オブジェクト指向

プログラミングの勉強、何をしたらいいか

— チュートリアルをやってみる(ネットにごろごろ転がっています)
— とにかく作りたいものを作ってみる
— **わからなかったらその都度調べる(すごく重要)**
— 他の人と手を取り合ってやる(習得スピードが段違い)
— とにかく書いてみる(やってみるのとやらないのじゃ大分違う)

とりあえず基本を抑えてしまえば、あとは超特急で理解できるようになります。

プログラミング(特に開発寄りなところ)で大事な事

— **読みやすいコード(コメントや命名規則やインデントをしっかり)**
— よく考えてから書く(プログラミングには紙と鉛筆も大切です)
— こまめなデバグ(println と友達に)
— タスク管理(だらだらやらないように)

オブジェクト指向とは何か

— プログラミングの形式の 1 つ (主流な言語はだいたいこれ)
— 全部を「オブジェクト(もの)」としてみてしまう(物でも概念でもなんでも)
— 「もの」だから人が考えやすい

キーワード：クラス、クラス変数、クラス関数、コンストラクタ、インスタンス、プリミティブ/データ型(Java では重要)、get / set、シャロー/ディープコピー、(継承、修飾子、親クラス、小クラス、多態性、デザインパターン、リファクタリング、etc...)

考えてみよう！

— 練習が大切(関数型からオブジェクト指向的にシフト)
— 紙と鉛筆使って考えてみよう！

基本。(実際に例を考えてみよう)
— パターンを見出してどんな「もの」があるか考えよう！ (クラス)
— 「もの」が持っている要素を考えよう！ (クラス変数)
— 「もの」ができることを考えよう！ (クラス関数)
— 「もの」の初期状態を考えよう！ (コンストラクタ)
— 「もの」ができないことは他の「もの」にやらせよう！ (詰め込みすぎない)
— 「もの」と「もの」のつながりを考えよう！
— 「もの」が他の「もの」の中でどう使われるか考えよう！(インスタンス)

発展。(基本ができるようになってから考えよう)
— いくつかの「もの」に共通項があったら抽象化した「もの」にしよう！(継承)
— 抽象的な「もの」は子「もの」を取りまとめられる！(親/子クラス、多態性)
— 「もの」の情報の公開範囲を考えよう！(アクセス修飾子)
— データには種類があることを理解しよう！(Primitive / Data 型、コピー)

超発展。(クラスなんて余裕だぜ！ってなったら)
— MVC (Model View Controller)：アプリケーションには最重要な概念
— デザインパターン：ソフトウェア開発によくでるパターンの集合
— interface : class の仕様書のような「もの」
— リファクタリング(コードの最適化や綺麗にしていく)
— UML diagram (クラスの繋がりの可視化(僕はやっていません))

有用なさいと
— JavaDrive(processing ならおすすめ。)
— Document (ようするに辞書)
— あとはやりたいことでぐぐれば大抵はでてきます。