

20161009

ゲームで学ぶオブジェクト指向！
初心者からRUBYを使って簡単に
マスターする3時間

座席について

写真撮影について

GOOD & NEW

- ①名前
- ②現在取り組んでいること
- ③将来やりたいと思っていること
- ④最近あった良い事・最近あった新しいこと

今日の流れ

～前提～
今日の目的



【Q】

なぜオブジェクト指向を
学ぶのか？

オブジェクト指向の考え方

- オブジェクト指向はほとんどの現場・アプリ開発で役立つ知識
- オブジェクト指向が分かれば効率よくアプリが作れる！
 - 拡張性の高いソース
 - 読みやすいソース
 - バグが出にくいソース

本日のゴール

- オブジェクト指向の基礎用語が分かる
- 自分でなんとなくオブジェクト指向の使い方が分かる



どうすればゴール
に近づけるか？

兎に角

トライ & エラー

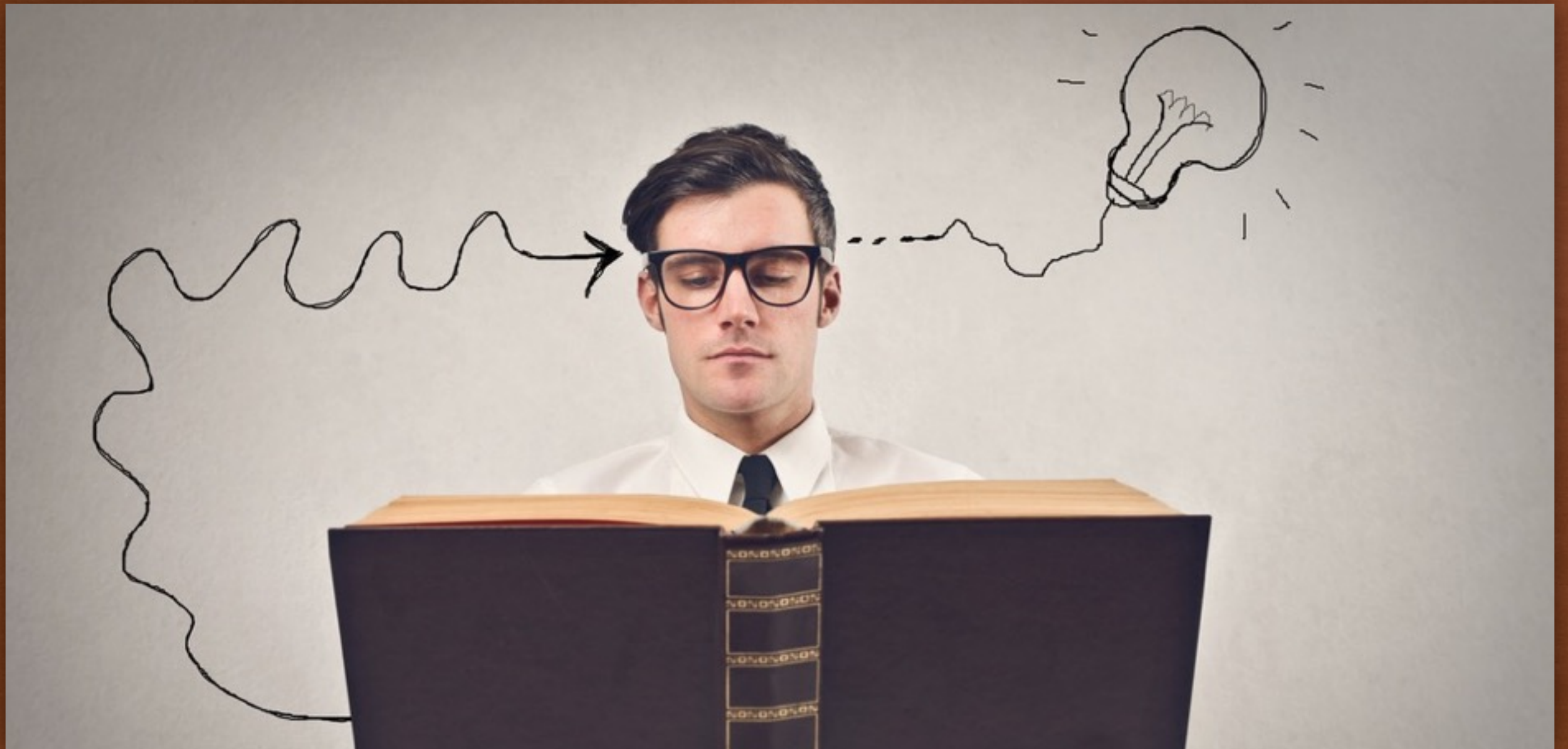
概念をインプット



自分で触ってみる！

～パート 1～

言葉・概念を理解しよう



オブジェクト指向

とは??

オブジェクト指向とは

- 特徴

- メジャーな言語で採用されているプログラミングの考え方
- ソースコードを「オブジェクト（物）」としてとらえて、プログラミングを進めていく
- しっかりと設計ができると再利用しやすいソースコードになる

- 難易度

- はじめはかなり難しいが、ある程度わかると便利に使える
- 特に設計は難しい。はじめは読みながら改造が出来れば十分！

オブジェクト指向の言語

- JavaScript
- Ruby
- PHP
- Python
- C++
- Java

「クラス」

「インスタンス」

「メソッド」

とは？

クラスとは

- 動作させる物体の設計図のようなもの
- これはどんなデータが出来るのか、これはどんな仕事をするのか、ということを決める



インスタンスとは

- 実働するオブジェクトのこと



メソッドとは

- 機能/処理のこと
- 言語によっては関数ともいう



クラス・インスタンス・メソッドの関係

- クラスを用意する
 - クラスにはそこから生まれるインスタンスが出来ること（メソッド）を書いておく
- クラスからインスタンスを作る
- インスタンスに対してメソッドの呼び出しを行う

【触ってみよう】 クラス・インスタンス・メソッド

- sample01_basic.rb
- ※Rubyの実行環境が無い方はCloud9等を御利用下さい。

```
1. sample01_basic.rb (~/Desktop/oop_ruby_sample) - VIM (Vim)
1 begin 【解説】
2 JackInTheBox (びっくり箱) : クラス
3 Open: メソッド
4 box : インスタンス
5 =end
6
7 class JackInTheBox          # びっくり箱クラスを定義 (仕様を宣言)、つまりこれが設計図となる
8   def open                  # びっくり箱はopenという振る舞いが可能という仕様にする
9     puts "べろべろばぁ～！" # びっくり箱に対してopenという動作を呼び出すとべろべろばぁ～！と言う
10  end
11 end
12
13 box = JackInTheBox.new      # びっくり箱の設計図
14 box.open
15
~
~
~
~
~
~
~
~
sample01_basic.rb 1,1 全て
```


コンストラクタ・デストラクタ

- コンストラクタ

- クラス（設計図）からインスタンス（実物）を生み出す時に自動的に動作する『特別なメソッド』
- Rubyでは initialize というメソッド名が割り当てられている
- （C++等ではClass名と同じ名前のメソッドにする等して宣言）
- 主に初期化（スタート状態に調整）する時に使う

- デストラクタ

- インスタンスがなくなった時に動作する処理
- （※Rubyにはないのでとりあえず知らなくて良い）

【触ってみよう】 コンストラクタ

- sample02_initialize.rb

```
1. sample02_initialize.rb (~/Desktop/oop_ruby_sample) - VIM (Vim)

1 =begin 【解説】
2 インスタンスをnewで生成することをインスタンス化という。
3 コンストラクタはインスタンス化するタイミングで実行される。
4 つまりインスタンスとして実物を作る際に始めに絶対に動作させることが出来る。
5 主にはそのインスタンスの「最低限必要な初期の情報」などを揃えるために使う。
6 =end
7
8 class JackInTheBox
9   def initialize                # Rubyのコンストラクタはinitializeという名前で決まっている
10     puts "プレゼントが届いたよ！" # インスタンス化されたタイミングで画面にメッセージを表示
11     @name = "びっくり箱くん"    # 名前を設定
12   end
13
14   def open
15     puts "べるべるばあ～！"
16     puts "ボクの名前は#{@name}だよ！びっくりした？"
17   end
18 end
19
20 box = JackInTheBox.new # newでインスタンス化したタイミングでinitializeが実行される
21 box.open
22
```

~

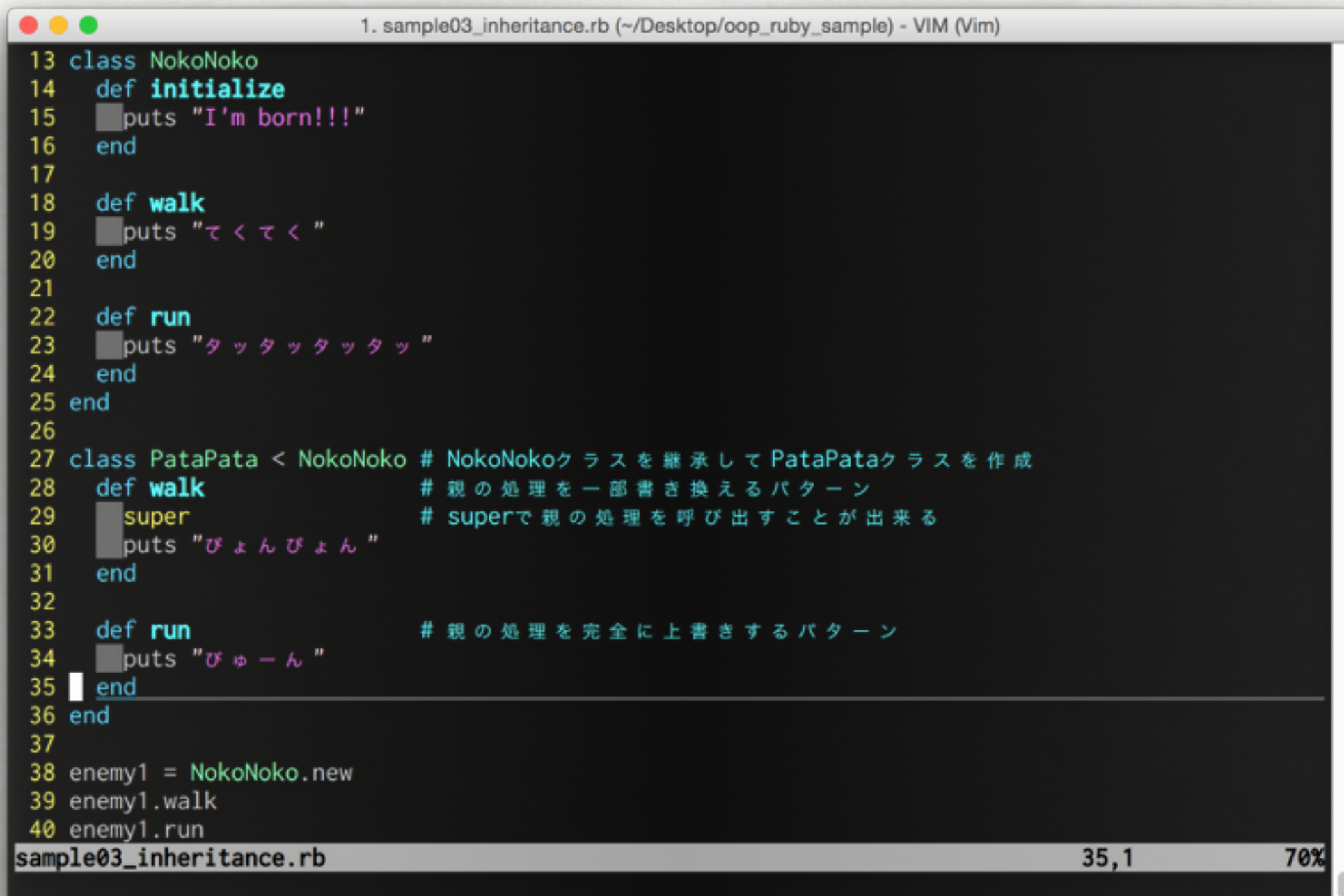
sample02_initialize.rb 22,0-1 全て

継承・オーバーライド

- 継承（インヘリタンス）
 - オブジェクト指向において重要な考え方のひとつ
 - 親の特徴を子供に引き継がせる行為のこと
- オーバーライド
 - 継承した処理を部分的に書き換えること
 - また、継承した処理を全部書き換えること

【触ってみよう】 継承・オーバーライド

- sample03_inheritance.rb



```
13 class NokoNoko
14   def initialize
15     puts "I'm born!!!"
16   end
17
18   def walk
19     puts "てく てく "
20   end
21
22   def run
23     puts "タッタッタッタ"
24   end
25 end
26
27 class PataPata < NokoNoko # NokoNokoクラスを継承してPataPataクラスを作成
28   def walk                # 親の処理を一部書き換えるパターン
29     super                 # superで親の処理を呼び出すことができる
30     puts "びょんびょん"
31   end
32
33   def run                 # 親の処理を完全に上書きするパターン
34     puts "びゅーん"
35   end
36 end
37
38 enemy1 = NokoNoko.new
39 enemy1.walk
40 enemy1.run
```

sample03_inheritance.rb 35,1 70%

カプセル化

- カプセル化
 - それぞれのインスタンスに固有に持つデータを、簡単に変更させない仕組み（考え方）のこと
 - 特定のメソッドを通じてデータにアクセスさせることによって、間違いを防いだり読みやすさを上げたりできる

【触ってみよう】カプセル化

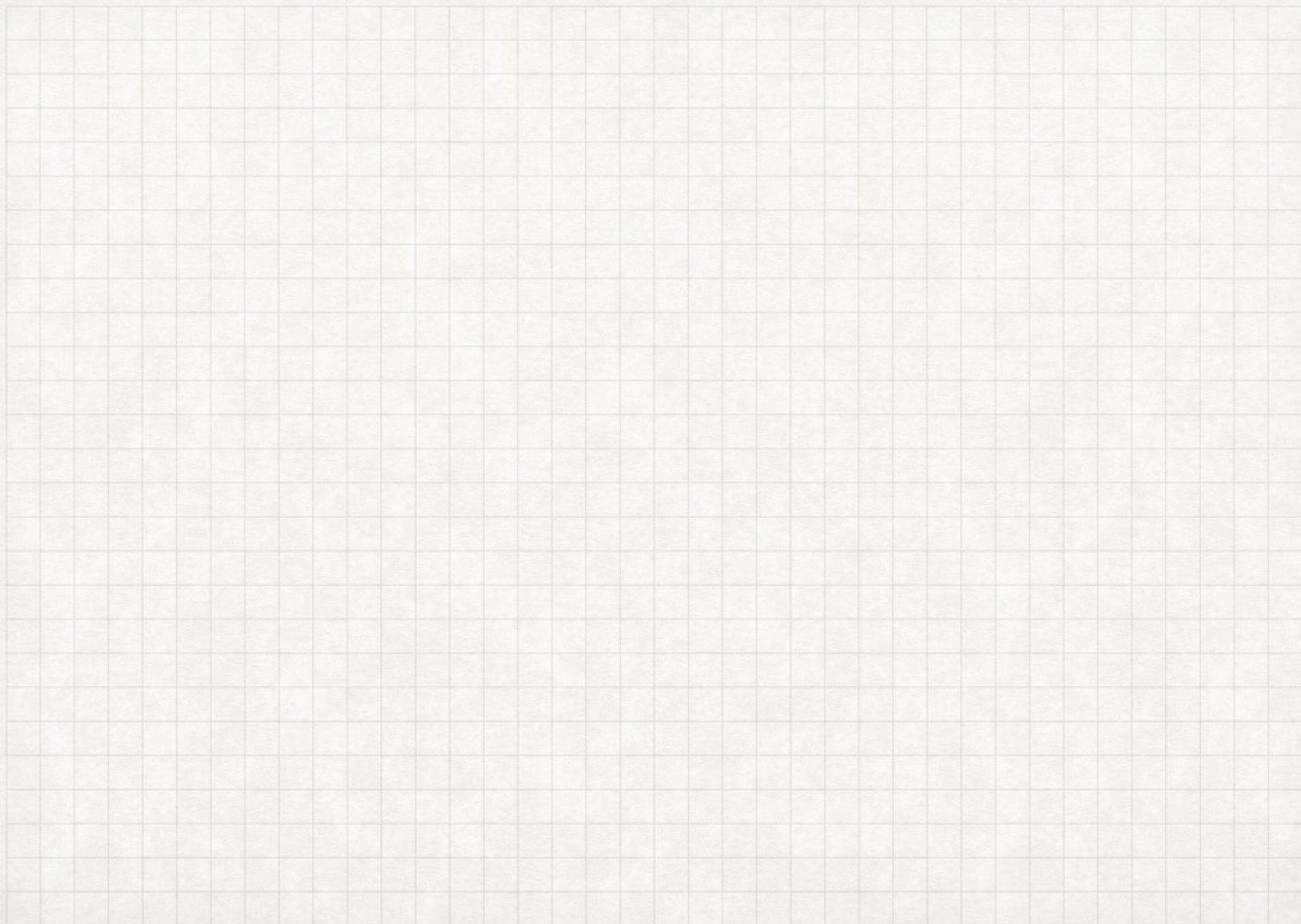
- sample04_encapsulation.rb

```
1. sample04_encapsulation.rb (~/Desktop/oop_ruby_sample) - VIM (Vim)

1 =begin 【解説】
2 簡単なRPG風プログラム
3 end
4
5 class Unit
6   def initialize
7     @hp = 10
8   end
9
10  def attack
11    puts "#{@name}の攻撃！"
12    rand(1..3)
13  end
14
15  def damage(attack)
16    @hp = @hp - attack
17    puts "#{@name}は#{attack}のダメージを受けた！"
18    if @hp < 0
19      death_scream
20    end
21  end
22
23  def death?
24    @hp < 0
25  end
26
27  def death_scream
28    puts "#{@name}「やーれーたー」"
29  end
30 end
```

sample04_encapsulation.rb 3,1 先頭

"sample04_encapsulation.rb" 66L, 937C 書込み



～パート2～
実際に触ってみよう！



RPG GAME MISSION

- 【初級】 ゲーム開始時に勇者の名前を入力してもらい、任意に変えられるようにしましょう。
- 【初級】 やられた時の叫び超えを勇者と敵でそれぞれ別の、オリジナルのものに変更して下さい。
- 【初級】 通常攻撃時に確立でダメージが増加する「クリティカルヒット」を実装して下さい。
- 【初級】 攻撃毎に2秒処理を止めて見ましょう、また攻撃時に攻撃を表現するアスキーアート（絵文字）を入れてみましょう。
- 【中級】 コマンド入力で「通常攻撃」か、「必殺技」かを選べるようにして下さい。
- 【中級】 HPや攻撃力などのパラメータの概念を取り入れる。
- 【中級】 対戦できる敵を選べるようにする。
- 【上級】 ドラクエ風にパーティで複数人对複数人对戦できるようにしましょう。

OTHER MISSION

- ゲーム
 - クイズゲームを作ってください。
 - じゃんけんゲームを作ってください。
 - すごろくを作ってください。
- ツール
 - クローラーを作ってみてください。

～パート3～
発表しよう！



良いプログラムとは？

- 「凝集度」が高いこと
 - 関係性の高い処理のみを集める
 - 例：RPGにおいてゲームマスターというクラスがターンを進めるのは分かるが、Enemyがターンを進めるなどの処理を持っていると意味不明
- 「結合度」が低いこと
 - メソッドやクラスがお互いに出来るだけ依存しないようにする
 - （別のプログラムに切り出しても使える状態が良い）

アンケートのお願い

集合写真 撮りましょう！

顔出しNGの方は隠して頂けると助かります。

次回以降のイベント予定

- 10/15（土） 14:15~17:00
 - 侍卒業生に学ぶ！プログラミング学習のコツとリアルな体験【懇親会付】
- 10/30（日） 14:15~17:00
 - ポケモンGOの作り方！簡単AR実装で最新技術を体験しよう
- 11/6（日） 14:15~17:00
 - 実践プチ起業！ゼロから誰でも1ヶ月で出来るサービスの始め方
- 11/12（日） 19:00~21:00
 - 【東京限定】プログラミングの難しさを楽しく飲んで愚痴る会
- 11/20（日） 14:15~17:00
 - これからの時代に置いていかれない為のAWS超入門