Swiftレクチャー２回目

溝口健太
村上ろまん

お詫びと訂正

```
func hoge(a a:String)->(String){
    return a+"java"
}

func addStr(function function:(String)->(String))->(String){
    return hoge(a:"sore")
}

print(addStr(function:hoge))
```

addStr関数でfunction関数を使ってない

```
func hoge(a a:String)->String{
    return a+"java"
}

func addStr(function function:(String)->(String))->String{
    return function("sore")
}
print(addStr(function:hoge))
```

今日の内容

いろんなUIに触ってみよう

UIの作成について

UIの作成にはコードで作成する場合とSB(story boad)で作成する場合があります。

基本的にはコードでしか作成できないもの以外はSB上でぽちぽちやっていきます

コード上での作成

基本中の基本のボタンを作ってみましょう

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view,
        typically from a nib.
    let btn = UIButton(frame:CGRectMake(100, 100, 100, 100))
    btn.backgroundColor = UIColor.blackColor()
    btn.setTitle("hoge", forState:UIControlState.Normal)
    btn.setTitleColor(UIColor.whiteColor(), forState:
        UIControlState.Normal)
    btn.addTarget(self, action: "btn_click:",
        forControlEvents:.TouchUpInside);
    self.view.addSubview(btn)

}
```

プロパティへのアクセスは”.”をつける！　後ほど説明する

コード上での作成

- let btn = ~ //buttonの位置とサイズを指定して作成
  (btn.frame = でも可 コードを短縮するため省略)

- btn.backgroundColor = ~ //buttonの背景色設定

- btn.setTitile(“buttonのタイトル”,forstate:ボタンの状態の設定)

- btn.setTitileColor(~) //buttonのタイトル色設定

コード上での作成

- btn.addTarget(“だれに”,action:”呼び出す関数名:”,forControlEvents:”どんなイベントの時か”)

ここでのselfとはボタンのことだからボタンにイベントが行われた際に関数を呼び出すイメージ

self = 自分自身

コード上での作成

ちなみにobjc的に書くと

self.btn.backgroundColor = ~ となります

objcは自クラス内で作成したプロパティへのアクセスには全てself.をつけなくてはいけません それに比べるとswiftは幾分楽

省略しなくても書けるっぽいけど面倒

コード上での作成

- self.view.addSubview(~)

ボタンを画面上に追加する

コード上での作成

```
func btn_click(sender:UIButton){
    sender.setTitle("abc",forState:UIControlState.Normal)
}
```

ボタンをクリックした際の処理を行う関数

ついで

コード書いたからわかるとは思うけど。

UIの部品もクラスでできています。

UIhogeクラスが存在すると仮定して説明すると

- btn.hoge クラス変数
- btn.hoge() クラスメソッド

となります。

ついで

あと、慣れてくればわかると思うけどUIButtonクラスはもちろん他のクラスを継承しているのでそのクラスもきちんと使えます

NSObject

↑

UIResponder

↑

UIView

↑

UIControl

↑

UIButton

SBでの作成

一緒にやってみよう

SBでの作成

Storyboad(UIを管理する画面)と
ViewController(プログラムの実装部)
との接続は２画面表示にして

ctr+ドラッグ

でおk

接続に関して

- Outlet

指定した名前のプロパティを作成

プロパティ…アクセサメソッド（setter,getter）

を使って操作(“.”でアクセス)

- Action

UIを操作した際のイベント処理(関数を作ってくれる)

練習

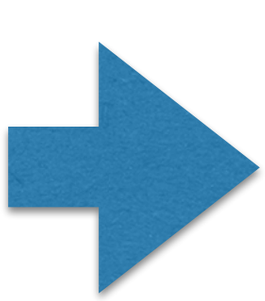
配列 hoge = [“a”,”b”,”c”,”d”,”e”]がある。この全ての要素をlabelに表示し、ボタンを押すと配列の最後尾の要素を削除するようにしてください。また、削除した際にその結果をlabelに表示することも忘れずに。
また、(速攻で終わった人は)配列の要素が無くなった際にはエラー文をラベルに表示してください。
文字列連結してやってもいいです。

配列.removeLast()で消去
配列.isEmptyで空ならTRUE
ラベル名.text でラベル名を決定

実行例1

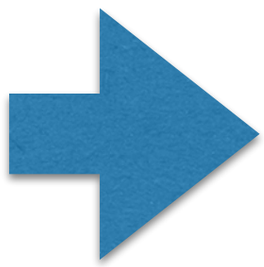
abcde

削除



abcd

削除



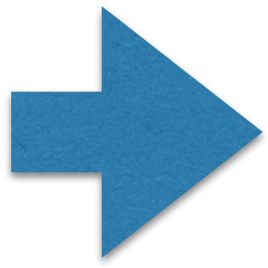
error

削除

実行例2

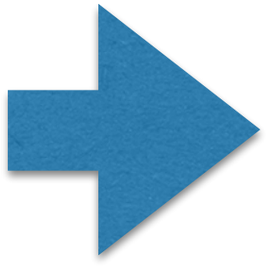
["a", "b", "c", "d", "e"]

削除



["a", "b", "c", "d"]

削除



error

削除

解答1

```
class ViewController: UIViewController {
    @IBOutlet weak var label: UILabel!
    @IBOutlet weak var deleteBtn: UIButton!
    var hoge = ["a","b","c","d","e"]

    override func viewDidLoad() {

        super.viewDidLoad()
        // Do any additional setup after loading the view,
            typically from a nib.a
        label.layer.borderColor = UIColor.blackColor().CGColor
        label.layer.borderWidth = 2.0
        showArray(hoge)
    }
```

解答1

```
@IBAction func deleteBtn(sender: AnyObject) {
    if hoge.isEmpty{
        label.text = "error"
    }else{
        hoge.removeLast()
        showArray(hoge)
    }
    /*if hoge.isEmpty{…}*/   //配列が空になった際にすぐに表示したけ
        れば書く ここがなければ次のボタンクリックでエラー表示
}

func showArray(hoge:[String]){
    var str = ""
    for value in hoge{
        str += value
    }
    label.text = str
}
```

解答2

解答1のshowArrayを削除して、showArray関数をすべてlabel.text = String(hoge)に置換

配列hogeの要素が[String]のため

画面遷移について

主に２つ

- segueの利用
- presentViewControllerの利用

UINavigationBarとか使えたらいいね

画面遷移について

とりあえず一番簡単なボタンを押した際に遷移する場合をやってみましょう

画面遷移について

1. SB上で新たなViewControllerを作成

2. 新たなViewControllerの管理をするためのファイルを作成

3. 作成したファイルが新たなViewControllerに対応していることをViewControllerへ教える

画面遷移について

もちろんボタンクリック等以外でも値の変化などでも遷移することができます。(今回はボタンクリックに条件をつけて遷移させます)

画面遷移について

```
class ViewController: UIViewController {

    @IBOutlet weak var btn: UIButton!
    var count = 0;
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view,
            typically from a nib.
    }

    @IBAction func btn_Click(sender: AnyObject) {
        if(count > 0){
            performSegueWithIdentifier("second",sender: nil)//指定
                したsegue名の先に飛ぶ つまりsecondViewControllerへ
            count = 0
        }else{
            count++
        }
    }
```

画面遷移について

遷移する際に値渡しをすることができます。とっても便利だね。

画面遷移について(値渡し)

ViewController.swift

```
class ViewController: UIViewController {

    @IBOutlet weak var btn: UIButton!
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

    @IBAction func btn_Click(sender: AnyObject) {
        performSegueWithIdentifier("second",sender: nil)//指定したsegue名の先に飛ぶ つ
            まりsecondViewControllerへ
    }
    /* 遷移する際にSBで指定したsegue先に飛ぶ際に文字列を渡す*/
    override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
        if(segue.identifier == "second"){
            let secondVC = segue.destinationViewController as! secondViewController
            secondVC.name = "hogehoge"

        }
    }
```

画面遷移について(値渡し)

secondViewController.swift

```swift
class secondViewController: UIViewController {

    @IBOutlet weak var label: UILabel!
    @IBOutlet weak var back_btn: UIButton!
    var name = ""    //初期化しないと出来ないっぽい

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        label.text = name
    }
```

言い忘れたけど基本的に初期値を入れないでの変数宣言はなるべく使わないようにしましょう

画面遷移について

前のViewへ移る処理

secondViewController.swift

```
    @IBAction func back_btn(sender: AnyObject) {
        dismissViewControllerAnimated(true, completion: nil)
    }
```

第一引数:アニメーションのオンオフの設定

第２引数:とりあえずnil クロージャー関連らしいがよく知らん

Tips

asはキャストすることができます

as! でダウンキャスト

as? でアップキャスト

ここでlet secondVC = ~ でas以降がない場合secondVC.nameはエラーになります

Tips

なぜエラーになるのか少し考えてみよう

ヒント クラス構造を考えよう

Tips

Ans.SecondViewはUIViewController
の派生クラスで、メンバ変数である
nameはUIViewControlllerでは定義さ
れていないからです。

ちなみにas!でキャストに失敗した場合ランタイムエラー
as?でキャストに失敗した場合はnilが返ります

画面遷移について

次はpresentViewControllerで遷移させてみよう!

画面遷移について

Segueとの違いは遷移先の
ViewControllerのオブジェクトを1度
生成(インスタンス化)して呼び出す必
要があります

画面遷移について

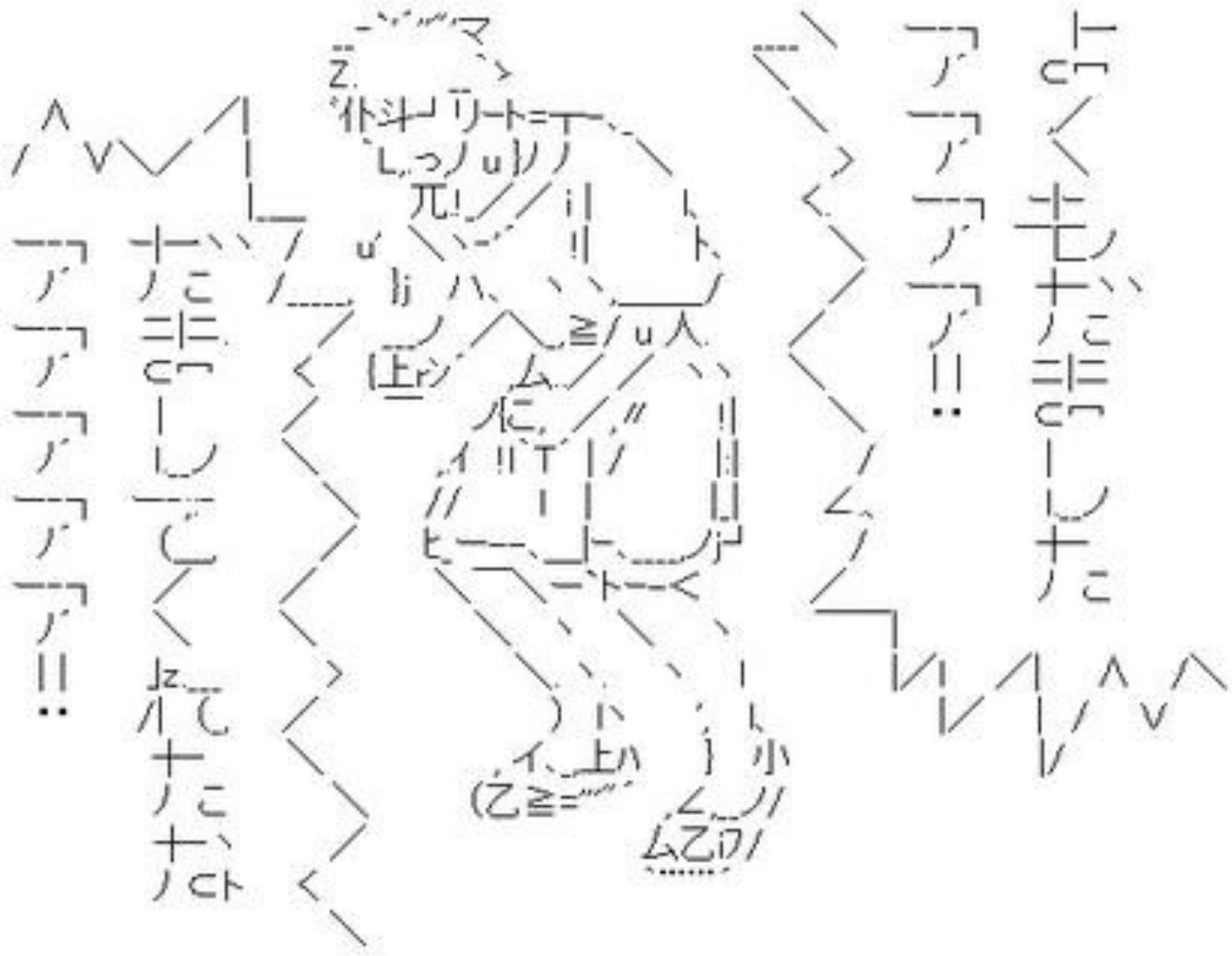
```
@IBAction func btn2_Click(sender: AnyObject) {
    let secondVC = secondViewController()
    presentViewController(secondVC, animated: true,
        completion: nil)
}
```

ビルドして確認してみよう！

ここでは第一引数を遷移先のVCにしているよ

画面遷移について

ビルドも上手くいったのになんでや!?



SBのUIが表示されない問題

なんでできないんだろう？

ViewControllerをインスタンス化してSB上で作成したUIをビルド時に表示させるには一手間かかります。

全てコード上でUIの作成や、SB上で行うとこのような現象にはなりません

SBのUIが表示されない問題

解決策としては、SB上で遷移先に
Storyboad IDをきちんとつけましょう！

画面遷移について

```
@IBAction func btn2_Click(sender: AnyObject) {
    let secondVC = UIStoryboard(name: "Main", bundle: nil).
        instantiateViewControllerWithIdentifier("secondVC")
        as! secondViewController
    secondVC.name = "sorejava"
    presentViewController(secondVC, animated: true,
        completion: nil)
}
```

UIStoryboardクラスの第一引数にSBのファイル名,第２引数にファイルのパス指定(ここでは必要なし)

instant~Identifierの引数にSB IDを持たせインスタンス化してsecondVCクラスにダウンキャスト

練習

UIButtonをOutlet接続して、その名前をbtn1,btn2にする。btn1をクリックした際にはsegueでの遷移を(関数呼び出しの方)、btn2をクリックした際にはpresentViewControllerでの遷移をしてください

btn1は先ほど作ったプロパティの名前を変更してください
簡単だと思うけど多分できないと思うよ

出来なかった人へ教訓

途中でプロパティ名や接続してる関数名を変える/消す場合は接続が続いているのできちんとSB上で切りましょう

デリゲートについて

デリゲート…処理を委譲するもの

-ex George君は明日カナダに帰らなくてはいけないが、明日はjava言語2の講義をしなくてはいけません。本来なら１限から学校に行かなくては行けませんが帰省したいので翔太郎くんに講義をやらせることにしました。

デリゲートについて

- Gerge君…delegateをお願いする(渡す)クラス
- 翔太郎…delegateを許諾(受け取る)するクラス
- java言語2の講義…delegateで渡される処理

デリゲートについて

今回は作ったコードを引き続き使って
secondVC→VCへとdelegateさせて
値渡しをします。

遷移先のボタンが押された際の処理を
遷移元にやってもらいます。

デリゲートについて

secondViewController.swift(delegate元)

```
protocol testDelegate{                          //protocolはjavaでいうインタ
    ーフェイスに近い
    func hogedelegate(hoge:String)      //delegateメソッドを宣言
}   //宣言したdelegateメソッドは必ずコーディングすること!

class secondViewController: UIViewController {

    @IBOutlet weak var label: UILabel!
    @IBOutlet weak var back_btn: UIButton!
    var delegate:testDelegate? = nil    //作成した testDelegateを宣
        言
    var name = ""    //初期化しないと出来ないっぽい

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        label.text = name
    }
```

デリゲートについて

secondViewController.swift(delegate元)

```
    @IBAction func back_btn(sender: AnyObject) {
        if label.text == "segue"{
            delegate?.hogedelegate("By segue delegate is
                success!")     //送信先へ
        }else{
            delegate?.hogedelegate("By presentViewController
                delegate is success!")     //送信先へ
        }
        dismissViewControllerAnimated(true, completion: nil)
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

デリゲートについて

ViewController.swift(delegate先)

```
class ViewController: UIViewController,testDelegate{

    @IBOutlet weak var btn1: UIButton!
    @IBOutlet weak var btn2: UIButton!
    @IBOutlet weak var delegateLabel: UILabel!  //送られてきた文字列
        を格納

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view,
            typically from a nib.
    }

    @IBAction func btn1_Click(sender: AnyObject) {
        performSegueWithIdentifier("second",sender: nil)//指定した
            segue名の先に飛ぶ つまりsecondViewControllerへ
    }
```

デリゲートについて

ViewController.swift(delegate先)

```
    @IBAction func btn2_Click(sender: AnyObject) {
        let secondVC = UIStoryboard(name: "Main", bundle: nil).
            instantiateViewControllerWithIdentifier("secondVC")as
            ! secondViewController
        secondVC.delegate = self
        secondVC.name = "presentViewController"
        presentViewController(secondVC, animated: true,
            completion: nil)
    }

    /* 遷移する際にSBで指定したsegue先に飛ぶ際に文字列を渡す*/
    override func prepareForSegue(segue: UIStoryboardSegue,
        sender: AnyObject?) {
        if(segue.identifier == "second"){
            let secondVC = segue.destinationViewController as!
                secondViewController
            secondVC.delegate = self      //delegateの受け取り先はこ
                こだと決める
            secondVC.name = "segue"
        }
    }
```

デリゲートについて

ViewController.swift(delegate先)

```
    func hogedelegate(hoge:String) {
        delegateLabel.text = hoge
        print(delegateLabel.text)
    }
```

secondVCで作成した関数をVC内で処理していることがこれでわかりましたね。

最後に

デリゲートは用途に応じて使いましょう。UIクラスの中にもデリゲートクラスが定められているものやAppDelegate.swiftもデリゲートです。これらは元々決められている条件で呼び出してくれるのでその関数内に処理を書くだけです。とっても楽だし便利ですね。