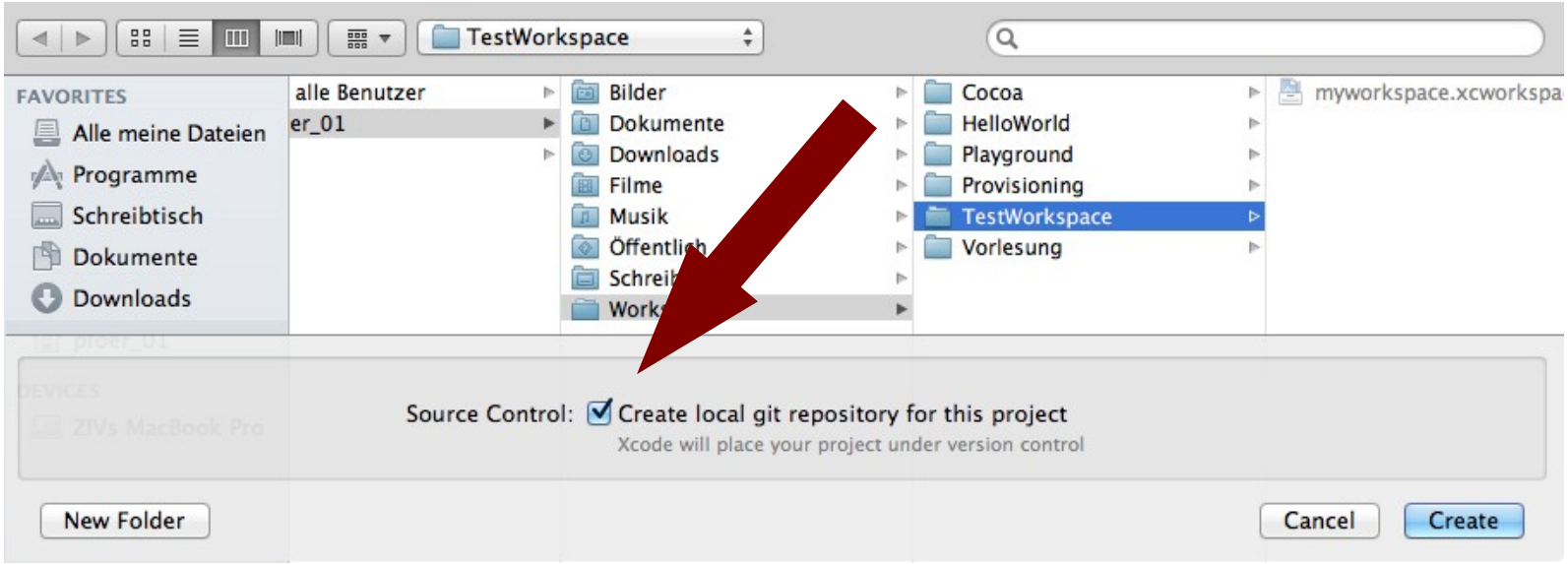




- Vorerst „Use Automatic Reference Counting“ deaktiveren (dazu später mehr)!

Xcode: Erste Schritte

- Source Control Managment (SCM): Xcode bietet von Haus aus die Möglichkeit den eigenen Code lokal mit Git (http://git-scm.com/) zu managen:



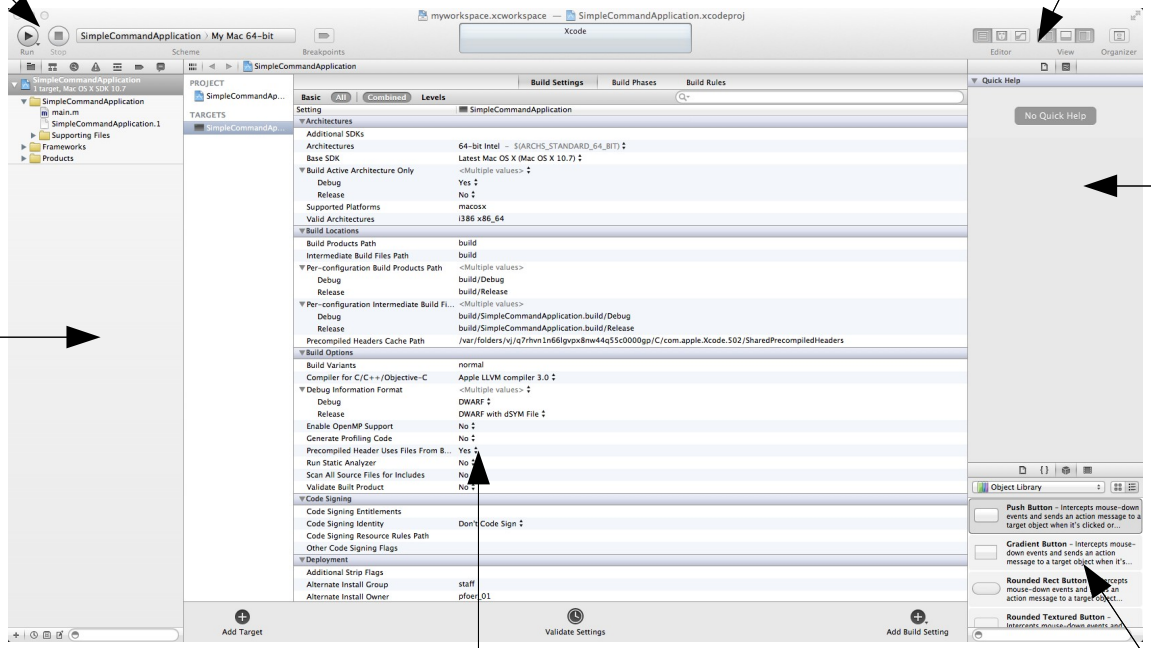
- Subversion wird ebenfalls unterstützt, allerdings nicht für die lokale Verwaltung
- Alternative können auch sog. „Snapshots“ vom aktuellen Stand erzeugt werden

Xcode: Erste Schritte

- Auf den ersten Blick:

Starten/Stoppen

Layout



Kurzhilfe

Navigatoren:

- Code
- Fehler
- Meldungen
- Suchen
- ...

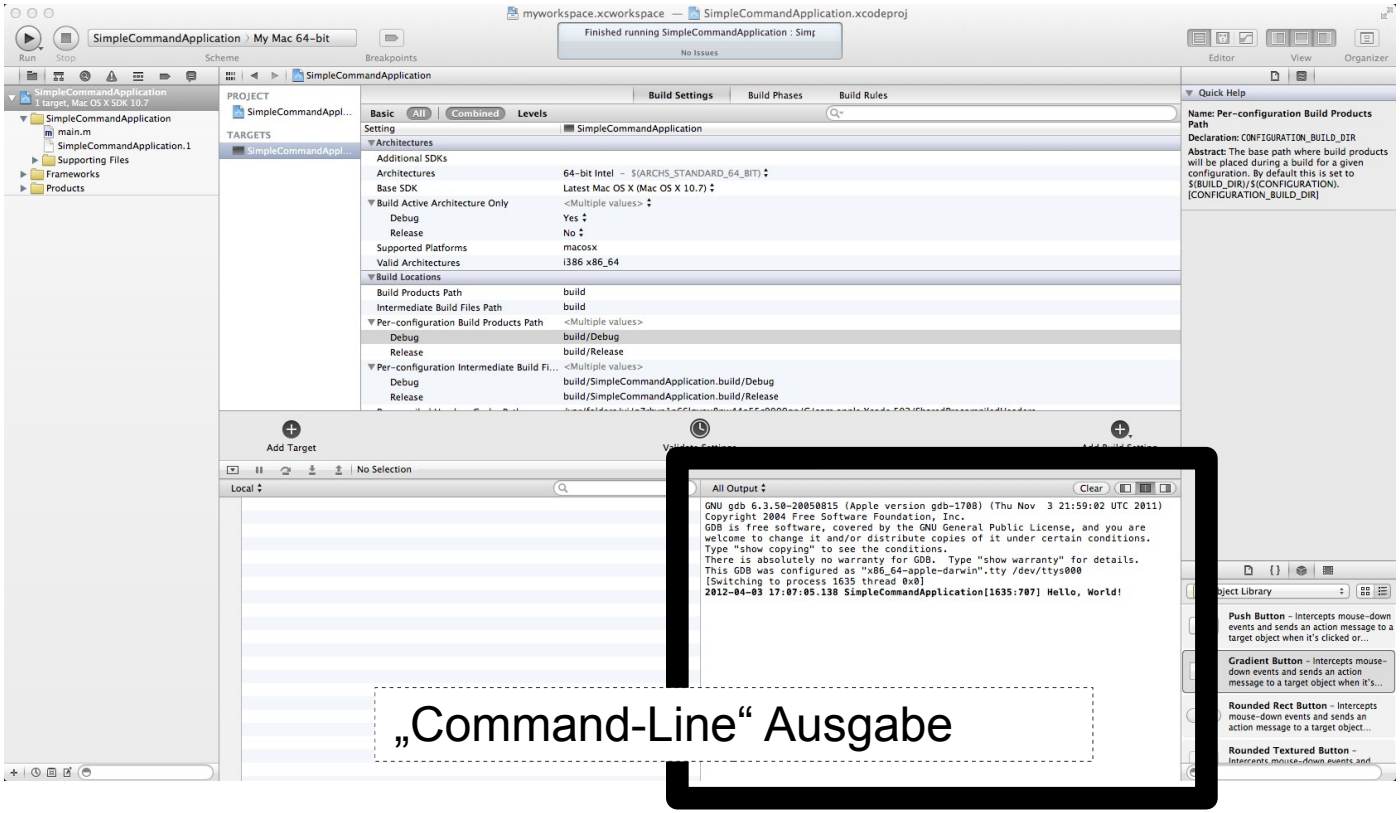
Editor/Infos

GUI-Bausteine

wissen.leben WWU Münster

Xcode: Erste Schritte

- Mit Run das erste Programm ausführen:



Die Geschichte von Cocoa

- Entwickelt durch NeXT-Computer/-Software, Inc
- 1986 gegründet von Steve Jobs
- NeXT-Computer entwickelte bis 1992 auch Hardware
- Betriebssystem: *NeXTSTEP* mit Unix-Kern
- Später kam zum Betriebssystem ein Window-Server hinzu
- *NeXTSTEP* wurde mit einer Reihe von „Frameworks“ ausgeliefert, um Programmierern den Umgang mit dem Window-Server zu erleichtern
- Diese Frameworks wurde dann unter dem Namen *OpenStep* zusammengefasst
- Programmiert wurde für das Framework in *Objective-C*, einer objekt-orientierten Erweiterung der Sprache C
- Im Gegensatz C++ ist *Objective-C* „schwach Typisiert“
- 1996 wurde *NeXT* von Apple übernommen, die *NeXTSTEP* als Basis für ein neues Betriebssystem nutzen wollten
- Aus *OpenStep* wurde *Cocoa*

[aaronh]

Objective-C: Nachrichten

- Nachrichten sind gewissermaßen das Äquivalent zu Methodenaufrufen in anderen OOP
- Im Prinzip sind Nachrichten der an ein Objekt gerichtete Wunsch eine bestimmte Methode auszuführen
- Im Folgenden ist mit „Methodenaufruf“ die Nachricht an ein Objekt gemeint
- „Eigenwillige“ Klammer-Syntax:

```
[calculator add: 5 to: 10];

[object methodenname: param1 alias2: param2 … aliasN: paramN];
```

- Jeder bis auf der erste Parameter ist mit einem Alias-Namen versehen
- Die Methode selbst ist dann nicht durch ihren Namen sondern durch den Namen in Kombination mit allen Alias-Namen eindeutig definiert:

```
methodenname:  alias2:  … aliasN:
```

- Häufige Fehlerquelle: Vergessene Doppelpunkte!

Objective-C: Kontrollstrukturen

- Die meisten Kontrollstrukturen wie if-else, for, while oder switch werden in Objective-C analog zu den meisten OOP-Programmiersprachen definiert
- Beispiel: if-else

```
if ([object size] > 10 ) {
    /* Kommentar  */
    ...
}
else {
    // Inline-Kommentar
    ...
}
```

- Beispiel: foreach

```
for (Klasse* temp in Array) {
    [temp run];
}
```

- Die „Sternchen-Notation“ besagt (analog zu C++) dass es sich bei Klasse* temp um einen Zeiger auf ein Objekt der besagten Klasse handelt

Objective-C: Strings

- Auch die Notation für String-Literale ist gewöhnungsbedürftig:

```
NSString* string_1 = @"Hello World";
```

- NSString ist die Basisklasse für Strings in Objective-C
- Die Notation mit @ ist darauf zurückzuführen, dass Objective-C als Obermenge von C auch die daher bekannten Strings oder Char-Arrays kennt
- @ erzeugt also implizit Objekte des Typs NSString
- Der Umgang mit @Strings ist komplexer als es den Anschein zu haben mag:

```
NSString* string_1 = @"Hello World";
NSString* string_2 = @"Hello World";
```

- Hier werden zwei Objekte erzeugt, die offensichtlich den gleichen Inhalt haben

```
BOOL equals =  [string_1 isEqualToString:string_2] ; // → YES
```

- Allerdings zeigen beide Zeiger auch auf die gleiche Stelle im Speicher

```
BOOL equals =  (string_1 == string_2) ; // → YES
```

- Wichtig: == vergleicht die Adresse des Speicher auf die der Zeiger zeigt, nicht die Objekte!

Objective-C: Strings (Teil II)

- Log Ausgabe auf der Konsole erzeugen:

```
NSLog(@"Hello World");
```

- Formatierte Strings:

```
NSString* string = @""; NSString* temp;
for (int i = 0; i < 10; i++) {
    temp = [NSString stringWithFormat:@"%i", i];
    string = [string stringByAppendingString:temp];
    if (i < 9) string = [string stringByAppendingString:@", "];
}
// Ausgabe: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
```

- Die Methode stringWithFormat erzeugt einen String in dem durch % Platzhalter definiert werden können, die dann durch die weiteren Parameter in der definierten Reihenfolge ersetzt werden
- Die Bezeichnung des Platzhalters ergibt sich dabei aus dem Wert des Parameters. Im Beispiel „%i“ für „Integer“.
- stringWithFormat ist eine sogenannte „Vaarg“ Funktion, die eine beliebige Anzahl von Parameter akzeptiert (Man beachte die Syntax!)

Objective-C: Strings (Teil III)

- NSStrings sind unveränderbar!
- Mit jedem stringWithFormat und stringByAppendingString wird ein neues String-Objekt erzeugt → Speicherverschwendung
- Veränderbare Strings werden mit NSMutableString erstellt

```
NSMutableString* mutable = [NSMutableString string];
for (int i = 0; i < 10; i++) {
    [mutable appendFormat:@"%i", i];
    if (i < 9)  [mutable appendString:@", "];
}
```

- Einschub: Wie gibt man einen NSString-Objekt per NSLog aus?
- Der Platzhalter für Objekte (besser: Zeiger auf Objekte) ist %@

```
NSLog(@"%@", object);
```

- Obwohl die NSLog Funktion ein Parameter vom Typ NSString* erwartet, übergibt man den Zeiger nicht direkt:
  - Der übergebene Wert könnte Platzhalter enthalten, die allerdings nicht definiert würden → dennoch würde versucht werden, diese zu ersetzen → Sicherheitslücke

Objective-C: Kontrollstrukturen (II)

- Exception-Handling
  - Abfangen und Behandeln von möglichen Fehlersituationen

```
NSMutableString* mutable = @"Stil mutable?";
@try {
    mutable.string = @"Test";
}
@catch (NSException *exception) {
    NSLog(@"The mutable string is not mutable anymore.");
}
```

- NSMutableString erbt von NSString. Objekte können daher auch direkt per @“...“ zugewiesen werden
- Der Inhalt eines NSMutableString kann über das Attribut string direkt gesetzt werden (die Punktnotation wird später näher erläutert)
- So erzeugte Objekte vom Typ NSMutableString sind **nicht** mehr änderbar
- Eine erneute Zuweisung würde zu einer Ausnahme führen. Nicht durch @try-@catch abgefangen, führt dies zum unplanmäßigen Abbruch des Programms
- Hier wird der Fehler durch einen Log-Eintrag an die Konsole weitergegeben

Xcode - Source Control Management

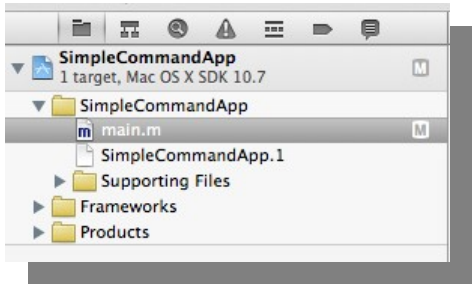
- Source-Code unterliegt naturgemäß ständigen Änderungen
- Bei großen Abhängigkeiten in Projekten kann die kleinste Änderung bereits zu großen Problemen führen
- Bevor Änderungen durchgeführt werden, neigt man daher zu Backups oder Sicherheitskopien des aktuellen Codes
- Source Control Management übernimmt die Verwaltung solcher Backups
- Ist dabei aber weitaus feingranularer und mächtiger als die üblichen „Copy&Paste“ Backups
- Alle Ressourcen eines Projektes werden in einem *Repository* versioniert
- Neue Versionen werden durch ein *Commit* in das Repository eingepflegt
- Jederzeit kann auf ältere Versionen einer Ressource zurückgegriffen werden
- SCM erleichtert vor allem das Arbeiten an einem Projekt durch mehrere Parteien, indem bspw. vor Konflikten gewarnt wird, wenn verschiedene Personen an der gleichen Ressource an gleicher Stelle unterschiedliche Änderungen anbringen

Xcode - Source Control Management: Git (I)

- Einfachklick auf „main.m“ im Source-Navigator und im Editfenster eine kleines NSString-Testprogramm schreiben

main.m

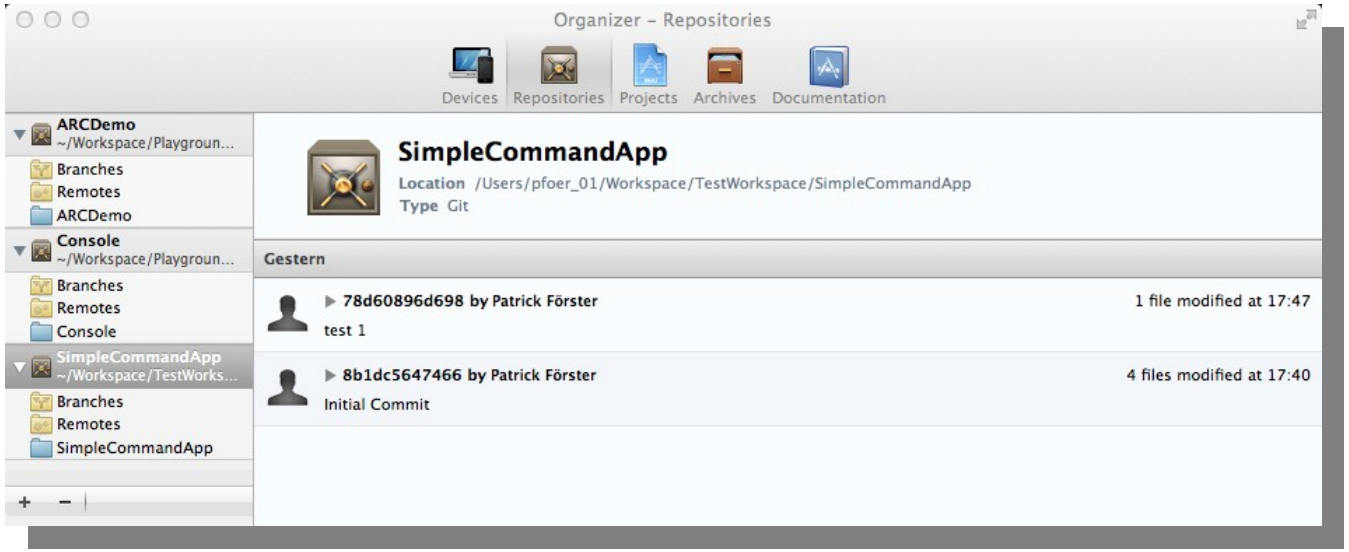
```
#import <Foundation/Foundation.h>

int main (int argc, const char * argv[]) {
    @autoreleasepool {
        // insert code here...
    }
    return 0;
}
```



- Datei speichern mit cmd + s
- Neben „main.m“ erscheint ein M für „Modified“
- Rechtsklick auf „main.m“→Source Control→Commit Selected Files... (alternative: File→Source Control→Commit)
- Fenster mit Vergleich zur letzten in Revision befindlichen Datei wird angezeigt
- Jedes „Commit“ muss mit einer Nachricht versehen werden
- Mit „Commit“ die Änderungen einpflegen

Xcode - Source Control Management: Git (II)

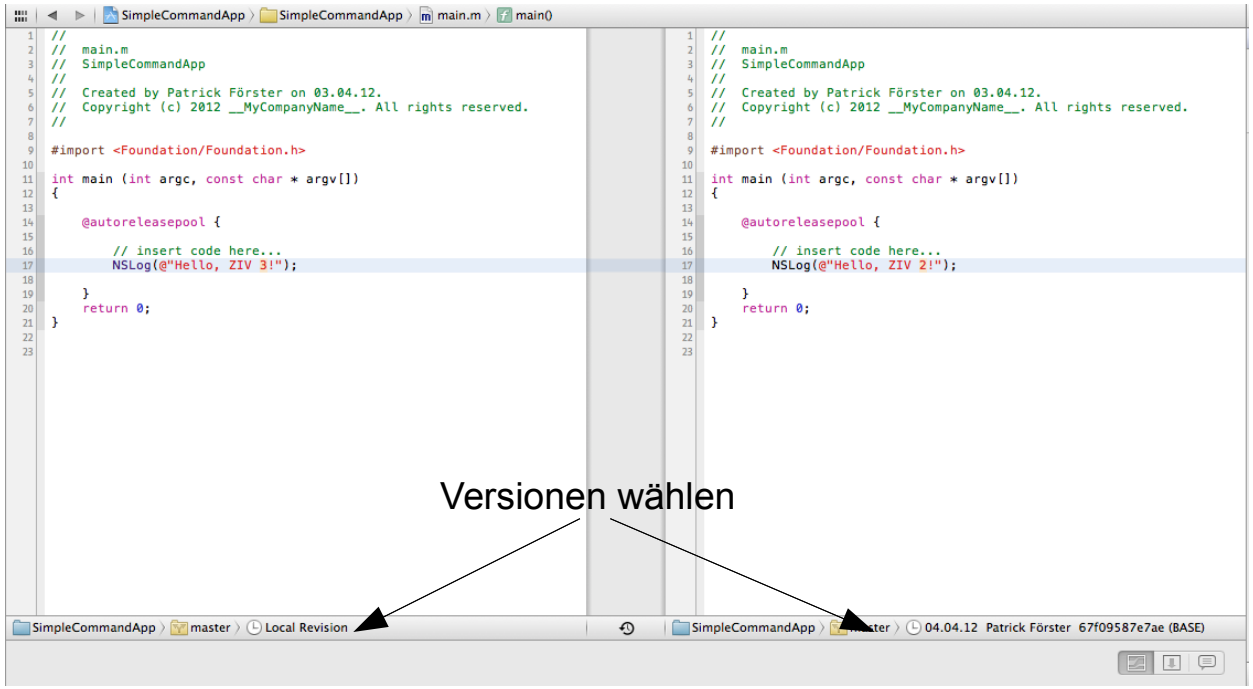
- Mit shift+cmd+2 (alternative: Window→Organizer) den Organizer öffnen und den Reiter „Repositories“ auswählen



- Das Respository für das Beispielprojekt sollte zwei Einträge haben:
  - Initial Commit: Automatisch erstellt beim Erzeugen des Projektes
  - *Commit-Nachricht*: Der soeben erzeugte Eintrag

Xcode - Source Control Management: Git (III)

- Um den Unterschied zwischen zwei Revisionen zu vergleichen muss im Hauptfenster von der Edit- in die Versionansicht gewechselt werden:



Xcode - Source Control Management: Snapshots

- Snapshots sind eine einfache Alternative zu SCM mit Git oder Subversion
- Ein Snapshot ist eine Kopie des aktuellen Standes
- Es kann jederzeit auf einen beliebigen Snapshot gewechselt werden
- Der aktuelle Stand geht allerdings verloren!
- Xcode erstellt automatisch Snapshots bei kritischen Operationen, wie etwa das Refactoring des Codes (z.B. Umbennen einer Klasse)
- Erzeugt und wiederhergestellt werden Snapshots über File→Create/Restore Snapshot