ClearCode

コードチェンジ

須藤功平

株式会社クリアコード

SEゼミ2015 - リーダブルコード勉強会

2015-06-06

目的の確認

参加者の
リーダブルコード力の
レベルアップ

やろうとしていること

1. コードを読んで
2. リーダブルコードを**見つける**
3. ↑を活かしてコードを書く

↑

OSSの開発では当たり前のこと

現状

1. 全員共通の課題を用意←Done
2. 課題を実装←Done
3. 実装を交換←**これから**
4. 交換した実装で開発継続

どうして交換するのか

- ✓ コードを読まざるを得なくなる
  - ✓ コードを読まずに開発継続できない
- ✓ 自分も実装した仕様
  - ✓ 仕様は理解済み→読む敷居は低い
  - ✓ 違う視点での実装を読むことになる
  - ✓ 新しい発見があるはず

やり方

- ✓ 交換相手を決める
- ✓ 交換相手のリポジトリーをfork
- ✓ ↑を使って開発継続

交換相手の決め方

- 基準
  - だいたい同じ状況同士
  - 例：進み具合、環境
- メンターがヒアリング
  - 積極的に協力して

リポジトリーをfork

- ✓ やり方がわからない人は挙手

メンターへ：forkできているか確認してあげて

開発継続

- どこまで進んでいるかを確認
  - README や log を参考に
    （どういう風に書いておけばよかったか考えてみて）
- ↑から開発を継続

忘れないで

- リーダブルコードを発見→メモ
  - memo.mdに追記して随時push
  - 後で共有する時に使う
- 書くこと
  - 実際のコードまたはコードのURL
  - リーダブルな理由
  - 見つけたきっかけ

ポイント

悪いコードより
リーダブルな
コード

悪いコード

- 見つけやすい
  - 異質
  - リーダブルじゃない
- 過剰に指摘したくなる人がいる
  - 指摘するならリーダブルなコードを書いてPull Request

    コミットメッセージによくなる理由を書く

    例: https://github.com/BLThunder1991/BLThunder1991-sezemi-2014-readable-code-2/pulls?q=is%3Aclosed

リーダブルなコード

- 見つけにくい
  - リーダブルだから
  - すーっと理解できてひっかからない
- 今日のチャレンジ
  - 意識して見つけよう！

発見方法のヒント

- リーダブルコード
  - **読む人**が
    読みやすいなら
    リーダブル
- 読む人視点が重要
  - 交換直後の今が読む人の視点！

忘れないで

- ✓ リーダブルコードを発見→メモ
  - ✓ memo.mdに追記して随時push
  - ✓ 後で共有する時に使う
- ✓ 書くこと（例はtask.mdに書いてある）
  - ✓ コード・理由・見つけたきっかけ

メンターへ：メビューで使うのでメンターもメモろう

メンターへ：たまに「どんなメモ書いた？」と聞いて課題をこなすことではなくリーダブルコードを書くことに参加者の注意を向けてみよう