資料編

目次

図１　教材の趣旨

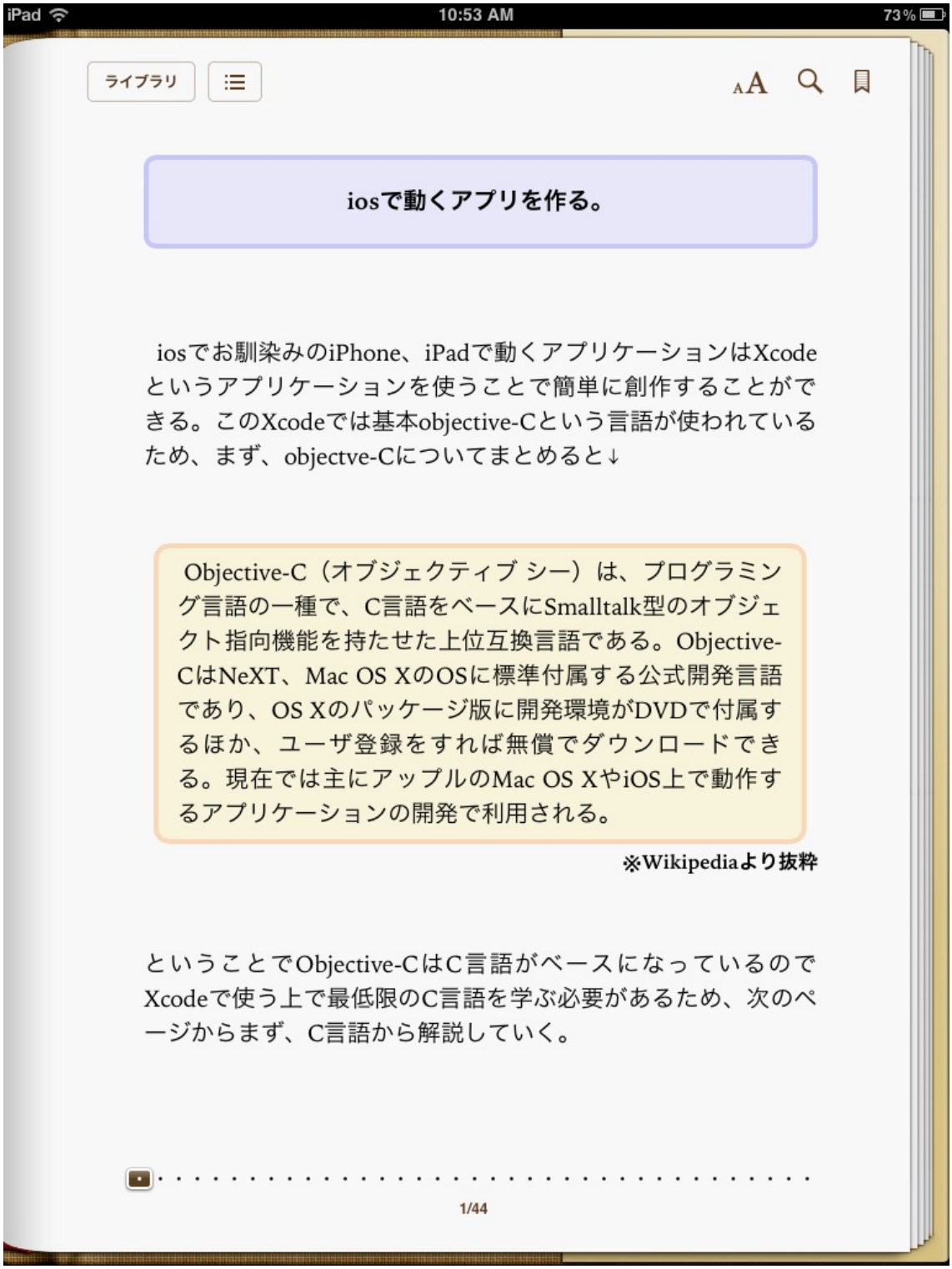

iPad
10:53 AM
73%
ライブラリ
iosで動くアプリを作る。
iosでお馴染みのiPhone、iPadで動くアプリケーションはXcodeというアプリケーションを使うことで簡単に創作することができる。このXcodeでは基本objective-Cという言語が使われているため、まず、objectve-Cについてまとめると↓
Objective-C（オブジェクティブ シー）は、プログラミング言語の一種で、C言語をベースにSmalltalk型のオブジェクト指向機能を持たせた上位互換言語である。Objective-CはNeXT、Mac OS XのOSに標準付属する公式開発言語であり、OS Xのパッケージ版に開発環境がDVDで付属するほか、ユーザ登録をすれば無償でダウンロードできる。現在では主にアップルのMac OS XやiOS上で動作するアプリケーションの開発で利用される。
※Wikipediaより抜粋
ということでObjective-CはC言語がベースになっているのでXcodeで使う上で最低限のC言語を学ぶ必要があるため、次のページからまず、C言語から解説していく。
1/44

図２　教材の目次


iPad
5:33 PM
充電していません
ライブラリ
再開
目次
ブックマーク
メモ
iosで動くアプリを作る。 1
C言語入門 2
Xcode入門1 ～ファイルの作成～ 9
Xcode入門２ ～Story board～ 12
Xcode入門３ ～BMI計算機～ 20
Xcodeプログラム解説 34
Xcode応用編 ～面積計算機～ 39
Xcode応用編 ～面積計算機２～ 49
Xcode応用編 ～数当てゲーム～ 52

図３　１．ソースプログラムの基本記号

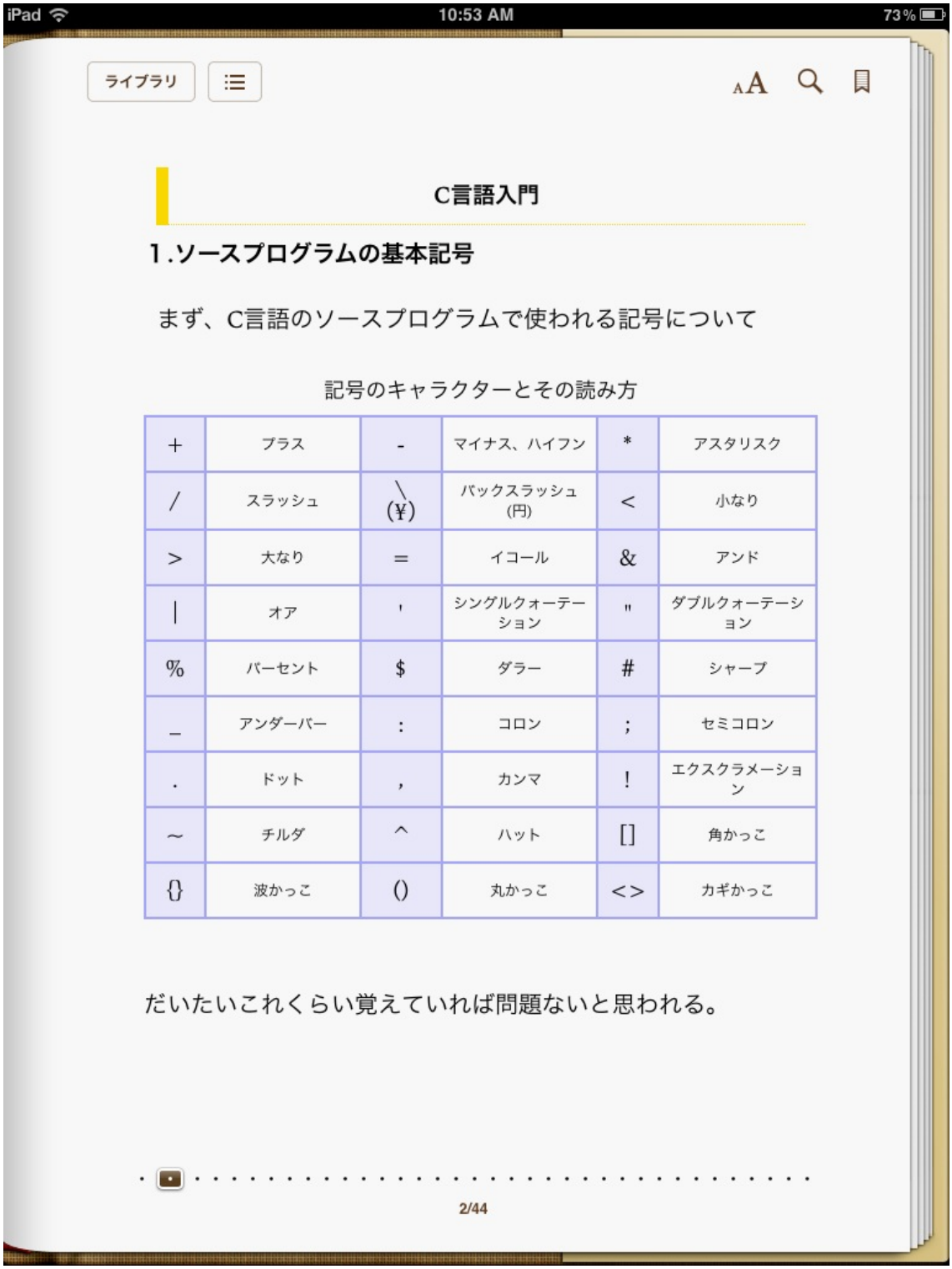


図４　２．基本プログラム作成

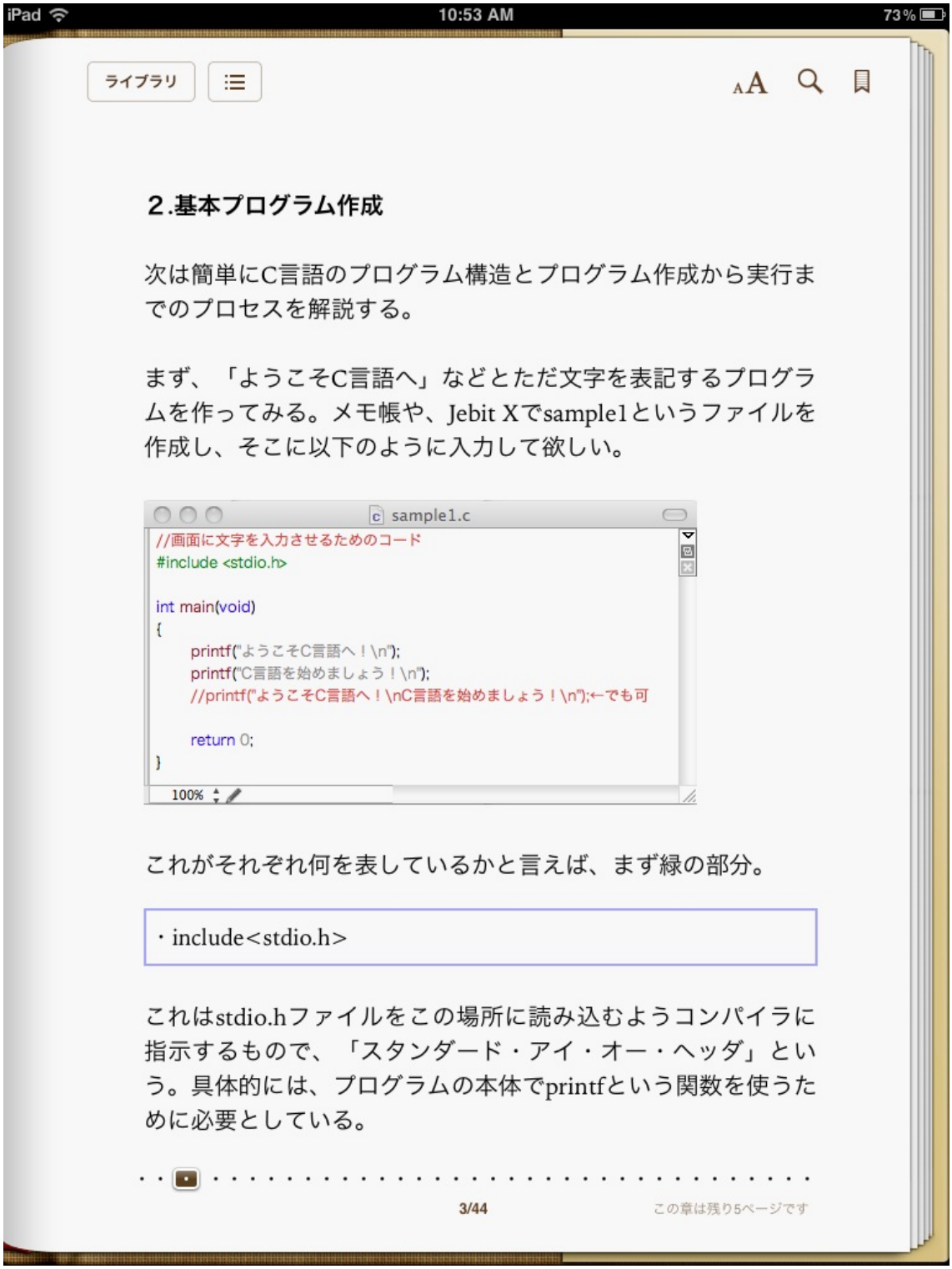


図５　３．動作確認

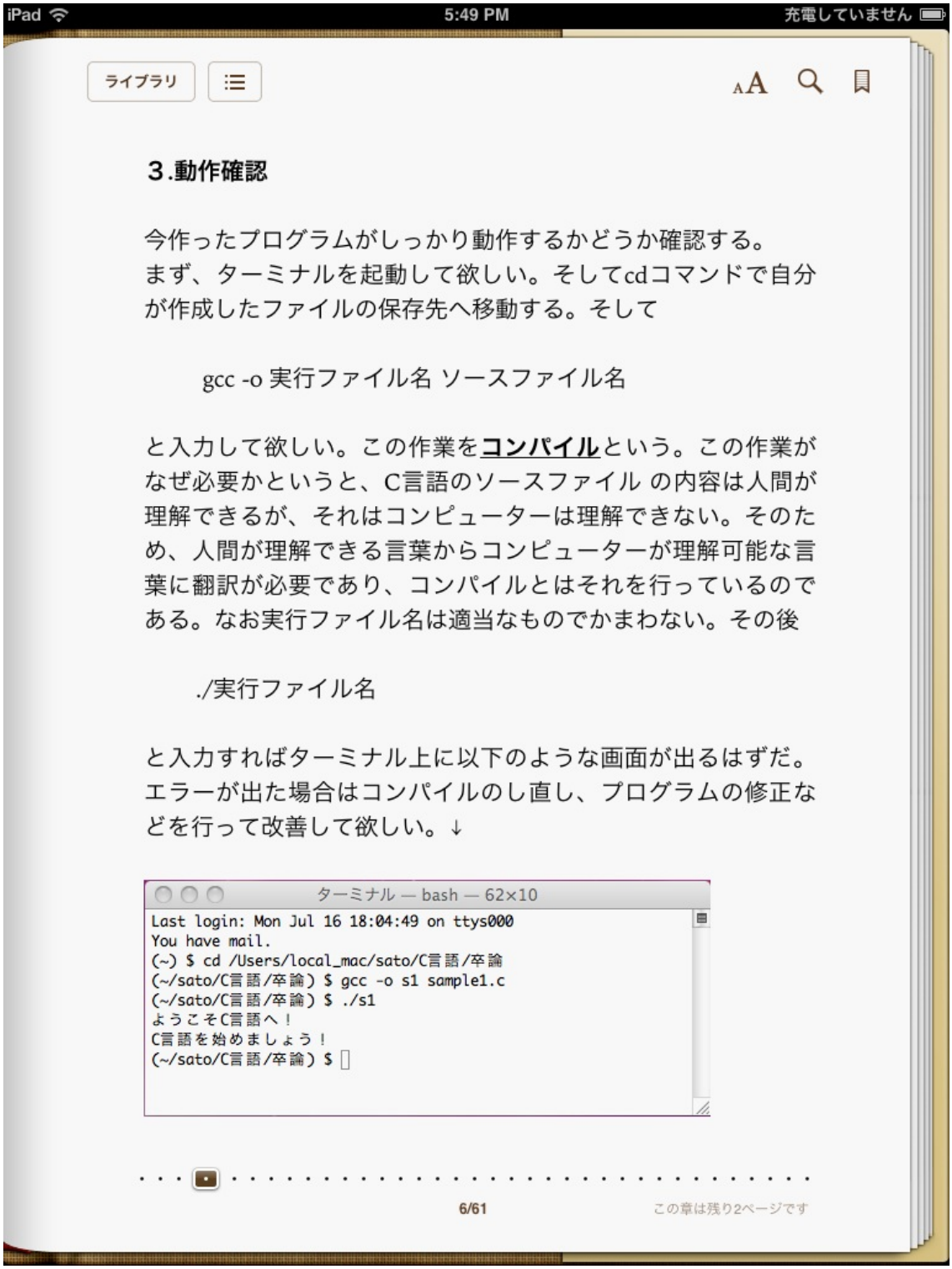

iPad
5:49 PM
充電していません
ライブラリ
3.動作確認
今作ったプログラムがしっかり動作するかどうか確認する。
まず、ターミナルを起動して欲しい。そしてcdコマンドで自分が作成したファイルの保存先へ移動する。そして
gcc -o 実行ファイル名 ソースファイル名
と入力して欲しい。この作業をコンパイルという。この作業がなぜ必要かというと、C言語のソースファイル の内容は人間が理解できるが、それはコンピューターは理解できない。そのため、人間が理解できる言葉からコンピューターが理解可能な言葉に翻訳が必要であり、コンパイルとはそれを行っているのである。なお実行ファイル名は適当なものでかまわない。その後
./実行ファイル名
と入力すればターミナル上に以下のような画面が出るはずだ。エラーが出た場合はコンパイルのし直し、プログラムの修正などを行って改善して欲しい。↓
ターミナル — bash — 62×10
Last login: Mon Jul 16 18:04:49 on ttys000
You have mail.
(~) $ cd /Users/local_mac/sato/C言語/卒論
(~/sato/C言語/卒論) $ gcc -o s1 sample1.c
(~/sato/C言語/卒論) $ ./s1
ようこそC言語へ！
C言語を始めましょう！
(~/sato/C言語/卒論) $
6/61
この章は残り2ページです

図６　４．必要用語解説

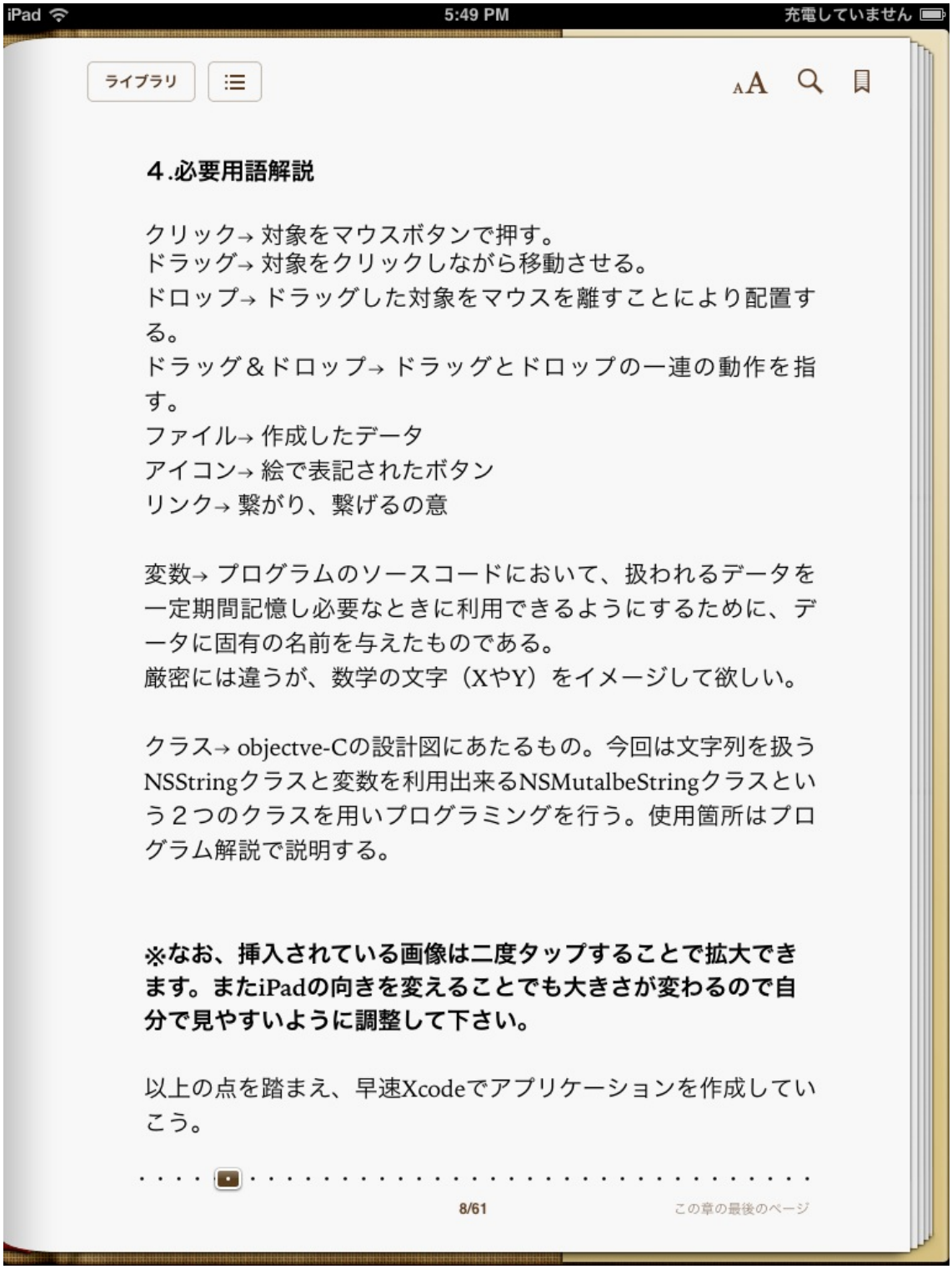

iPad
5:49 PM
充電していません
ライブラリ
4.必要用語解説
クリック→ 対象をマウスボタンで押す。
ドラッグ→ 対象をクリックしながら移動させる。
ドロップ→ ドラッグした対象をマウスを離すことにより配置する。
ドラッグ＆ドロップ→ ドラッグとドロップの一連の動作を指す。
ファイル→ 作成したデータ
アイコン→ 絵で表記されたボタン
リンク→ 繋がり、繋げるの意
変数→ プログラムのソースコードにおいて、扱われるデータを一定期間記憶し必要なときに利用できるようにするために、データに固有の名前を与えたものである。
厳密には違うが、数学の文字（XやY）をイメージして欲しい。
クラス→ objectve-Cの設計図にあたるもの。今回は文字列を扱うNSStringクラスと変数を利用出来るNSMutalbeStringクラスという２つのクラスを用いプログラミングを行う。使用箇所はプログラム解説で説明する。
※なお、挿入されている画像は二度タップすることで拡大できます。またiPadの向きを変えることでも大きさが変わるので自分で見やすいように調整して下さい。
以上の点を踏まえ、早速Xcodeでアプリケーションを作成していこう。
8/61
この章の最後のページ

図７　Xcode Ver3.2 と XcodeVer4 の表記の違い

上が Ver3 シリーズ。

下が Ver４シリーズ。

表記が日本語から英語に変わっている。

図８　２つの画面がボタンを押すことによってループするアプリケーション

画面 A　　画面 B

画面 A と B がボタンを押すことでループする。

図９　Story board の解説

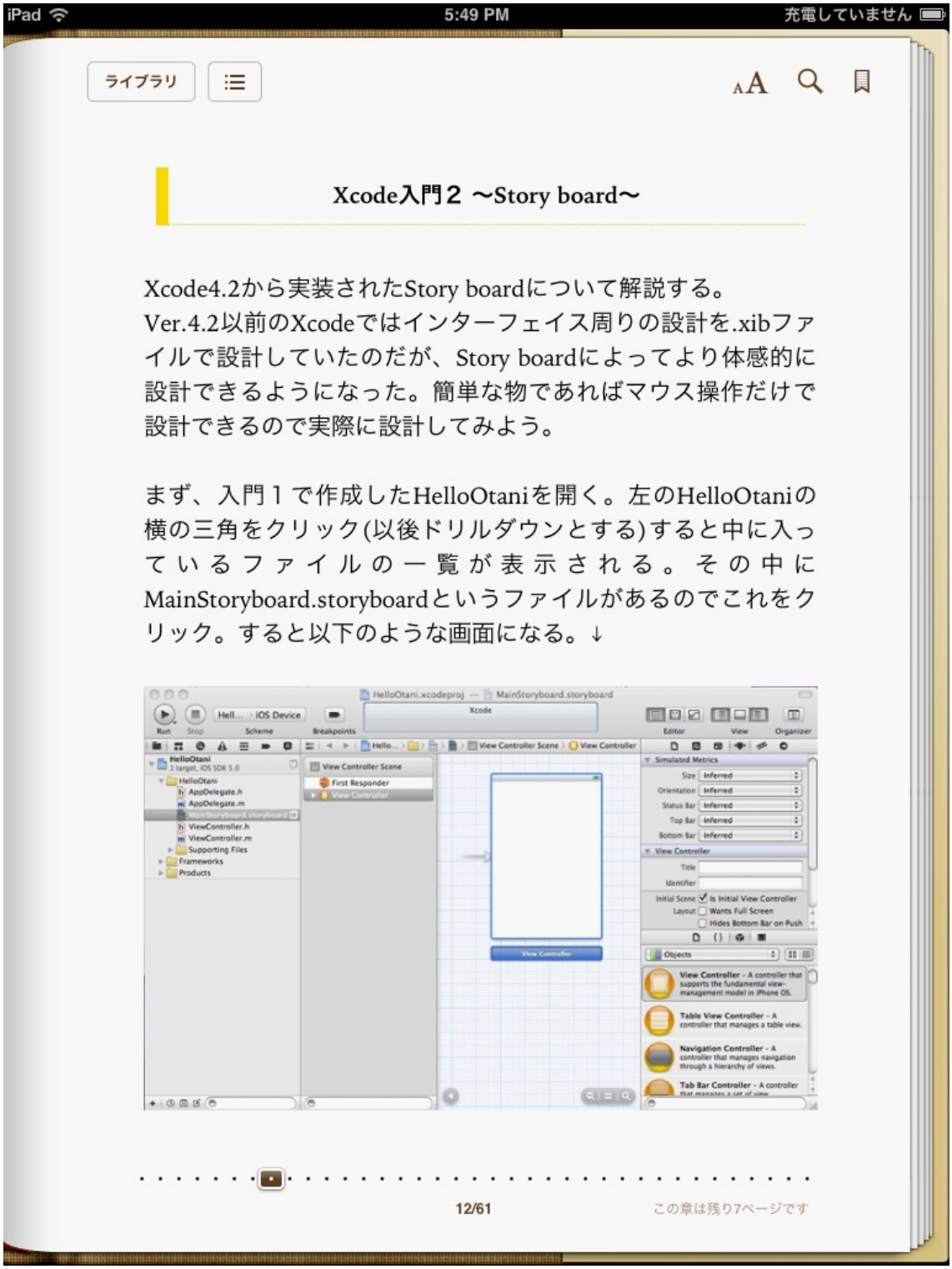


図１０　ワークスペースの名称設定

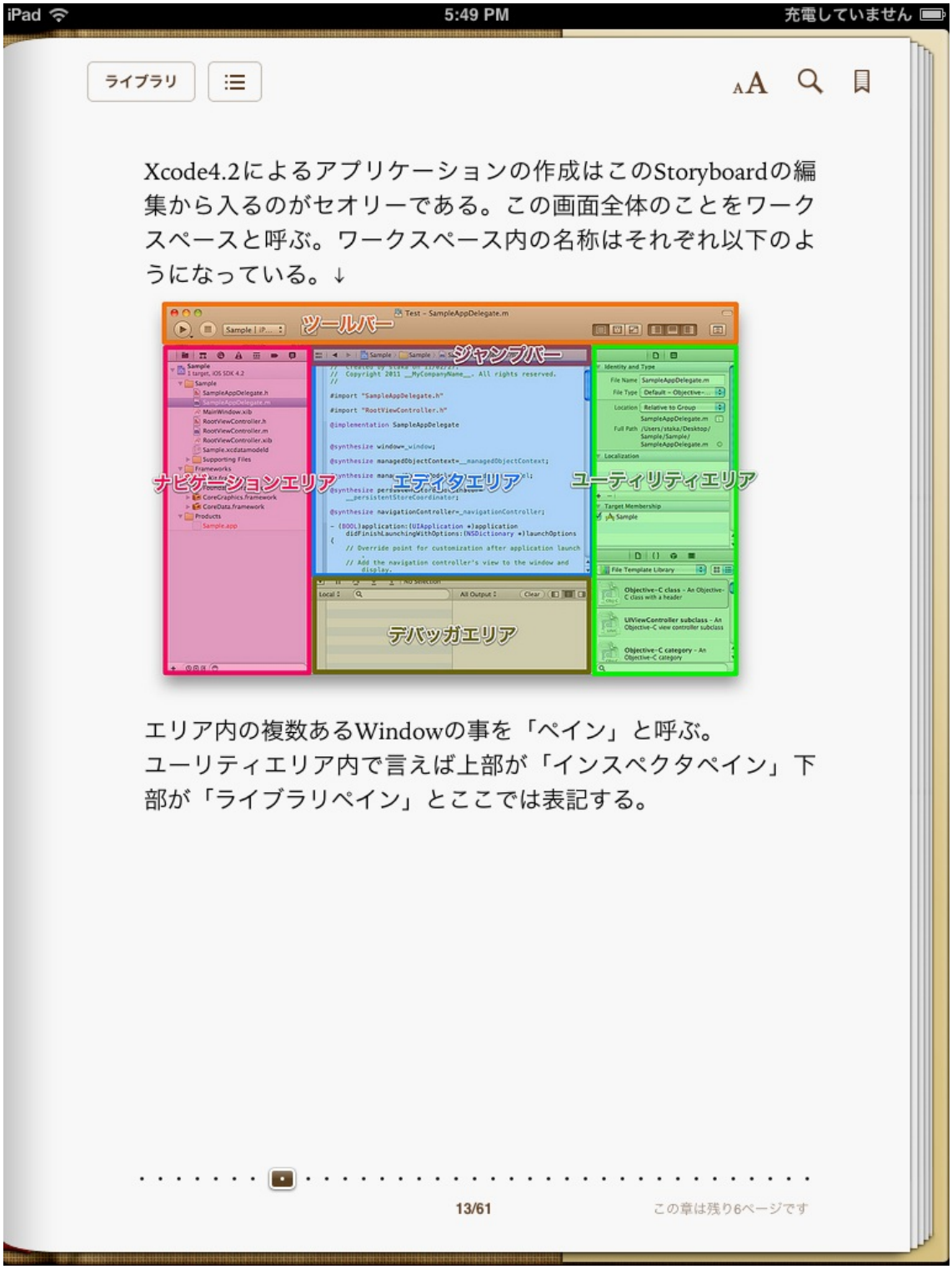


図１１　BMI 計算機


Carrier
7:10 PM
BMI値計算機
身長
cm
体重
kg
計算
リセット
四角形面積
三角形面積
数当て
Carrier
7:11 PM
BMI値計算機
身長
170
cm
体重
60
kg
計算
リセット
BMI値は 20.7612です。
標準体重は 63.58kgなので
「理想の体重」です。
四角形面積
三角形面積
数当て

図１２　Story board におけるプログラムソースを組み込む方法

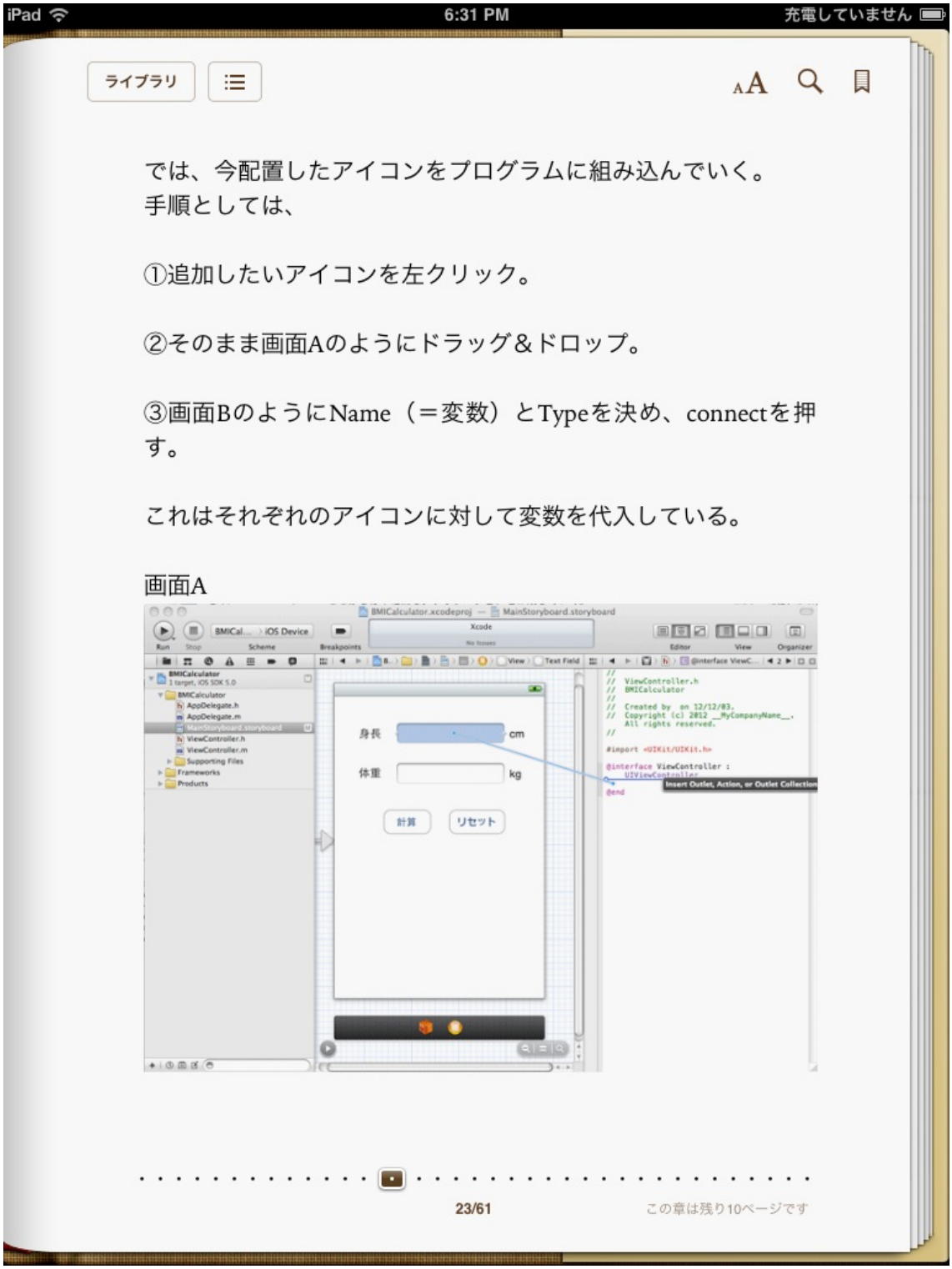


図１３　プログラムソース画面

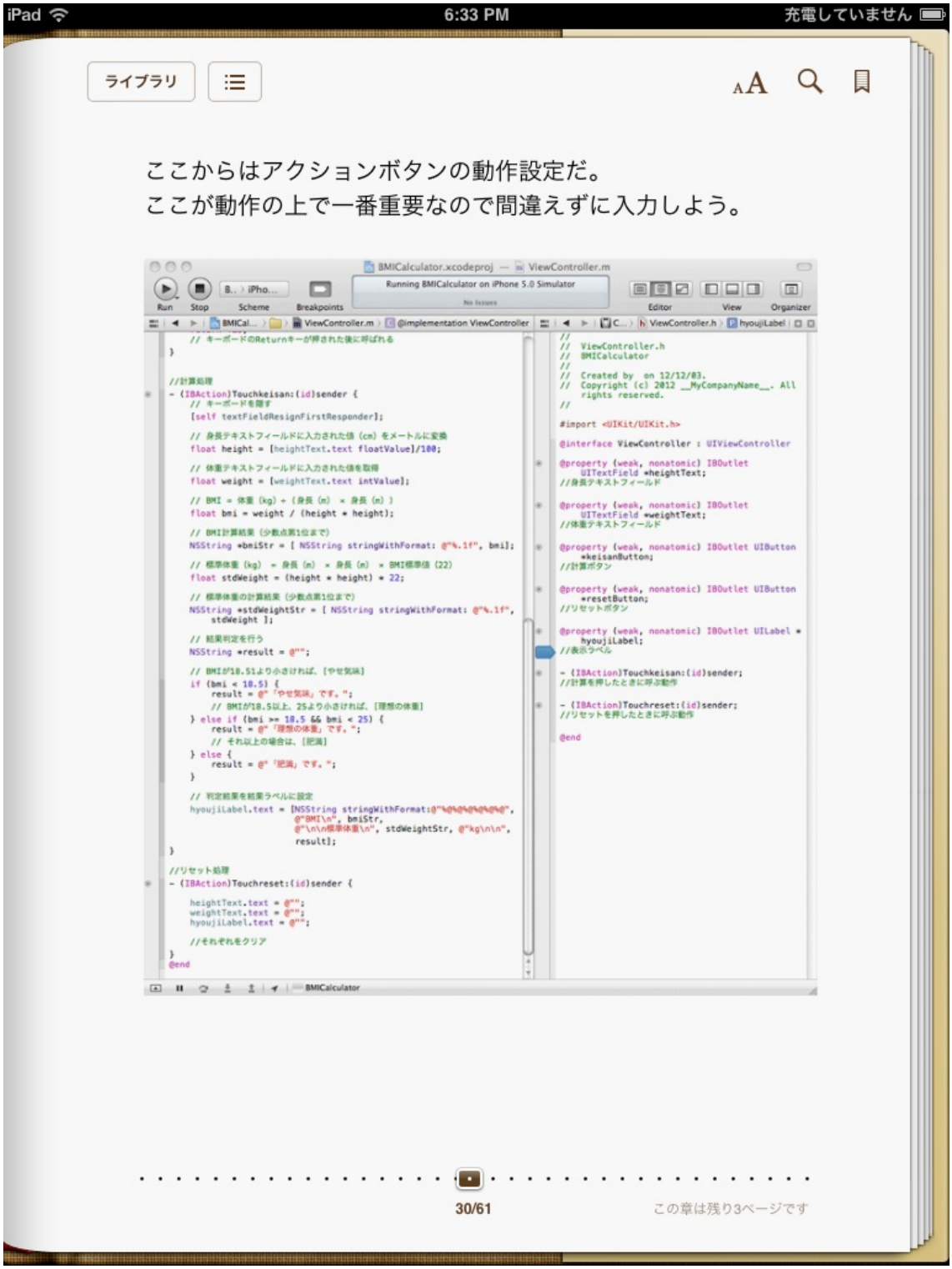


図１４　プログラム解説

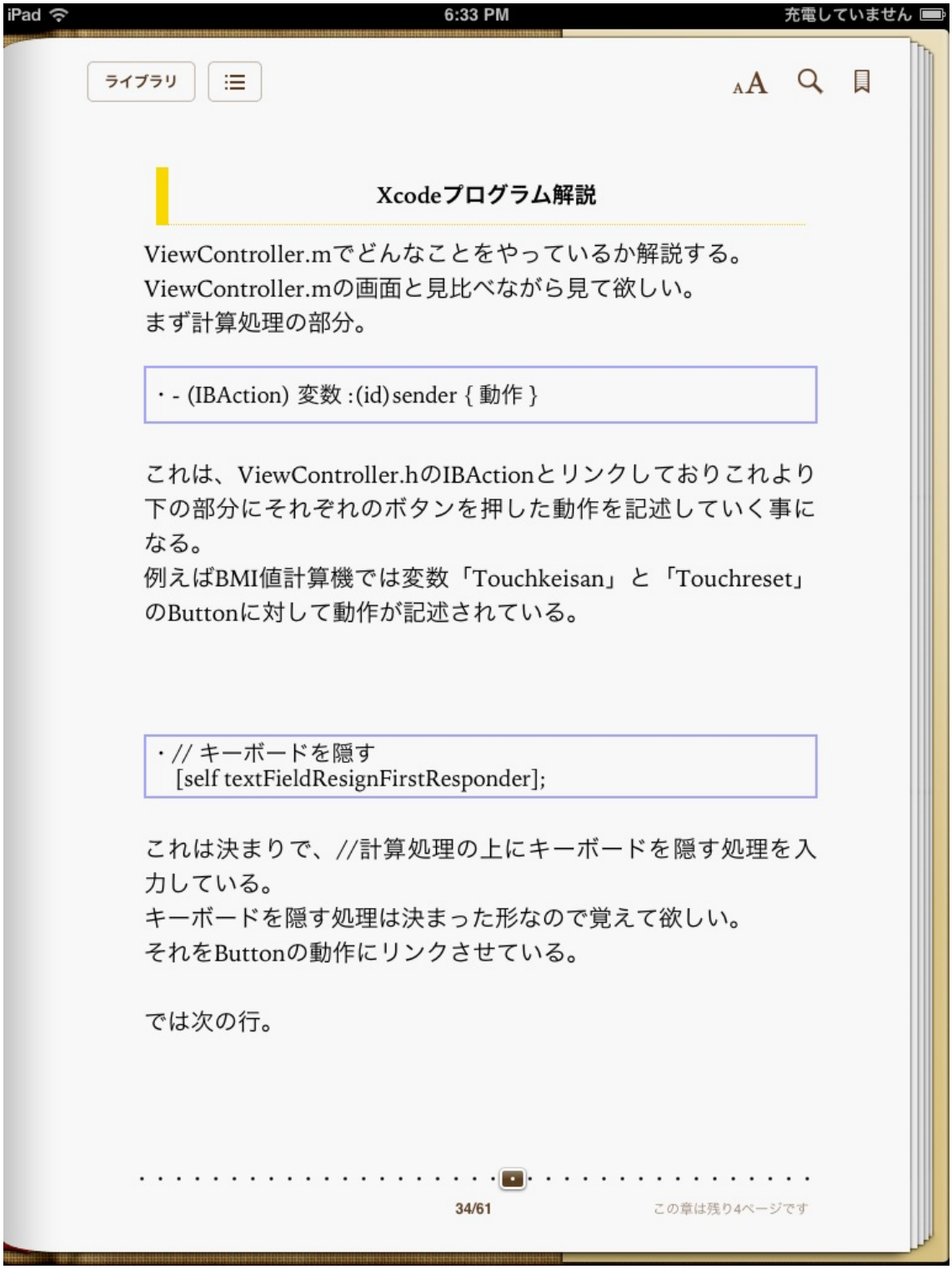

iPad
6:33 PM
充電していません
ライブラリ
Xcodeプログラム解説
ViewController.mでどんなことをやっているか解説する。
ViewController.mの画面と見比べながら見て欲しい。
まず計算処理の部分。
・- (IBAction) 変数 :(id)sender { 動作 }
これは、ViewController.hのIBActionとリンクしておりこれより下の部分にそれぞれのボタンを押した動作を記述していく事になる。
例えばBMI値計算機では変数「Touchkeisan」と「Touchreset」のButtonに対して動作が記述されている。
・// キーボードを隠す
[self textFieldResignFirstResponder];
これは決まりで、//計算処理の上にキーボードを隠す処理を入力している。
キーボードを隠す処理は決まった形なので覚えて欲しい。
それをButtonの動作にリンクさせている。
では次の行。
34/61
この章は残り4ページです

図１５　応用に使う予定だった電卓のプログラムソース

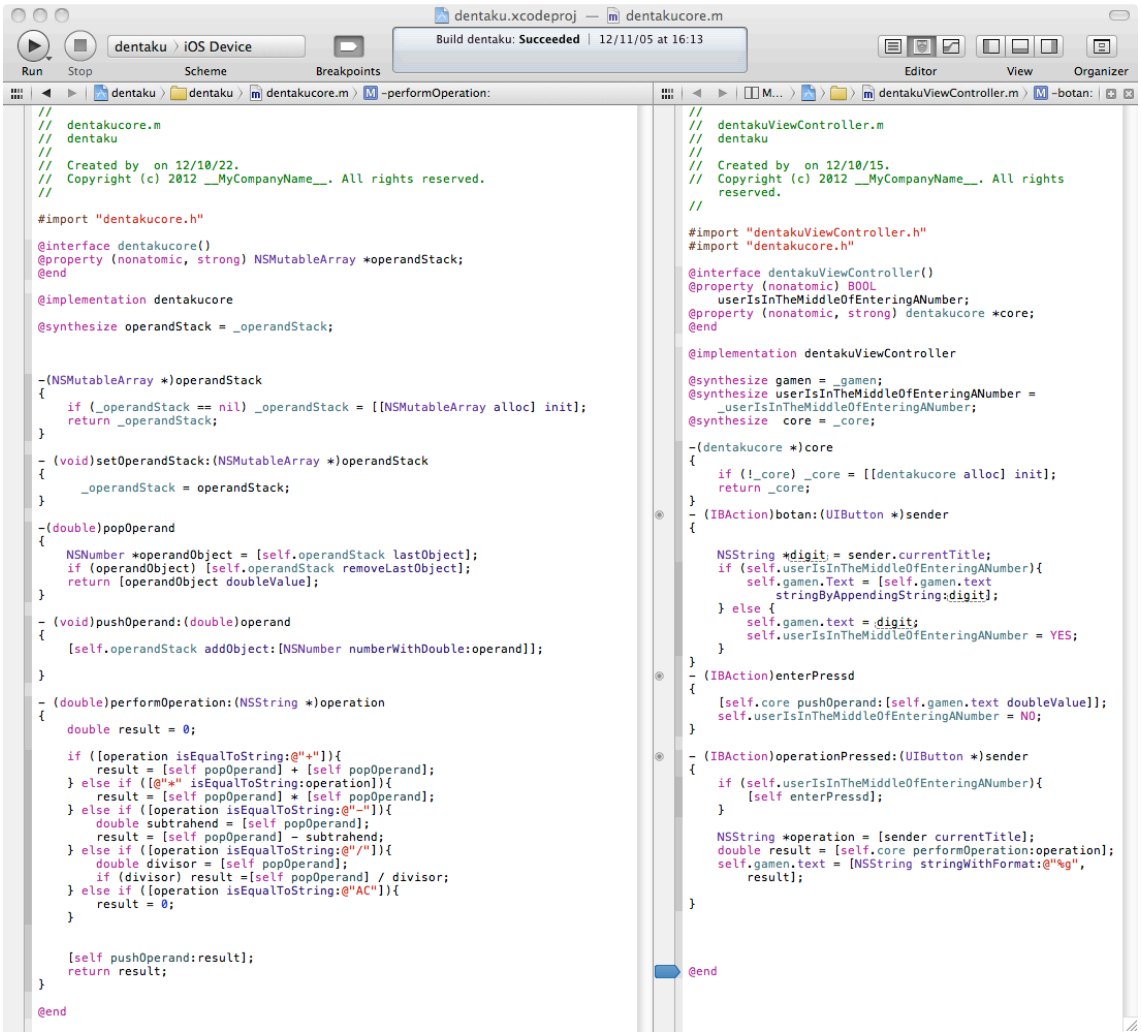


図１６　電卓のシミュレータ画面

図１７　改善後の BMI 計算機プログラムソース画面

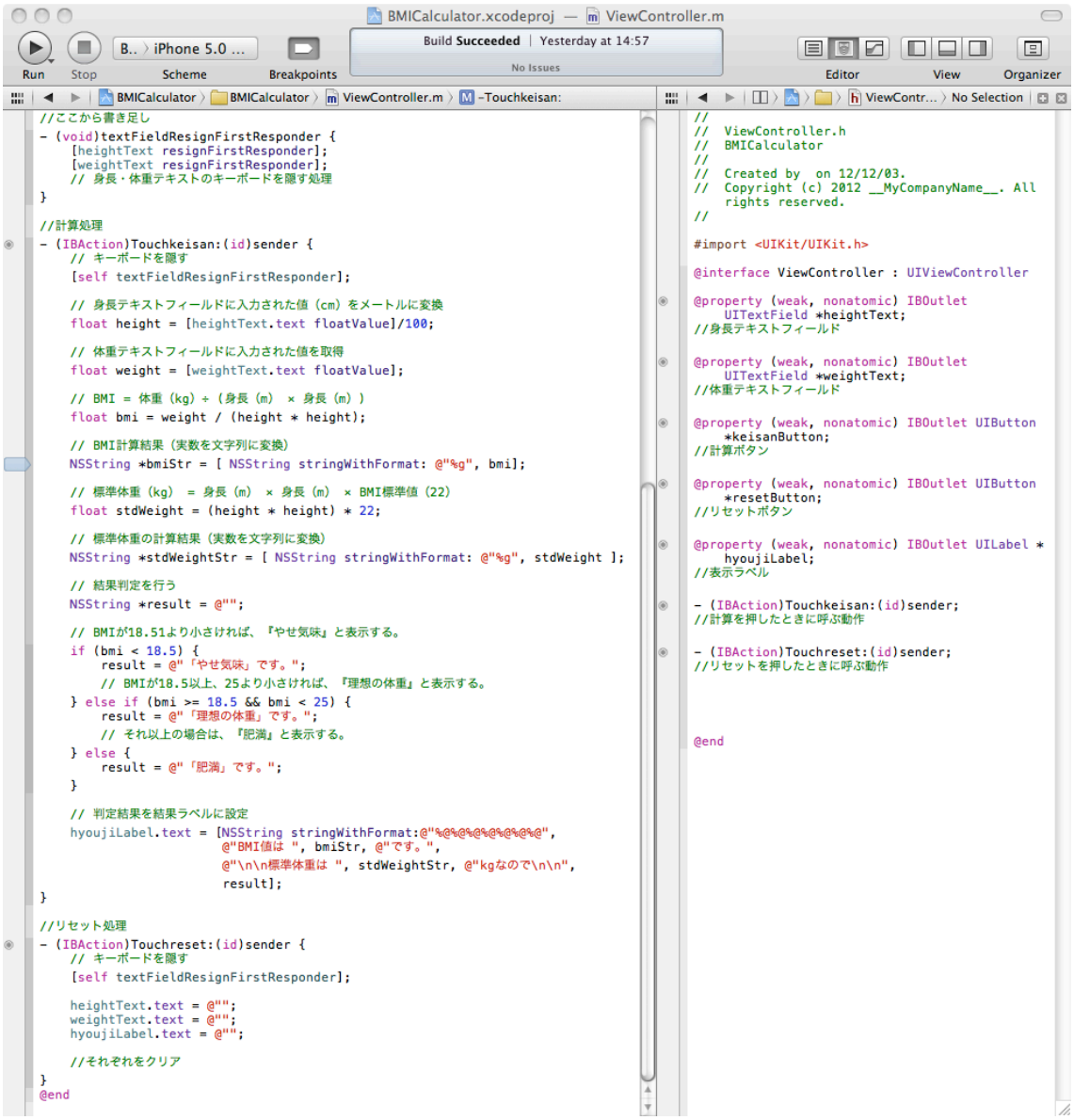


図１８　応用問題の作成

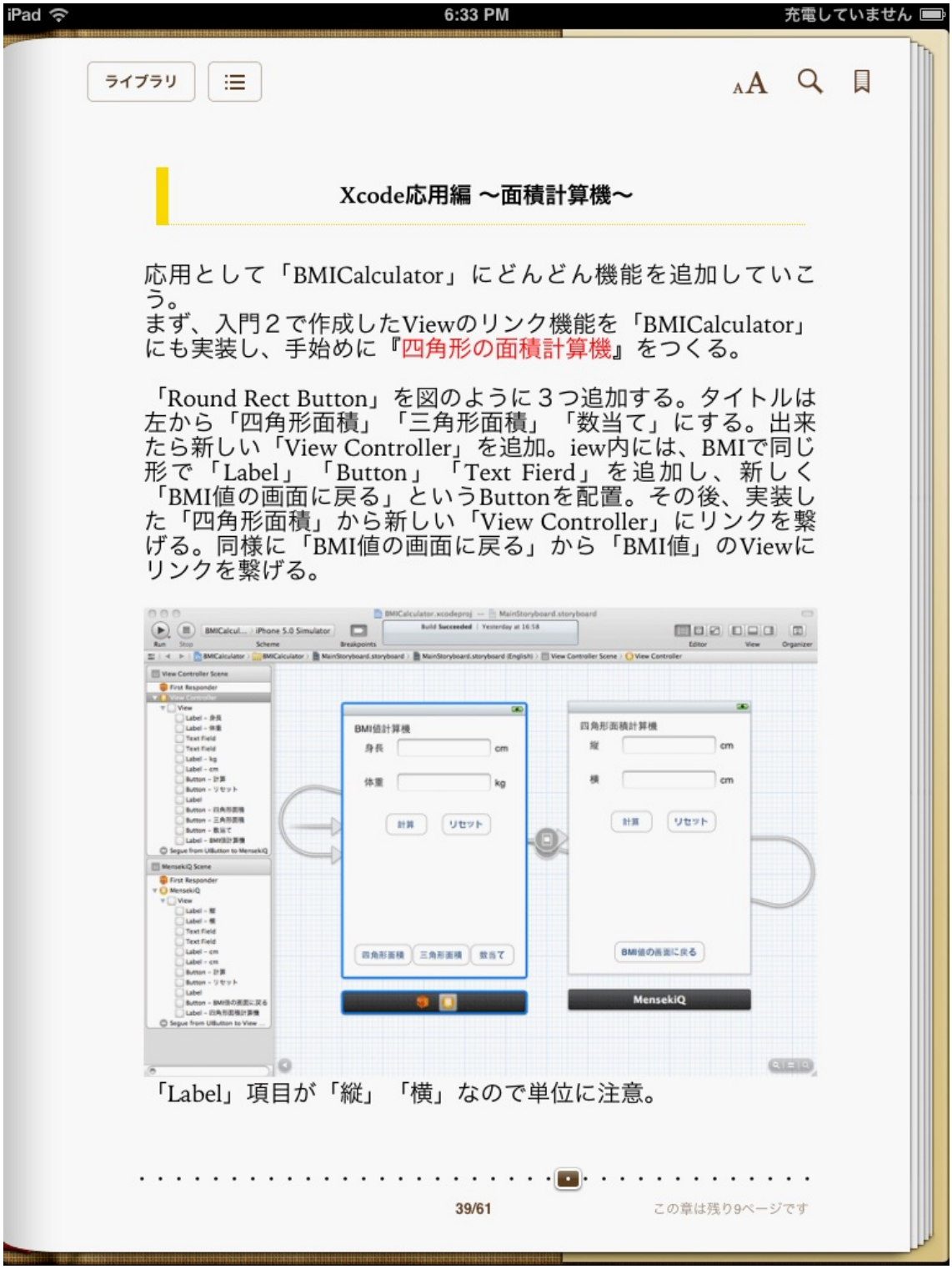

iPad
6:33 PM
充電していません
ライブラリ
Xcode応用編 ～面積計算機～
応用として「BMICalculator」にどんどん機能を追加していこう。
まず、入門２で作成したViewのリンク機能を「BMICalculator」にも実装し、手始めに『四角形の面積計算機』をつくる。
「Round Rect Button」を図のように３つ追加する。タイトルは左から「四角形面積」「三角形面積」「数当て」にする。出来たら新しい「View Controller」を追加。iew内には、BMIで同じ形で「Label」「Button」「Text Fierd」を追加し、新しく「BMI値の画面に戻る」というButtonを配置。その後、実装した「四角形面積」から新しい「View Controller」にリンクを繋げる。同様に「BMI値の画面に戻る」から「BMI値」のViewにリンクを繋げる。
BMI値計算機
身長
cm
体重
kg
計算
リセット
四角形面積
三角形面積
数当て
四角形面積計算機
縦
cm
横
cm
計算
リセット
BMI値の画面に戻る
MensekiQ
「Label」項目が「縦」「横」なので単位に注意。
39/61
この章は残り9ページです

図１９　応用問題作成後の画面推移見込み

画面 A　　画面 B

画面 C　　画面 D

画面 A の３つのボタンを押すことで対応した画面 B、画面 C、画面 D の３種類の画面に変わる。

図２０　四角形面積計算機

図２１　四角形面積計算機プログラムソース画面

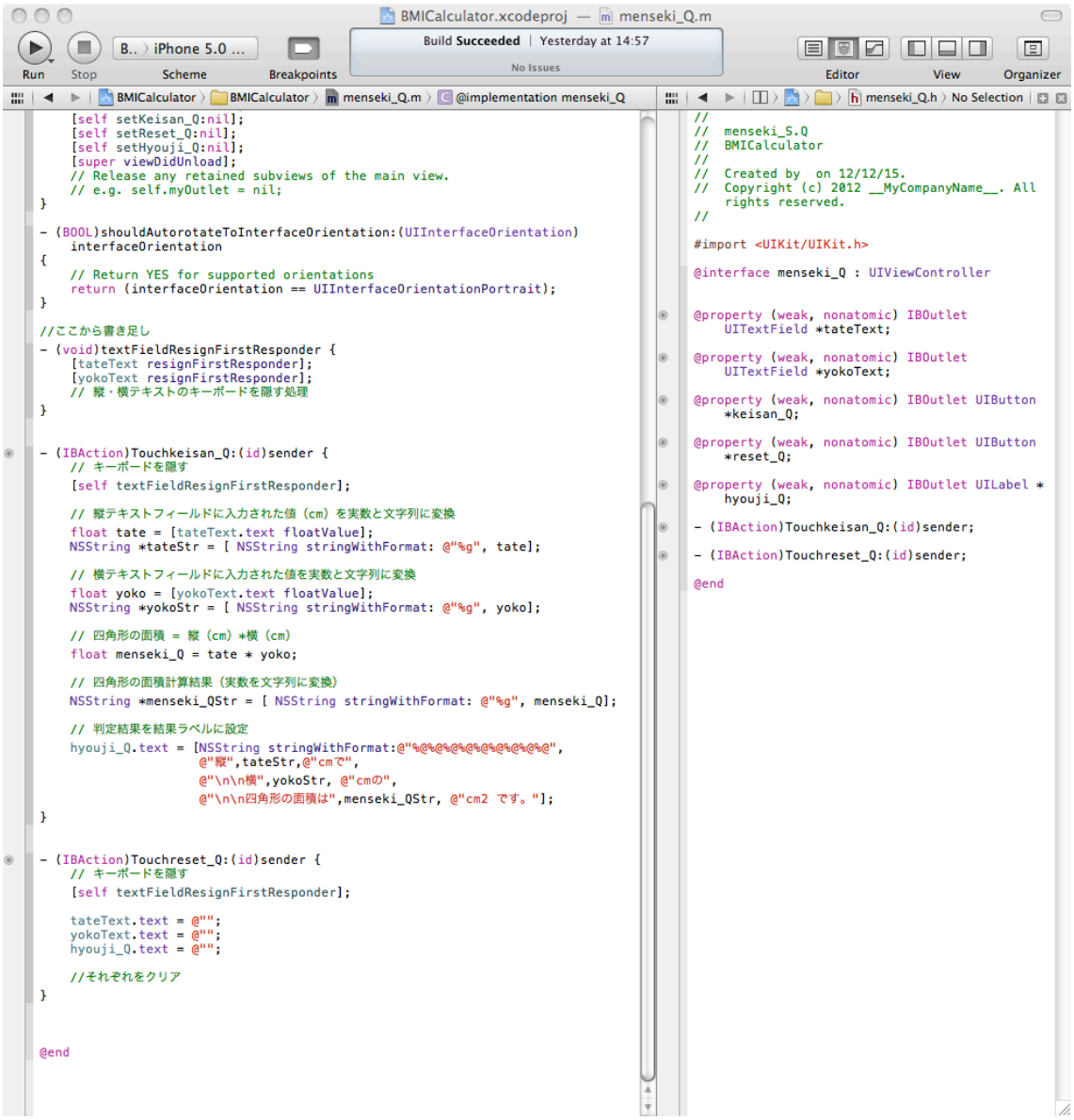


図２２　三角形面積計算機

図２３　三角形面積計算機プログラムソース画面

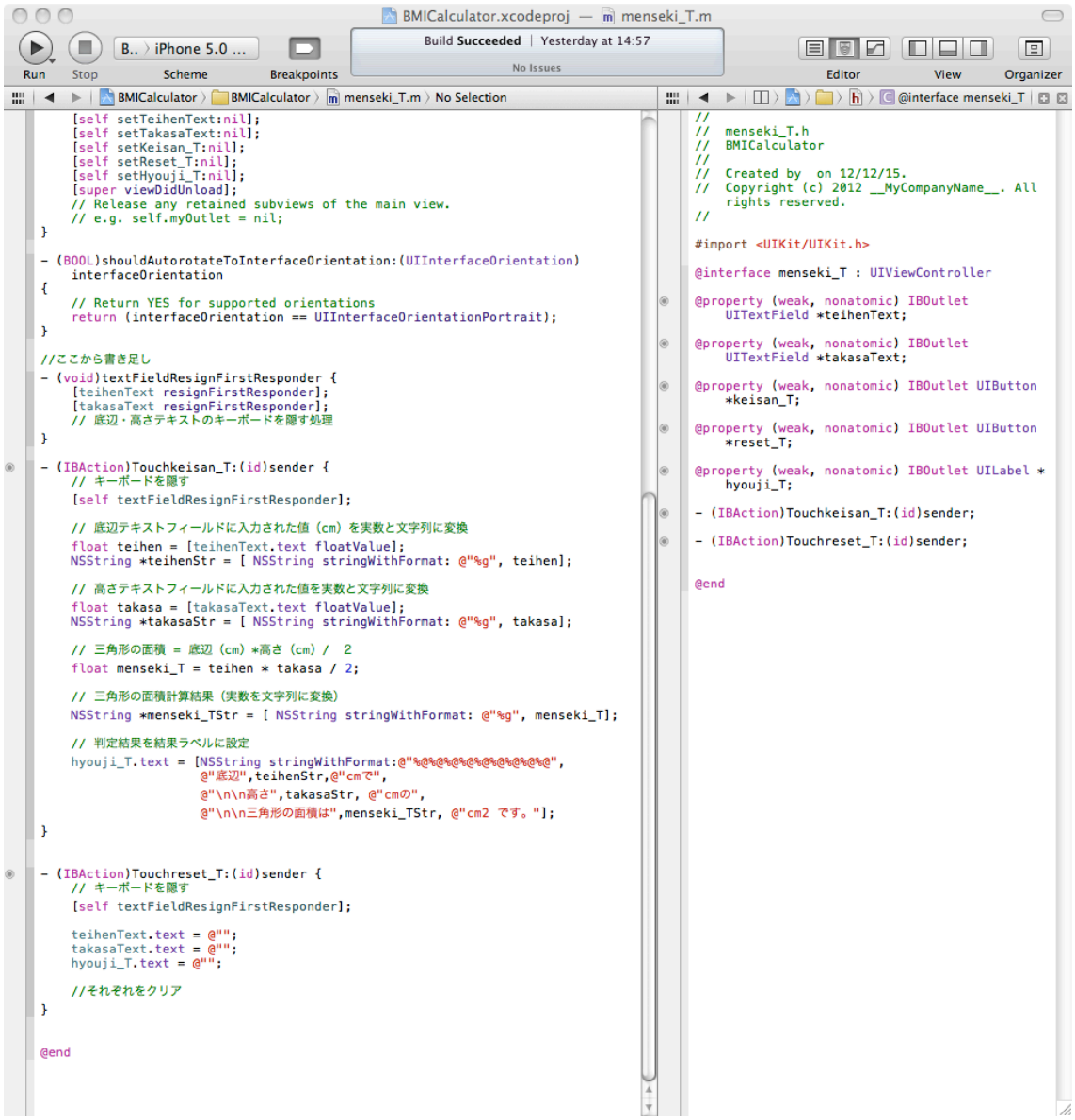
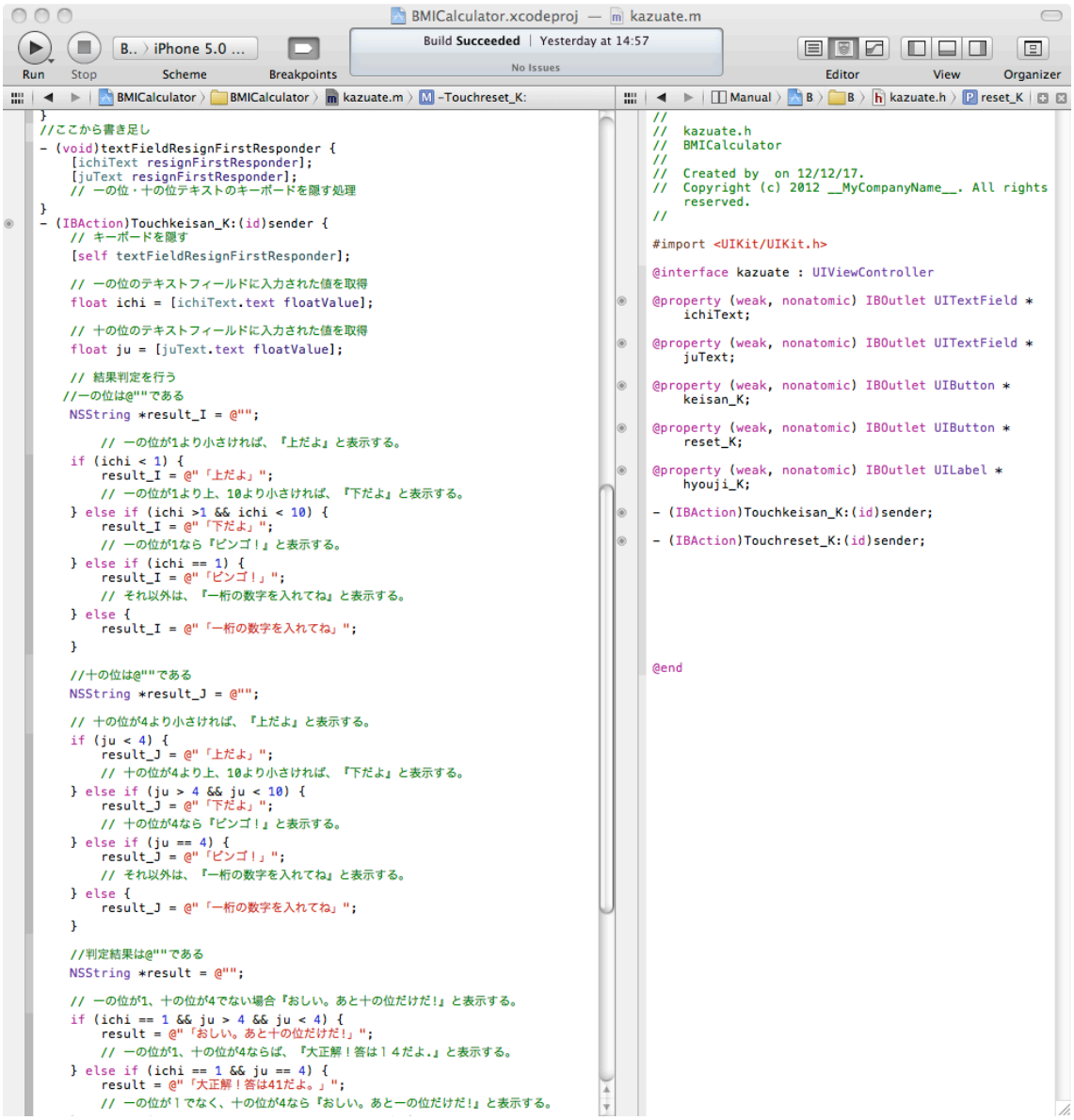
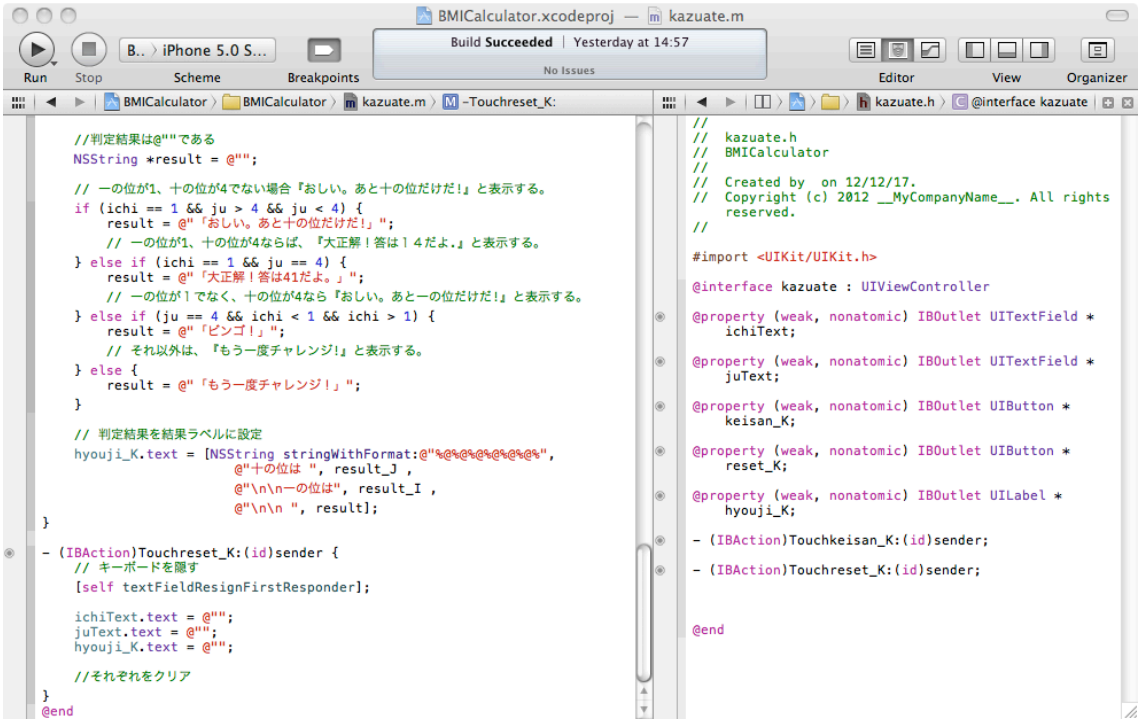


図２４　数当てゲーム


Carrier
6:50 PM
数当てゲーム(二桁の数字を当てよう)
十の位
一の位
1~9を入力
1~9を入力
これだ！
リセット
BMI値の画面に戻る
Carrier
6:51 PM
数当てゲーム(二桁の数字を当てよう)
十の位
一の位
5
0
これだ！
リセット
十の位は 「下だよ」
一の位は「上だよ」
「もう一度チャレンジ！」
BMI値の画面に戻る
Carrier
6:50 PM
数当てゲーム(二桁の数字を当てよう)
十の位
一の位
1
4
これだ！
リセット
十の位は 「上だよ」
一の位は「下だよ」
「もう一度チャレンジ！」
BMI値の画面に戻る
Carrier
6:51 PM
数当てゲーム(二桁の数字を当てよう)
十の位
一の位
4
1
これだ！
リセット
十の位は 「ビンゴ！」
一の位は「ビンゴ！」
「大正解！答は41だよ。」
BMI値の画面に戻る

図２５　数当てゲームプログラムソース画面




BMICalculator.xcodeproj — kazuate.m
Run
Stop
Scheme
Breakpoints
Build Succeeded | Yesterday at 14:57
No Issues
Editor
View
Organizer
BMICalculator › BMICalculator › kazuate.m › -Touchreset_K:
kazuate.h › @interface kazuate
//判定結果は@""である
NSString *result = @"";
// 一の位が1、十の位が4でない場合『おしい。あと十の位だけだ!』と表示する。
if (ichi == 1 && ju > 4 && ju < 4) {
result = @"「おしい。あと十の位だけだ!」";
// 一の位が1、十の位が4ならば、『大正解！答は１４だよ.』と表示する。
} else if (ichi == 1 && ju == 4) {
result = @"「大正解！答は41だよ。」";
// 一の位が１でなく、十の位が4なら『おしい。あと一の位だけだ!』と表示する。
} else if (ju == 4 && ichi < 1 && ichi > 1) {
result = @"「ピンゴ！」";
// それ以外は、『もう一度チャレンジ!』と表示する。
} else {
result = @"「もう一度チャレンジ！」";
}
// 判定結果を結果ラベルに設定
hyouji_K.text = [NSString stringWithFormat:@"%@%@%@%@%@%@%@",
@"十の位は ", result_J ,
@"\n\n一の位は", result_I ,
@"\n\n ", result];
}
- (IBAction)Touchreset_K:(id)sender {
// キーボードを隠す
[self textFieldResignFirstResponder];
ichiText.text = @"";
juText.text = @"";
hyouji_K.text = @"";
//それぞれをクリア
}
@end
//
// kazuate.h
// BMICalculator
//
// Created by on 12/12/17.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import <UIKit/UIKit.h>
@interface kazuate : UIViewController
@property (weak, nonatomic) IBOutlet UITextField *
ichiText;
@property (weak, nonatomic) IBOutlet UITextField *
juText;
@property (weak, nonatomic) IBOutlet UIButton *
keisan_K;
@property (weak, nonatomic) IBOutlet UIButton *
reset_K;
@property (weak, nonatomic) IBOutlet UILabel *
hyouji_K;
- (IBAction)Touchkeisan_K:(id)sender;
- (IBAction)Touchreset_K:(id)sender;
@end