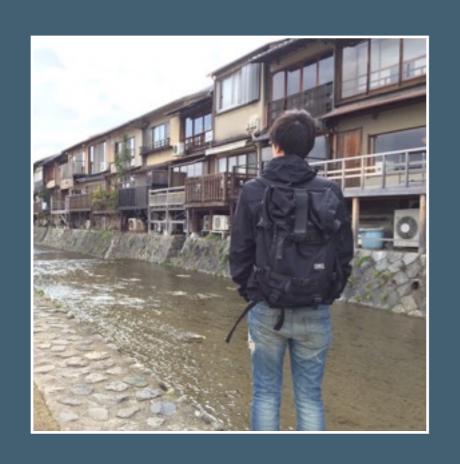
メルカリiOSインターンでの 幾多なる失敗で学んだリーダブル コードの大切さ!読みやすい コードとはこういうものだ**(**)

yuzushioh

自己紹介



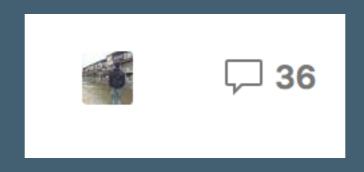
- · 福田 涼介 (yuzushioh)
- 2016年1月にソウゾウに入社 (インターン)
- ・19歳の大学2年生

ソウゾウが人生初めてのチーム開発

初めてのコードレビュー・・

ワクワクい

たくさんの修正コメント電



そしてそこから学びがあった。

PRのコメントの中でも特に多かったものが...

▲: 命名が適切でないようです

4:メソッド名と引数があっていません

↑:メソッド名からこのメソッドがなにを しているのかはっきりわかりません

▲: 読む人の負担を下げるようにしてください

ほとんどがリーダブルコードに関する事



リーダブルコードって??

: 読んだ時に最短で理解出来る

●:パターンに一貫性がある

●:メソッド内も読みやすい

コードがあるべき理想像学

命名を見て誰が想像しても同じ処理になる

メソッドを追わなくても実装が理解出来る

探した際に見つけやすい

●: 読んだ時に最短で理解出来る

●:パターンに一貫性がある

●:メソッド内も読みやすい

密命名は超重要

本当にあった悪い例

func createActivityIndicator(view: UIView)

メソッド名から内容が理解できない

メソッド名と引数が合っていない

命名が悪いと読んだ際に中身を確認しないといけないで

メソッドが、何を、何してる、かわかるようにする。

func addActivityIndicatorOnView(view: UIView)

このメソッドがIndicatorViewを viewに追加していることがわかる かといって、命名が冗長になるのは良くないの

恵 思い例

```
struct InAppNotification {
    let contentType: InAppNotificationContentType
    enum InAppNotificationContentType {
        case DefaultType(InAppDefaultNotification)
        case MessageType(InAppMessageNotification)
        case Unknown
    }
}
```

```
struct InAppNotification {
    let contentType: ContentType
    enum ContentType {
        case DefaultType(DefaultContent)
        case MessageType(MessageContent)
        case Unknown
    }
}
```

冗長じゃなくてすっきり

内容が理解出来る最短の命名をし

PRの前に本当にその命名が正しいのか 再度確認してみてください。

●: 読んだ時に最短で理解出来る

●:パターンに一貫性がある

●:メソッド内も読みやすい

メソッドは同類でまとめて定義すべき。

```
class TimelineViewController: UIViewController {
    func pushToProfileViewContoller(user: User) {}
    func pushToMapViewController(item: Item) {}
    func showAddressToastView() {}
    func hideAddressToastView() {}
    func addBalloonViewOnView(view: UIView) {}
    func addIndicatorViewOnView(view: UIView) {}
    func addTableHeaderView() {}
extension TimelineViewController: UITableViewDataSource {
    func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return 0
    func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {
        return UITableViewCell()
extension TimelineViewController: UITableViewDelegate {...}
extension TimelineViewController: UIScrollViewDelegate {...}
```

メリット

メソッドを探す時にどこを見るかがすぐに分かる。

プロジェクト内が整理される。

似ているメソッド名は命名も規則的にし

```
// 画面をPushするメソッド
func pushToSearchViewController()
func pushToProfileViewController()
// 画面をModal表示するメソッド
func presentCategorySelectViewController()
func presentPostViewController()
// 画面にViewを追加するメソッド
func addBalloonViewOnView(view: UIView)
func addChildViewControllerToSelf()
// なにかを表示するメソッド
func showSearchFilterView()
func hideSearchFilterView()
// 状態を管理するメソッド
func setItemEditable(editable: Bool)
func setButtonsHidden(hidden: Bool)
```

メリット

命名に規則性を持たせる事で読み手が 理解しやすくなる 👍

●: 読んだ時に最短で理解出来る

●:パターンに一貫性がある

●:メソッド内も読みやすい

ネストを極力深くしない。

本当にあった 悪い例

```
func updateItemInformationWithItem(item: Item) {
    if item.isDeleted {
        if item.isMyItem {
            actionButton.setTitle("編集", forState: .Normal)
        } else {
            actionButton.setTitle("購入", forState: .Normal)
            if item.isClosed {
                commentButton.enabled = false
           if item.isLiked {
               likeButton.setImage(UIImage(named: "is_liked_icon"), forState: .Normal)
                likeCountLabel.text = "\(item.likeCount)"
            } else {
                likeButton.setImage(UIImage(named: "not_liked_icon"), forState: .Normal)
                likeCountLabel.hidden = true
    } else {
       addDeletedIconView()
}
```

不要なケースの場合、関数から早く返すし

+

三項演算子でシュッと書くん

ネストを削除しシュッと書く

```
func updateItemInformationWithItem(item: Item) {
    guard !item.isDeleted else {
        addDeletedIconView()
        return
    }

    commentButton.enabled = item.isClosed
    actionButton.setTitle(item.isMyItem ? "編集" : "購入", forState: .Normal)

let iconName = item.isLiked ? "is_liked_icon" : "not_liked_icon"
    likeButton.setImage(UIImage(named: iconName), forState: .Normal)
    ilikeCountLabel, text = "\(item.likeCount)"
    likeCountLabel, hidden = !item.isLiked
}
```

メリット

早期に返すことで条件を減らせる。

三項演算子で実行されることがわかる。

まとめ

これだけは覚えて帰ってください。

yuzushiohですへ



命名は、何を、何してる、かわかるようにする。

内容が理解出来る最短の命名をし

命名を規則的に

ネストを極力深くしない。

ありがとうございました

