ClearCode

課題の実装の進め方

結城洋志

株式会社クリアコード

実践リーダブルコード

2022-11-02

実装の目的

- ✓ 課題の仕様を理解する
- ✓ 「読まれることを意識して書かれたコード」を用意する
  - ✓ **リーダブルなコード**を書こう！

（後半戦への布石）

目的じゃないこと

- ✓ テクニックをたくさん覚える
- ✓ 難しいプログラムを実装する
- ✓ プログラムを速く実装する
- ✓ 高性能なプログラムを実装する
- ✓ 奇抜な方法で目立つ

課題の傾向

- ✓ 技術的に難しいことは意図的に避けている
- ✓ 段階的に改良していく
- ✓ 時間内で実装しきれない分量
  - ✓ **すべて実装する必要はない！**

メモを書く

- 工夫したことはissueに書く
  - 1つのissueに1つの工夫でOK
- メモに含めること
  - 実際のコードのURLか
    コード片（差分）
  - リーダブルな理由

メモ例

タイトル： 統一されたスタイル ← リーダブルな書き方の名前

↓本文↓
--
URL: https://github.com/kou/.../commit/eb02be

コード:
```
@@ -64,7 +64,7 @@ void add_term_to_TermList(...) {
 void open_TermList(TermList *list, char *path) {
   FILE *fp;
   /* ファイルを開く */
-  if ((fp = fopen(path , "r")) == NULL){
+  if ((fp = fopen(path , "r")) == NULL) {
     fprintf (stderr, "ファイルが開けません¥n");
     fclose(fp);
     exit(EXIT_FAILURE);
```

理由:
ファイル内でスタイルを統一しているので読みやすくなっている。

リーダブルコードとは

- 読む人基準
  - 「読む人」が読みやすいならリーダブル
- 読む人の視点を意識してみて
  - どんな人が読む？
  - どんな前提知識がある？

リーダブルにするヒント

- 要件が増えて、それまでと**前提が変わった**とき
  - 新しい要件に
    惰性ややっつけで対応すると
    リーダブルでなくなりやすい

リーダブルにするヒント

- ✓ × とりあえず
  こう書き足したら動く
- ✓ ○ 新しい要件に基づいて
  コードを見直し、
  適切な書き方に改める
- ✓ 「どうだったらリーダブルか」
  をその都度考え直してみよう

実装に困ったら1

- 講師に相談
  - 講師は**答えを教えない**
  - 一緒に考えてくれる
- 答えを教えない理由
  - 参加者が考える機会を奪わないため
  - 一緒に考えると新しい視点が増える

実装に困ったら2

- 他の人に相談
  - 答えを教えてもよい
  - 一緒に考えてもよい
- 条件
  - 答えを教えるときは理由も伝える
    （教える参加者が考える機会になる！）
  - 聞く人は理由も聞く
    （考え方を知ると新しい視点でコードを読める！）

実装開始

- 仕様
  - task.md
- 練習
  - 講師がtask.mdを読んで説明
  - 仕様1を各自で実装
  - 実装できたらチューターに声掛け
    →仕様2以降実装

ここまでの説明

腑に落ちましたか？

開始

12:15まで