Objective-Cについて

2013.7.5

目次

- 概要
- メッセージ
- クラス
- インターフェースセクション
- インプリメントセクション
- カテゴリ
- プロトコル
- その他
- まとめ

概要

- C言語をベースにSmalltalk型のオブジェクト指向機能を持たせた上位互換言語
- C言語の機能は全て使用できる
- Objective-Cに特有の部分は、@で始まるコンパイラディレクティブで明示される
- Mac OS, iOSの標準開発言語

メッセージ①

- メッセージとは他の言語で言う関数、メソッドのことで、構文をメッセージ式という
- メッセージ呼び出しをメッセージ送信という

メッセージ②

- 書式

引数無

戻値有 : 戻値 = [インスタンス名 メソッド名];

戻値無 : [インスタンス名 メソッド名];

引数1つ

戻値 = [インスタンス名 メソッド名:引数値];

引数2つ

戻値 = [インスタンス名 メソッド名:引数値 引数名:引数値];

メッセージ③

例 引数無
```
val = [obj func];
```

例 引数1つ
```
val = [obj func:10];
```

例 引数2つ
```
val = [obj func:10 with:@”文字”];
```

例 入れ子
```
val = [obj1 func1:[obj2 func2]];
```

クラス

- Objective-Cのクラスは定義部(インターフェースセクション)と実装部(インプリメントセクション)に分かれている
- インターフェースセクションは.hファイルに記述する
- インプリメントセクションは.mファイルに記述する
- カテゴリによりクラス定義を複数のパートに分割できる
- クラスは単一継承、インターフェース(プロトコル)は多重継承
- いわゆるコンストラクタは存在しない。慣習的に+allocでインスタンスを生成し、-initで初期化を行う
- デストラクタとして-deallocがあり、終了時かならず呼び出される

インターフェースセクション①

- @interfaceで始まり、@endで終わる
- インスタンス変数とメソッドの宣言を行う

書式
@interface クラス名 : スーパークラス名
{
　インスタンス変数の宣言
}

メソッドの宣言
@end

インターフェースセクション②

例 MyObject.h

```
@interface MyObject : NSObject {
    // インスタンス変数
    int val;
    id obj;
}

// クラスメソッド
+ (void)classMethod:(id)arg;
// インスタンスメソッド
- (id)method:(NSObject*)arg1 with(int)arg2;
```

インプリメントセクション①

- @implementationで始まり、@endで終わる
- メソッドの定義を記述する

書式

```
@implementaion クラス名
    メソッドの定義
@end
```

インプリメントセクション②

例 MyObject.m

```
@implement MyObject
+ (void)classMethod(id)arg {
    // 何かの処理
}

- (id)method(NSObject*)arg1 with(int)arg2 {
    return obj;
}

// よくある初期化処理
- (id)init {
    self = [super init];
    if (self != nil) {
        val = 1;
        obj = [[NSObject alloc] init];
    }
    return self;
}

// 終了処理
- (void)dealloc {
    [obj release];
    [super dealloc];
}
@end
```

カテゴリ①

- クラスが持つメソッドを、名前の通りカテゴリごとに分類するための機能
- カテゴリで実装するメソッドは、通常のメソッドと何ら変わりはなく、インスタンス変数にも自由にアクセスできる
- カテゴリ側にインスタンス変数は宣言できない
- カテゴリの直接的な利点は、大きいクラスの実装を複数のファイルに分割できること
- 実装面でいうと、元のクラスとカテゴリのクラスは別々のオブジェクトで生成されるため、元のオブジェクトを変更せず、動的にクラスの拡張が可能となる

カテゴリ②

書式
@interface クラス名 (カテゴリ名)
@implement クラス名 (カテゴリ名)

例(インターフェース部のみ)

```
// Documentクラスの宣言
@interface Document : NSObject
{
    // インスタンス変数
    NSString* title;
}

- (NSString*)title;
- (void)setTitle:(NSString*)title;
@end

// DocumentクラスのPersistenceカテゴリ
- (void)writeToFile:(NSString*)path;
- (void)readFromFile:(NSString*)path;
@end
```

プロトコル①

- プロトコルはクラスのメソッドインターフェースを規定する機構であり、Javaでいうインターフェースのようなもの
- プロトコルに準拠するクラスは定義されたメソッドを全て実装しなければならない(ただし例外あり)
- プロトコルは多重継承を許す

プロトコル②

- @protocolで始まり、@endで終わる
- 実装必須な定義は@requireか何も指定しないで記述する
- 実装しなくて良いものは@optionalを指定する

書式
@protocol プロトコル名
@require
　メソッド名1
@optional
　メソッド名
@end

(実装側)
@interface クラス名 <プロトコル名>
メソッド名1
メソッド名2
@end

その他

- NSObject:全てのクラスの基底クラス
- id:汎用のデータ型　あらゆるクラスのインスタンスを格納できる。変数名の前に「*」は不要
- self:インスタンスそのものを表す
- super:スーパークラスを表す
- nil:nullオブジェクトを表し、オブジェクトが未初期化であることや、クリアされた状態であることを表わすのに使用される

まとめ

- Objective-CはC言語＋オブジェクトシステムなので、C言語を知っていれば学習難易度は低い
- メッセージ、カテゴリ、プロトコルなどは独自の機能だが、Javaなどのオブジェクト指向言語を知っていれば概念は理解可能
- Mac OS, iOSの開発を行う場合は必須だが、他での利用は今のところ皆無

以上