第3回
ルールとアイデア出しと

Toru.Inoue@KISSAKI.tv

Agenda1/1

- 今日やる事
  - 0.籤、番号発表っつーか未だの人居てスマソ
  - 1.iPhoneプログラムについて
  - 2.最終回までに作りたいアイデアを答えよ(100,000点)(妙に震える大学 2010年 本試)
  - 3.アイデアから仕様を作る(序)

Drive1/44

1. iPhoneプログラムについて

Drive2/44

1-0-0:正解の中心で、愛をさけぶ

0.一、二回目の授業の、プログラム解説をします。

自分の作ったPrototypeか、再配布しているPrototypeプロジェクトを開くかしてな。

PW>PrototypeAppDelegate.m　　の、中の、//2とか、//3　と、

```
#pragma mark Application lifecycle

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    /*(0)*  NSLog(@"むにゃむにゃ");    /* */

    //IBを使用したケース
    /*(2)*/ prototypeViewController = [[PrototypeViewController alloc] init];   /* */
    /*(3)*/ [window addSubview:prototypeViewController.view];                    /* */
```

PW>PrototypeAppDelegate.h　　//4と//5のトコについて

```
//

#import <UIKit/UIKit.h>
/*(4)*/ #import "PrototypeViewController.h"                /* */

/*(4m)* #import "ManualPrototypeViewController.h"          /* */


@interface Prototype_CFBF63D6_86E8_48E8_BAA7_7AAF8899DAAAAppDelegate : NSObject <UIApplicationDelegate> {
    UIWindow *window;
    /*(5)*/ PrototypeViewController * prototypeViewController;  /* */
```

結構意地悪してまして、
大文字小文字の嫌がらせ、他一件があります。

Drive3/44

1-0-1:正解の中心で、愛をさけぶ

1.一回目の授業の、プログラム解説をします。

実は：2つ、嫌がらせがあるぞ！
一つは大変不幸なミスだけどココだけの秘密だ！

先ずはミスの告白から。
2~5の内容をファイルとか無視して並べて書くと、、

```
2.prototypeViewController = [[PrototypeViewController alloc] init];
3.[window addSubview:prototypeViewController.view];
4.#import "PrototypeViewController.h”
5.PrototypeViewController  * prototypeViewController;
```

こんな感じになるのですが、

Drive4/44

1-0-2:正解の中心で、愛をさけぶ

2.書く順番を変えてみる

実は、こん44番に書いてもOKです。
4、5を先に書いて、
2、3を後に書いてます。

```
4.#import "PrototypeViewController.h”
5.PrototypeViewController  * prototypeViewController;
2.prototypeViewController = [[PrototypeViewController alloc] init];
3.[window addSubview:prototypeViewController.view];
```

初回の授業の時だと、パネルの関係でプログラムのエラーが出やすいように意地悪してました。

この時点で、順番替えについて、
「なんだよそういう事かよ！！　やっと判った、なら最初からそう書けよ！！」
ってガッテン出来る人、いたら後で謝ります。ぶっちゃけ判りにくいだけで逆にする必要なかった。

責めたければ責めてみればいいじゃない！　俺の心臓はココだァァァァ！！

Drive5/44

1-1-0:関連性を見てみよう

0.色分けしてみた。

```
4.#import "PrototypeViewController.h”
5.PrototypeViewController  * prototypeViewController;
2.prototypeViewController = [[PrototypeViewController alloc] init];
3.[window addSubview:prototypeViewController.view];
```

同じ色のモノは、まったく同じ文字で出来てる文章です。

で、実は、

同じ色の内容は、「同じふうに変えれば、変えても動く」のがあります。

Drive6/44

1-1-1:関連性を見てみよう

1.例えば、

```
4.#import "PrototypeViewController.h”
5.PrototypeViewController  * pVCont;
2.pVCont = [[PrototypeViewController alloc] init];
3.[window addSubview:pVCont.view];
```

5のprototypeViewControllerを、pVContに変えてみました。
2、3のprototypeViewControllerも、pVContに変えてみました。

これで動くの？　っていうと、動くんですね。コレが。

ただし、どこか一カ所でも変え損ねたりすると、エラーがでます。

丁度前回の皆さんの状態が、
“どっか一カ所でも違う”　っていうのを含んでいたと思います。

Drive7/44

1-1-2:関連性を見てみよう

2.プログラムの内容は文字。

プログラムは、文字でいろんなものを書く都合上、

文字の内容が一緒かどうかで、
プログラムに登場する要素が同じかどうか判断してるんです。

ここでは、登場するprototypeViewControllerを
全部、全て、須くpVContに変えれば、動く、と。

実は、プログラムから見たら、
「文字の内容が一緒であれば同じもの」、という原則があります。ミソです。

まーぶっちゃけ、それ以外の方法で動いてるプログラムなんてあんまりないんですが。

逆に、大文字小文字がちょびっとでも違うと、
プログラムさんは激しく拒絶してエラーを出したりしてくれます。

Drive8/44

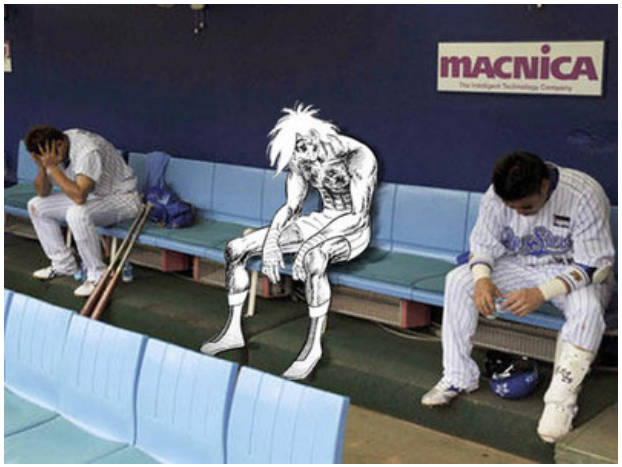
1-1-3:関連性を見てみよう

3. エラー、おぼえていますか

PrototypeViewController
と
prototypeViewController

大文字小文字でわざと一カ所しか違わないように書いたのは、
理由はあるんだけどね。プログラムを勉強し始めたときに嵌っておくと
いろいろ得する内容だからです。

大文字小文字、たったこれだけで、エラーが出ます。
また、こういうプログラムをされるとまあ、迷惑です。



第一回から仕込んで4週間。長かった。
嘘をつくのは私のような正直者には
身を切られるような痛みの伴うモノなのですよ。

Drive9/44

1-1-4:関連性を見てみよう

4.そのほか、今まで特に言ってなかったけどルール

5.PrototypeViewController * pVCont; の米印ってなに!? テリーマンなの!?
みんなは誰それって感じ!!? (→ポインタ)

とか、

そのあたりについてもiPhoneに関わる範囲で話しますので、
まあ、地味に地道に行きましょう。

Drive10/44

1-2:変えられないものもある(キリッ)

さっきはprototypeViewControllerをpVContに変えました。それ以外の部分について、
じゃあ、

4.#import “PrototypeViewController.h” とかはPVController.hに変えられるのか

というとぶっちゃけ変えられます。ただし、

PrototypeViewController.hを変えたければ
PrototypeViewController.mファイルと
PrototypeViewController.hファイルのファイル名を変え
ファイルの中のPrototypeViewControllerって書いてあるところを全て変え
ビルドパスを書き換え(何それ!?)
インターフェースビルダーファイルとの接続(何それ!!?)を書き換える
必要があります。 大変なのだよ。

返る事の大変さに差があるのは、
変えようとしている対象が、実は巨大なものだからです。
こいつは、大分先でやります。

Drive11/44

1-3-0:画面に表示する、の中身を探ってみるなど

0.変化が目に見えるところから始めよう

なにかを画面に表示する、からプログラムを調べてみましょう。
プロジェクトPrototypeでは、PrototypeViewControllerというファイルを作って、
IBでいろいろやっちゃって、画面に表示させました。

何処の部分で、PrototypeViewControllerをiPhoneの画面に表示してるか、
見当は着くかしら？　　っていうか書いてあるんだよね。。

//3のとこです。ついでに内容を色分けしてみた。

```
3.[window addSubview:prototypeViewController.view];
```

window addSubview: と書いてあって、さらに先に、
prototypeViewController.viewとか書いてある。

iPhoneのプログラムでは、これ、
prototypeViewControllerという物体の持っている要素.viewを、window　に
addSubview:します、という意味になる内容なんです。

Drive12/44

1-3-1:画面に表示する、の中身を探ってみるなど

1. window addSubview:prototypeViewController.view　つづき

こんな風に、Objective-Cだと、

Cを、AにBします

というようなプログラムを、

(カッコ開ける [ )　A　(スペース)　B(コロン:)　C　(カッコ閉じる])　(セミコロン ;)

という風に書きます。

[A B:C];

ぶっちゃけ「決まり事なんで、、そういう決まりになってるんです」
としか言えないんですが、覚えなきゃいけない事だと思ってください。

因にこの決まり、なんでこうなったかの理由はありますが、プログラム史の内容です。
プログラム史の授業やりたいな。ほぼ雑学だしな。

Drive13/44

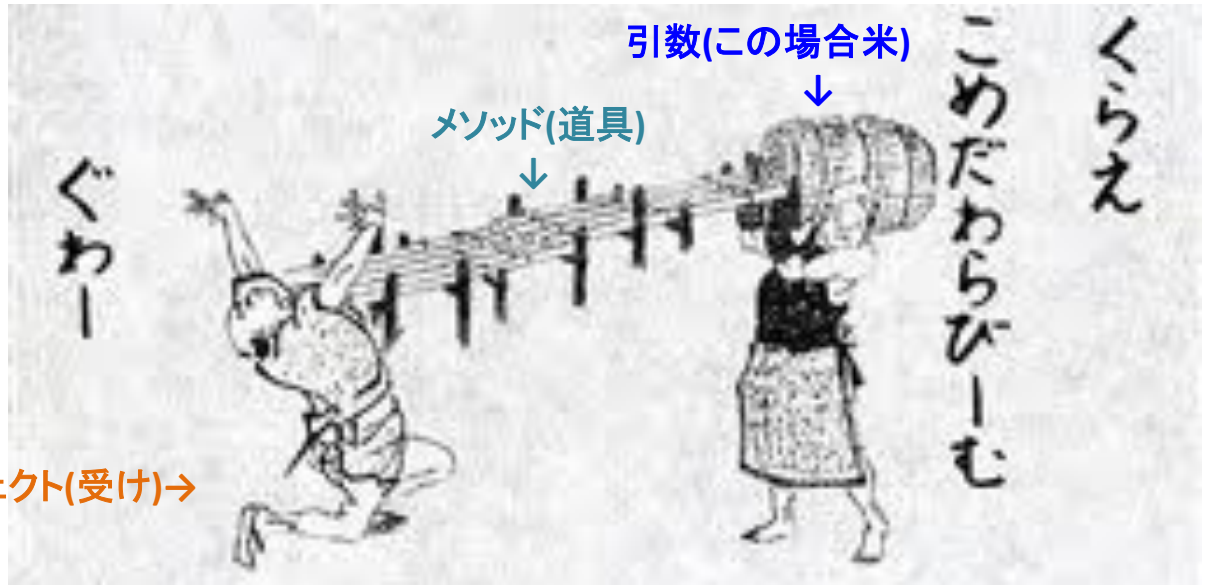
1-3-2:画面に表示する、の中身を探ってみるなど

2.それぞれ名前があるんでやんす

特にAの部分の事を、オブジェクト、もしくはレシーバ(受け、攻めでいえば受けです)、
特にBの部分のことを、メソッド、
特にCの部分の事を、引数（ひきすう） と呼びます。

オブジェクトに対して、メソッドを使って引数を突っ込んでなんかさせる というのが、
Objective-C(に限らずこの手の)プログラムのキモというか、実質的な動作内容です。

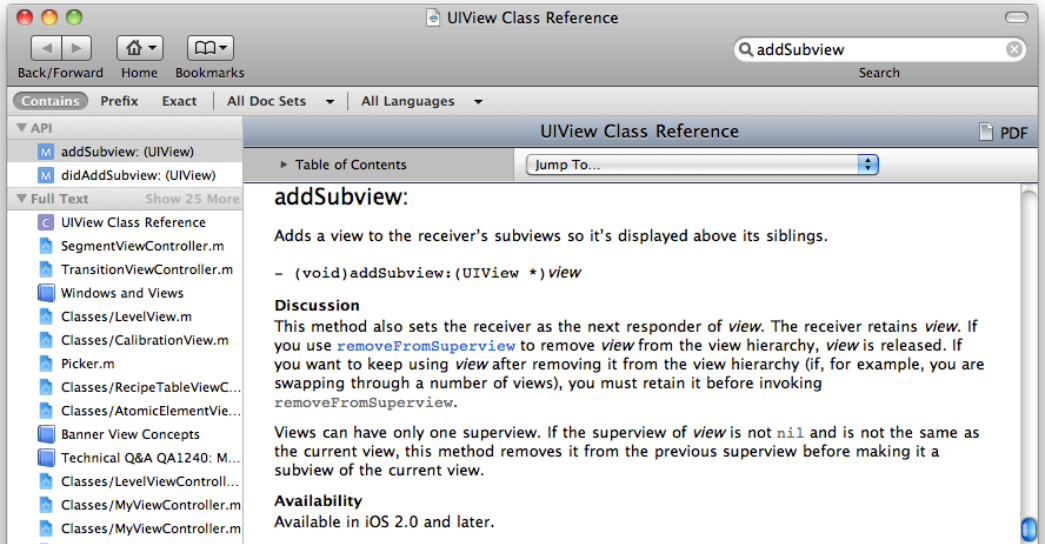
なんか
抽象的な図



Drive14/44

1-4-1:で、画面表示をもうすこし詳しく。

1.ちょっともどって、辞書でaddSubviewを牽いてみよう



addSubview:

Adds a view to the receiver's subviews so it's displayed above its siblings.

-(void)addSubview:(UIView *)view

牽いてみたら、こんな感じに書いてある。

ぶっちゃけ訳すと、

"ビューをレシーバーのサブビューに加え、
それらの兄弟(既存のビュー)の上に表示します"とある。

Drive15/44

1-4-2:で、画面表示をもうすこし詳しく。

2.よく読む
で、- (void)addSubview:(UIView *)viewとか書いてあるのだが、
これが「このメソッドの例文」みたいなもの。

window addSubview: と見比べると、ま一辞書で牽いたんだから
そりゃ一緒だろうと思うんだが、似てるのが判ると思う。

色付けよう。

-(void)addSubview:(UIView *)view

ここで、- (void) とかの部分は、後で説明する。(→返り値 2回目)

(UIView *)viewの部分は、このメソッドで「:の後にくっつけられる引数」を表している。
(UIView *)vi44

またテリーマンかよ！
そう、さっき出てきたのと同じ意味であろう、何か、です。

Drive16/44

1-4-3:で、画面表示をもうすこし詳しく。

3.「ポインタ、、？ そういえば聞いたことがある」

＊は、テリーマンではない。

ポインタ(って何)とか言われる物を表す記号で、これで結ばれているモノどうしは、

A * a とか書くと、

Aでできている、名前がaのもの という意味になる。

たとえば、

人間 * 井上徹

とか。 ほんとはもっと深淵なアレやコレがあるのだが、今はやらない。(→ポインタ)

今は、「左側ので出来てる右側の名前のヤツ」とか覚えてもらえればOKです。

PrototypeViewControllerとpVContの関係も、これでスッキリしたかしら、しないかしら。

PrototypeViewController * pVContで、

PrototypeViewControllerで出来てる名前がpVContのもの、という内容になる。

Drive17/44

1-4-4:で、画面表示をもうすこし詳しく。

4.今までのおさらい

addSubview:(UIView *)viewということは、addSubview:の引数には、
UIView でできている、viewとかいうもの、が着くということになる。

addSubview:(UIView *)view

ということは、次の2つが言える。

addSubview:のコロンの先には、UIViewで出来てる何かが、引数として入る
UIViewで出来てる何かでないと、addSubview:できない

3.[window addSubview: pVCont.view];　とかありましたね。

ということは、引数であるpVCont.view　は、UIViewで出来てる物体らしい。
だからaddSubview:できて、表示される、と。

.view　の、ドットの内容については、今度。(→ドットの内容)

Drive17/44

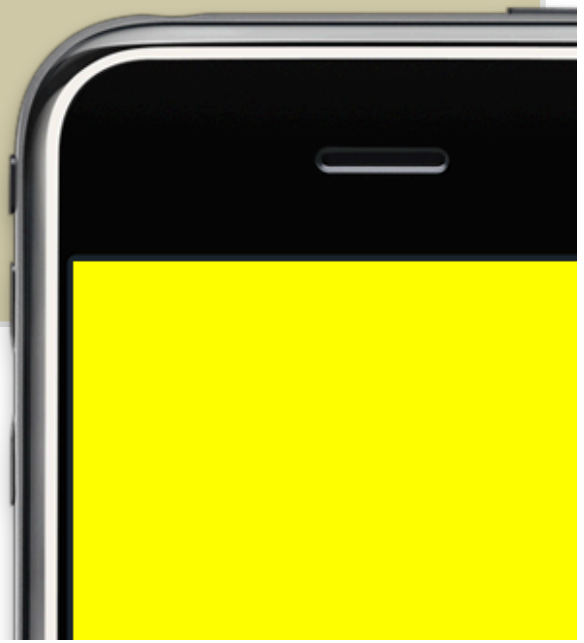
1-5:勝手にUIViewの何かを作っても、表示されるん!?

→されます。折角だからやってみようぜ！！

Prototypeプロジェクトを開いてみよう。
PW>PrototypeAppDelegate.m　を開いて、下記を追加

```
    UIView * hoatyaa = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];
    [hoatyaa setBackgroundColor:[UIColor blueColor]];
    [window addSubview:hoatyaa];

    [window makeKeyAndVisible];
    return YES;
}
```



blueColorの部分は、yellowColorとかも使える。
と、どうなる？

起動したら、真っ青とか真っ黄色とか好きな色になったかな？

Drive18/44

1-6-0:メソッドについて

0. メソッドの種類

実は、さっき説明したメソッドは過去に何度も作ってきてます。
[むにゃむにゃ ぺらぺら]; とか、カッコがあったら、だいたい、、、メソッドです。

今さっき書いた、
hoatyaaの近所 とか、よーっくみると、そこかしこに書いてあります。

1↓ 2↓

```
    UIView * hoatyaa = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];
3→  [hoatyaa setBackgroundColor:[UIColor blueColor]];
    [window addSubview:hoatyaa];
```

4↑

5↑

Drive19/44

1-6-1:メソッドについて

1.メソッドの種類

```
        1↓                    2↓
    UIView * hoatyaa = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];
3→  [hoatyaa setBackgroundColor:[UIColor blueColor]];
    [window addSubview:hoatyaa];        4↑
              5↑
```

あれ、1とか4とか、コロン：なくね？　つか引数なくね？

オブジェクトとメソッドしか無いんじゃね？

そのとおり。

メソッドによっては、引数の数は0個、つまり無かったりします。反対に一杯あったりも。

Drive20/44

1-6-2:メソッドについて

1.飲み込まれております

なんか、食い込んでるのがある。

1↓　　2↓

```
    UIView * hoatyaa = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];
3→  [hoatyaa setBackgroundColor:[UIColor blueColor]];
    [window addSubview:hoatyaa];
```

4↑

5↑

1は2に、

```
    UIView * hoatyaa = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];
```

3は4に、食い込まれております。

```
    [hoatyaa setBackgroundColor:[UIColor blueColor]];
```

とかね。こういうパターンもあります。 内容は今度。

Drive21/44

1-6-3:メソッドについて

2.前回、Calculatorにて、メソッドを一つ、作ってもらっています。

-(void) update ってやつです。

PW>CalcViewController.hのほうには、

```
-(void) update;
```

PW>CalcViewController.mのほうには、

```
-(void) update {
	m_field.text =  [NSString stringWithFormat:@"%d", m_sum];
}
```

とか、書きました。

Drive22/44

1-6-4:メソッドについて

3.知らない間に使ってた

さらに、PW>CalcViewController.m　では、使ってもいます。

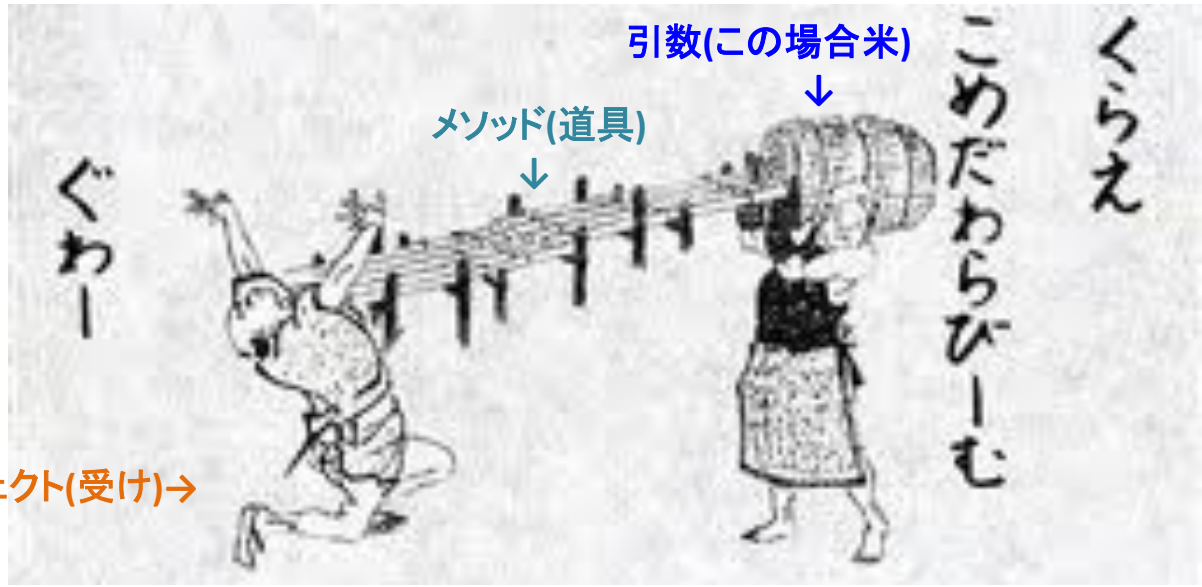
```
-(IBAction) tapped0 {
        m_sum += 0;
        [self update];
}
```

Drive23/44

1-6-5:メソッドについて

4.使い方の解説

さっきは



とか、やってましたが、

[self update]; オブジェクト/受けにあたる部分には、

self とか書いてあります。 これは、自分を表してます。
つまり自分受け。高等テクです。

Drive24/44

1-6-6:メソッドについて

6.メソッドを作る手順☆

.hと.mファイルの両方に、書き込む必要があります。

２つのファイルに書くメソッドの内容、その違いは明確で、

.hのほうには「中身のないメソッド」を書き、
- (void) メソッド名;
- (void) メソッド名:(引数の種類)名前;

.mの方には、「中身を含めた処理」を書きます。

```
- (void) メソッド名 {
        //内容!
}
```

Drive25/44

1-7:(ドット .)ってNA-NI!?　ほか三本+αでお送りします

サザエではございません。

命名ルール、返り値の話、ドットの内容、これらは、もうちょいプログラムに慣れてから、やります。

具体的には次回やります。

Drive26/44

2. 最終回までに作りたい
アイデアを答えよ(100,000点)
(妙に震える大学 2010 本試)

Drive27/44

2-0:ここからは、仕様とアイデアの話です

0.だってそういうタイトルなんだもん

前回に引き続き、仕様の話をしましょう。

各人の持ってきたアイデアに対して、
・どうしたら出来るのか = 出来上がり時の仕様を考えることにしましょうかねェ。

っていっても全員のアイデアを受け付けるのは

面倒 <<< おもしろそう！

なので、**フィルタ**を通して考えさせてみようと思います。

今日はこっからハンズオン形式な。

Drive28/44

2-1:ここからは、仕様とアイデアの話です

1.スッゲー手探りで頑張る仕様の作り方

アイデア持ってきたよね。

きっとね、漠然とね。きっちりかっちりとね。
そしたら、次の3つの**フィルタ**を通した上で、アイデアを教えてください。

で、、、、、まずはフィルタについて説明します

フィルタは全部で3枚あり、

フィルタその１		フィルタその２		フィルタその３
果て	→	実現するための何か	→	その構成要素や内容

の3マイになってます。

Drive29/44

2-2:ここからは、仕様とアイデアの話です

2.フィルタその１:

アイデアから、そのアイデアが実現された果てに、
「こうなるといいな！」という願いを考えてください。

特に今回は、iPhone講座なんで、iPhoneを使って、どんな願いが
「叶った」と判れば、アイデアは完成した事になるのだろう、と考えてください。

Ex:メール(コレもまあ、誰かのアイデアの結晶なわけだ。)
今現在、めっちゃ使われてる、使われて使われて使われまくってる。

果ては、、→新しいコミュニケーションができた

今は例なので、すでにこの世にある「メール」を題にしてますが、
皆さんは自分のアイデアについて考えてほしい。
この後に続く、フィルタその２、フィルタその３の解説の後にでも。

Drive30/44

2-3:ここからは、仕様とアイデアの話です

3.フィルタその２:

その「叶った」を実現するために、どんな機能、能力、部品、見た目とかが必要だろう

その１の→の先の物事の、中身について、特徴を洗い出すように
列記してみてくださいな。

Ex:新しいコミュニケーションができた

→コミュニケーションを行う機能
→コミュニケーションの内容を作る/書く機能
→なにこれ素敵！　って思わせる見た目
→殉職を厭わない屈強なファン、を作り出す仕組み

Drive31/44

2-4:ここからは、仕様とアイデアの話です

4.最後、フィルタその３:

フィルタその２の内容は、具体的に細かく書くと、どんな感じ？

Ex:　コミュニケーションを行う機能
→友人になんかコミュニケーションを出す
→友人がそれを受け取る
→友人がそれを見る、読む
→友人がそれに返信できる
→自分がそれを受け取る…ループ

コミュニケーションの内容を作る/書く機能
→受け取る相手を決める
→内容を書く(文章)
→画像とか文字以外の情報をセットする

etc

いっっっぱい、あると思うよ。　　使ってる時の行為や動作をイメージしてほしい。
この時点で末端の項目が他の項目とダブるのはよく有る事なので、
気にせずガンガンイメージしよう。　　さすがに末端の項目数が30を超えたら、考える。
相談にきてな。

Drive32/44

2-5:ここからは、仕様とアイデアの話です

5.ちょっと思い出す　前回の電卓、、だと？

前回、計算機を作ったと思います。

足し算しかできないアレね。
２桁以上の値を入れられないアレね。

仕様というか作る前の宣言的に「足し算ができる」しか盛り込まなかったので
あんな**残念な感じ**になりましたが、、

今回のフィルタその3みたいな事をしてから作れば、
まあまだちょっとはマシになったはず、です。たぶん。きっと。

サンプルその2としてやってみよう。

Drive33/44

2-6:ここからは、仕様とアイデアの話です

6.電卓の場合の、フィルタ443

フィルタその１：普通に計算、割り勘できた。

普通が一番。だからlogとかイラネ。

→足す引く掛ける割るがかろうじてできる、電卓

フィルタその２：

足す引く掛ける割るがかろうじてできる、電卓

→計算した結果を見る機能

→計算してイライラしない見た目(new)

→１０桁くらいまでの整数を扱える機能(new)

→四則演算ができる機能(new)

→少数点第二位まで表示する機能(new)

フィルタその３：

計算した結果を見る機能

→計算結果を表示する

計算してイライラしない見た目(new)

→今なんの四則の計算してるかわかるように、ボタンが光る　etc,,,,

Drive34/44

2-7:ここからは、仕様とアイデアの話です

7.ちなみに井上からもサンプルを1つ。

新規、、ナヤミングナウ。

マインドマップ型電卓かな。 iPadでやるかな。 どんだけ電卓好きなんだ。

アイデア：計算をマインドマップの記法で表現できるツール。iPad向け。
と、その計算式を他人と共有できるサイト/サービス。
今後、共同購入や所有が流行るとおもうので、そういう物品売買と絡める、計算ソフト/サービス

フィルタ1： このアイデアを実現した果てに、どんなことがあるといいかな。
めっちゃ使われてる
→全国の皆さんのやりくりがウマくなる
→シェアリングの代名詞になる
買える、売れる、募(つの)れるなど。ご家庭の経済がすべてココでやり取りされる
→国家アカウントが出来る

Drive35/44

2-8:ここからは、仕様とアイデアの話です

8.フィルタその２:どんな内容があるかな

シェアリングの代名詞になる、という項目が、いろいろイメージしやすそうだ。
(買える、売れる、募れるなど。ご家庭の経済がすべてココでやり取りされる)

→計算する機能

→図的に計算する機能

→計算結果を並び替える機能

→計算内容を変更する機能

→判りやすいインターフェース

→保存する機能

→共有する機能

→決済する機能

Drive36/44

2-9:ここからは、仕様とアイデアの話です

9.フィルタその3-1　　フィルタその3は大分多い。

計算する機能

→数字と式を入力したら計算結果が出る
→数字は10桁まで入るようにする
→小数点以下は1桁まで出すようにする

図的に計算する機能

→数字と式を画面で自由に繋ぎ、置ける計算結果を並び替える機能
　きっと画面がぐちゃぐちゃになるので、(左)計算開始→計算結果(右)　と
　なるように並び替える

計算内容を変更する機能

→一度入れた数字を変えると計算結果が変わる
→一度入れた四則演算を書き換えると計算結果が変わる
→繋ぎ変え、移動で計算結果が変わる
→商と余りとかで分岐する機能、//←あ、やべ、一個新機能がでちゃった。

商と余りとかでマップが分岐する機能//←というわけでフィルタ2に改めて追記

→とりあえず割り算で商と余りに分かれる
→それ以外の計算については今は考えない。

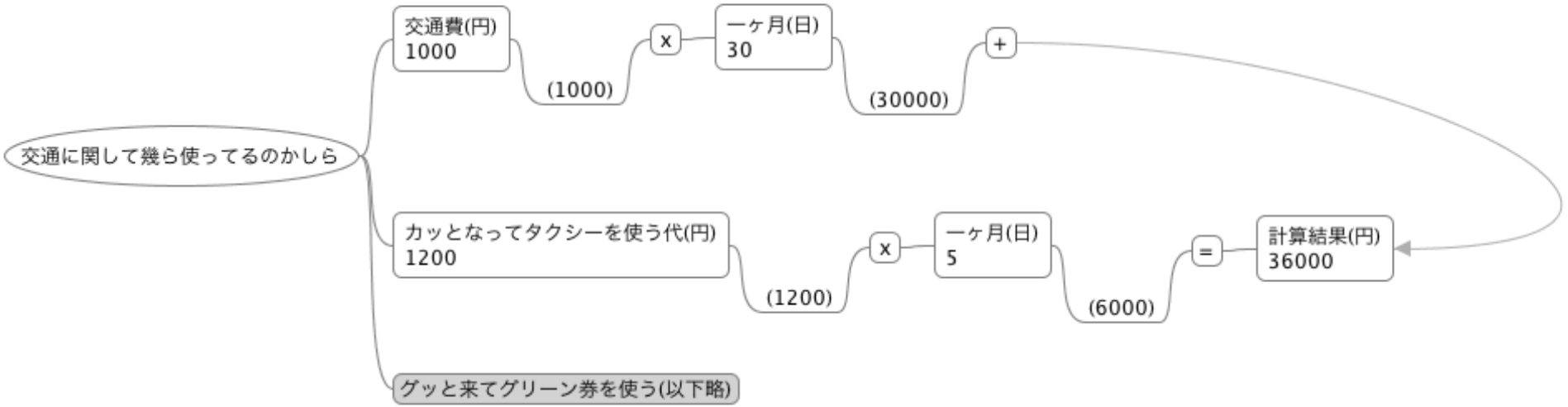
Drive37/44

2-10:ここからは、仕様とアイデアの話です

10.フィルタその3-2

判りやすいインターフェース

→ボタンとか数字、計算結果が見やすい
→こんな感じ
→たぶん拡大縮小とかが必要



Drive38/44

2-11:ここからは、仕様とアイデアの話です

11.フィルタその3-3

保存する機能

→とりあえずiPad内に保存する
→次回開いたときに表示される
→履歴機能はとりあえずパス。できるようには考えておこう。

共有する機能

→iPhoneだけではなくサーバとかが大量に必要なので、今回はパス！(苦い笑い)

決済する機能

→共有する機能の先かな。モノを買うサイトとかにつなげる。

末端の合計、17個！

Drive39/44

2-12:ここからは、仕様とアイデアの話です

12.あー疲れた、、、

とかね。アイデア作りからここまで、20分くらいです。図までつくってそんなもん。

慣れると速い、っつーかアイデア思いつくまでが長い。

あと、どこまで先が細かく読めるか、は、正直いってトライアンドエラーの世界です。

ここは、将来一般化(経験0からでもわかるように)したい。
さらにいうと、複数人でやったほうがいいです。

それに、この内容は進めていくにつれて、今後どんどん整理されていきます。

Drive40/44

2-13:ここからは、仕様とアイデアの話です

13.すごく雑なやり方ですが、これが仕様の作り方(みたいなもの)です。

疑問とかあるかな。正直ファジーなんだ、このへん。
で、いきなりですが、持ち込んできたアイデアについて、これ、やってみましょう。

フィルタ1~3まで、ガッと書く。最大45分使います。(休憩時間を兼ねます。)

我慢したい人はどうぞ我慢してください。何を。

フィルタその3まで終わったら or 3分経っても何も出てこない状態や詰まったら、
or フィルタその3の末端の項目数が合計で30 個を超えたら、井上まで。

最後20分で、出ているアイデアを井上のほうから発表します。つか確認ね。
この時間ですべてを書ききるなんて先ず無理なので、
やりたい事、やりたい部分から順に書いてしまえ！！ 30個書いたら持ってきて。

どうせ次回もあるんで、ブラッシュアップしていけるからね。違う物になってもOK。

Drive41/44

3. アイデアから仕様を作る(序)

ラスト20分で出来る事

Drive42/44

3-0:どーかなー。。

0.アイデアのフィルタリング、シューリョー！　　そしてその時仕様は、、、！

いままで書いたフィルタ１～３の、

それぞれ体裁を整えると、それが仕様だ!!というかんじ。
えって感じ？　でも、どうかな？なんとなく、

その３ができればその２が、
その２ができればその１ができるような気になってこないかしら。

まあ正直まやかしですが
何を作りたいのかはっきりさせておくのは
いい事です。

Drive43/44

3-1:ideaと戦った者たちへ

1.なんかCoccoの曲のタイトルみたい

先ず、系統分けしちゃいましょうかね。

予想としては、、

Webサービス連動型 : あのサービスがiPhoneから使いやすくなってわっしょい！

ゲーム型 : 遊んでわっしょい！

機能型 : 便利になってわっしょい！

この3パターンのどれか or 複合になってるんじゃないかな？

コレ以外のパターンがあったら是非超見てみたいが。私の経験にはまだないから。
おまけ情報ですが、フィルタその3については、各アイデアで、よっく見たら似てる、など
類似点が多い筈。　次回までに井上とかがまとめておくかな。

Drive44/44

3-2:アイデアと戦った人たちへ

2.土台をつくろうか

これで、全員がいる場所がスタートポイント。

チームわけェ、、、
とかはしません。

いきなりアプリを、、、
つくりません。

先ずは基礎を続けようか。でも、
作りたいものに繋がる基礎を。

Webブラウザとか、ミニゲームとか作ってみるべさ。

LookBack

ハイ、おつかれさまです。

- 0.籤、番号発表っつーか未だの人居てｽﾏｿ
- 1.iPhoneプログラムについて
- 2.最終回までに作りたいアイデアを答えよ(100,000点)(妙に震える大学 2010年 本試)
- 3.アイデアから仕様を作る(序)
- 4.コレ。

techhubteach@googlegroups.com

Continue?

次回予告:

次回は、

iPhoneでのアプリ製作をつらつら学びながら、
アイデアを形にしていきましょう。

次回までの間に、「ぷにぷにテ〇ーマン」アプリを学習サンプルとして送付します。

次回、

「アイデア出しとアプリ作りと」