

Objective-C und Cocoa Programmierung

Objective C

- Erweiterung von C
 - Objektorientiert
 - Erweiterte Syntax
 - Erweiterung des GCC-Compilers
- Von NeXT entwickelt worden

Objekte und Klassen in ObjC

- Standardtyp id (sowas wie void *)
- Trennung in interface und implementation
- Vererbung von Oberklasse (aber keine mehrfache Vererbung)
- Kennzeichnen von Klassenmethoden und Instanzmethoden

Beispiel: Interface

```
@interface BlorgObject : NSObject {  
    int x;  
    id anObject;  
}
```

```
+ (void)eineKlassenMethode:(id)einObjekt;
```

```
- (id)init;
```

```
- (id)initMitInt:(int)einInt;
```

```
@end
```


Beispiel: Implementation

```
@implementation BlorgObject
```

```
+ (id)eineWeitereKlassenMethode {  
    return nil;
```

```
}
```

```
- (id)init {  
    return [super init];
```

```
}
```

```
@end
```


Nachrichten senden

- Smalltalkähnliche Syntax:
 - [object eineNachricht:argument]
- Nachrichten senden wird zur Laufzeit erledigt, nicht bei der Übersetzung
- Nachrichten dynamisch erstellen (selectors)

Beispiel: Nachrichten

```
/* Zur Übersetzungszeit */  
SEL einSelektor =  
    @selector(eineNachricht);  
/* Dynamisch */  
SEL einSelektor =  
    NSSelectorFromString(@"eineNachricht");  
/* Aufruf */  
[object performSelector:einSelektor];
```


Vererbung

- Vererben von anderen Klassen mit ":"
- Abstrakte Klassen die nicht instanziiert werden
- Keine mehrfache Vererbung, dafür ähnlich wie bei Java erben von Protocols (Interfaces in Java)
- Erweitern von Klassen durch Categories

Beispiel: Protocol

```
@protocol BlorgProtocol
```

```
- (void) machWas;
```

```
@end
```

```
@interface BO: NSObject <BlorgProtocol>
```

```
...
```

```
@end
```


Beispiel: Categories

```
@interface BlorgObject(geheim)
```

```
...
```

```
@end
```

```
@implementation BlorgObject(geheim)
```

```
...
```

```
@end
```


Memory Management

- Instanziierung von Objekten mit "alloc"
- Freigeben von Objekten mit "release"
- Reference Counting Garbage Collector (also von Hand)
 - Hochzählen des Referenzzählers mit "retain"

Beispiel: Memory Management

```
id blorg = [[BlorgObject alloc] init];  
[blorg release];
```

```
- (void)rememberObject:(id)anObject {  
    [anObject retain];  
}
```


Cocoa

- NeXT Api
- Foundation Classes
 - Strings, Arrays, Hashes, Kommunikation, usw...
- Application Kit
 - Grafische Widgets

Entwicklungsumgebung

- Project Builder
 - Editor, Make, Debugger, Information, CVS
- Interface Builder
 - Klickibunti Interfacedesign Tool
 - Produzierte serialisiert gespeicherte Klassen

Programmdesign

- Meistens werden Applikation nach dem MVC Modell gebaut
- Model: Daten, Algorithmen, "eigentliche Funktionalität"
- View: Darstellung dieser Daten (Oberfläche)
- Controller: Verknüpfung von View und Model

Vorgehensweise

- Nachdenken (aber wer macht das schon :)
- Implementieren der Funktionalität im Model
- Zusammenklicken der Oberfläche
- Verbinden von Oberfläche und Controller im IB
- Implementieren der Controller-Model Funktionalität

Klicken und Verbinden

- Zusammenlegen der Oberfläche
 - Helpers helfen beim Erfüllen der HIG
- Verbinden von Widgets an Outlets des Controllers
- Verbinden von Widgets an Methoden des Controllers (Actions)

Programmdesign

- Oft werden sogenannte Delegates benutzt
 - Ein Objekt schickt Nachrichten an seine Delegates, um externe Funktionalität zu benutzen
- Ähnlich: Notifications
 - Ein Objekt kann eine Reihe an Objekten benachrichtigen

Beispiel: Addierer

- Doofes Programm zum Addieren zweier Zahlen
- Kein Model, weil das Addieren ja wohl wirklich trivial ist
- Auftrennen in View (in IB bauen), Controller (in PB)

Beispiel: IB

- Zusammenziehen der Oberfläche
- Unterklasse der NSObject Klasse erstellen (Controller)
- Controller instanzieren und als Delegate eintragen
- Outlets und Actions verknüpfen