

リーダブルコードの意義 と、実践の方法

結城洋志
株式会社クリアコード
リーダブルコード演習
2021-09-13

全体の流れ

- ✓ 1日目（本日）
 - ✓ 前半：概要と進め方の説明
 - ✓ 後半：チームで実装
- ✓ 2日目（明日）
 - ✓ 前半：チーム同士で実装を交換、
続きを実装
 - ✓ 後半：全体で結果を共有

講師紹介

結城洋志 (ゆうき ひろし)
aka Piro

- ✓ 株式会社クリアコード所属
- ✓ FirefoxやThunderbirdの法人サポートに従事
- ✓ トラブルの原因や対策を探るためソースコードを調査することが多い

チューター紹介

足永拓郎
(あしえ たくろう)

- ✓ 株式会社クリアコード所属
- ✓ Fluentd開発メンバー
- ✓ 参加者のサポート係（オンライン）

チューター紹介

大場光一郎
(おおば こういちろう)

- ✓ 株式会社Speee所属
- ✓ JRubyコミッター
- ✓ 参加者のサポート係

アジェンダ

- ✓ 講座の目的を確認
- ✓ リーダブルコードの
必要性を確認
- ✓ リーダブルコードの
実践方法を紹介
- ✓ 実践方法を練習 (次のコマから)

講座の目的

- ✓ リーダブルコードを日常的に書く上での基礎となる考え方を実践し、持ち帰る

目的でないこと

- ✓ 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

そもそもの話

- ✓ リーダブルコードはなぜ必要か
- ✓ 何の役に立つのか？

リーダブルコードが必要な理由

- ✓ 既存のコードを読んで素早く内容を把握したい
- ✓ 既存のコードに素早く手を加えたい
- ✓ 開発速度を落としたくない

「速度」？

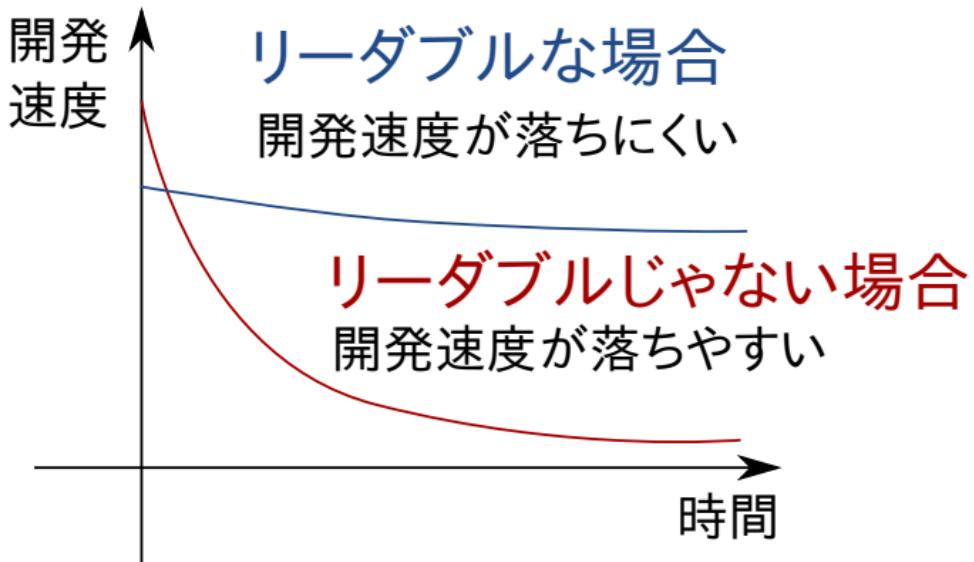
- ✓ 「読みやすさに気をつけたら
開発スピードが
落ちそう……」

むしろ「読みにくくと開発が遅くなる」

- ✓ 既存のコードを理解しにくくと……
- ✓ 修正・機能追加に時間がかかる
(理解しないと変更できない)
- ✓ 後退バグが発生しやすい
(理解しないまま変更すると問題発生)

→コストがかかる

時間が経つほど影響大



(注意：グラフではなく概念図です)

つまり

- ✓ 現実的なコストの範囲で
 - ✓ 既存のコードを継続的に、無理なく改良・修正したい
- なので、リードブルコード

既存のコードを読んだり手を 加えたりする場面

在学中だとどんな場面がありそう？

既存のコードに手を加える場面

- ✓ 複数人で研究
 - ✓ 共同研究、院生と学部生で分担
- ✓ 自分の研究を発展
 - ✓ 修士で作った物を博士で使う
 - ✓ 途中で方針修正・転換
- ✓ 先輩の研究を参照

既存のコードを読む場面

- ✓ 論文に含まれるコード
を読むとき
- ✓ 自分が誰かの論文を読むとき
- ✓ 自分の論文が誰かに読まれるとき

コード"がリーダブルだと参照されやすくなるかも……

営利企業ではどうか

- ✓ 分かる人が1人しかいない→危険
- ✓ 実装者が抜けたら詰む
- ✓ 変更できてもものすごくコストがかかる

↓
チームで開発し、
チームの誰でも作業を引き継げる
状態にしておきたい

リーダブルコードの実践

どうすれば無理なく実践できる？

リーダブルコードの実践

コードを読む
習慣を作る

読む？書くじゃないの？

- ✓ リーダブルコードを書くには
コードを読むことが欠かせない
- ✓ なぜ？

↓
書いている最中は
読みやすさ・読みにくさ
を実感しにくいいから

実際のコードで考えてみる

- ✓ Excel|ワークシートの生成で見出しセルを結合する関数

A	B	C	D	E	F	G	H	I
カタゴリー	項目設定番号	カスタマイズ項目(既定)	選択肢番号	選択肢	設定内容の基準 (ESR91)	変更した設定値 (ESR91)	ESR70→ESR91 での変更	確認手帳書付 応答用
Install	1 インストールの表示名	1 Fa Meta Installer (既定)	-	-	-	-	-	-
3	2 位置の表示	2 [空欄] (既定)	define PRODUCT_FULL_NAME " (名称)"	define PRODUCT_FULL_NAME "Fa Meta Installer"	-	-	-	-
4	3 インストーラーのファイル名	3 [FaMetaInstaller]	-	-	-	-	-	-
5	4 位置の表示	4 [空欄] (既定)	define PRODUCT_NAME " (名称)"	define PRODUCT_NAME "FaInstaller"	-	-	-	-
6	3 インストーラー動作モード	5 ワイザードを表示し、操作場所を選択しない (既定)	define PRODUCT_INSTALL_MODE "NORMAL"	define PRODUCT_INSTALL_MODE "NORMAL"	-	-	-	-
7	6 ワイザードを表示し、操作場所を選択しない (既定)	6 ワイザードを表示し、操作場所を選択しない (既定)	define PRODUCT_INSTALL_MODE "PASSIVE"	define PRODUCT_INSTALL_MODE "PASSIVE"	-	-	-	-
8	7 ワイザードを表示し、操作場所を選択しない (既定)	7 ワイザードを表示し、操作場所を選択しない (既定)	define PRODUCT_INSTALL_MODE "QUIET"	define PRODUCT_INSTALL_MODE "QUIET"	-	-	-	-
9	4 インストール完了後のメッセージ	8 表示しない (既定)	InstallInstall;	-	-	-	-	-
10	9 表示する	9 表示する	InstallInstall;	[InstallInstall];	InstallInstall;	InstallInstall;	-	-
11	5 インストール完了後の再起動選択	1 再起動しない (既定)	InstallInstall;	[InstallInstall];	InstallInstall;	InstallInstall;	-	-
12	2 表示する	2 表示する	InstallInstall;	[InstallInstall];	InstallInstall;	InstallInstall;	-	-
13	3 ConfirmRestartAfterInstallFa Meta Installer	3 ConfirmRestartAfterInstallFa Meta Installer	-	-				
14	4 ConfirmRestartAfterInstallFa Meta Installer	4 ConfirmRestartAfterInstallFa Meta Installer	-	-				
15	7 Firefox/Thunderbirdの各種	1 日本語 (既定)	define PRODUCT_LANGUAGE "Japanese"	define PRODUCT_LANGUAGE "Japanese"	-	-	-	-
16	2 英語	2 英語	define PRODUCT_LANGUAGE "English"	define PRODUCT_LANGUAGE "English"	-	-	-	-
17	8 Firefox/Thunderbirdのインストール先	1 固定のインストール先 (既定)	FirefoxSetup.inif /InstallDir=	FirefoxSetup.inif /InstallDir=	FirefoxSetup.inif /InstallDir=	FirefoxSetup.inif /InstallDir=	-	-
18	2 变更のインストール先	2 FirefoxSetup.inif /InstallDir=%userprofile% Firefox	-	-				
19	3 メインストーラー表示バージョン	1 未登録しない (既定)	InstallInstall;	-	InstallInstall;	InstallInstall;	-	-
20	4 表示する	4 表示する	InstallInstall;	[InstallInstall];	InstallInstall;	InstallInstall;	-	-
21	5 FirefoxSetup.inifバージョン番号	5 FirefoxSetup.inifバージョン番号	DisplayVersionName=FirefoxSetup	DisplayVersionName=FirefoxSetup	DisplayVersionName=FirefoxSetup	DisplayVersionName=FirefoxSetup	-	-
22	10 スタートメニュー、タスクバー上のショートカット	1 未登録しない (既定)	-	-	-	-	-	-

やりたいことを言語化してみる

関数 項目の見出し列 (B~C) のセルを結合する(行番号, 項目):

項目の選択肢の数が1以下だったら:

何もせず終了

そうでないなら (選択肢が2つ以上あるなら)、

「見出し列の定義の配列」の全要素について、要素の番号 を使って:

シートの範囲を結合(

開始行 → 行番号,

開始列 → 要素の番号,

終了行 → 行番号 + 選択肢の数 - 1,

終了列 → 要素の番号)

項目の見出し列の……

A	B	C	D	E	F	G	H	I	
	項目番号	カスタマイズ項目(目的)		選択肢 番号	選択状態	設定内容の範囲 (E0801)	反映した設定値 (E0801)	E1076→E1081 での実装	補足手順付 応答号
1	Install	1 インストーラの表示名		1 Fx Meta Installer (既定)	-	#define PRODUCT_FULL_NAME "（名前）"	#define PRODUCT_FULL_NAME "Demo Fx Installer"		
2		2 インストーラのファイル名		1 FxMetaInstaller	-	#define PRODUCT_NAME "（名前）"	#define PRODUCT_NAME "DemoInstaller"		
3		3 インストーラの動作モード		1 ウィザードを表示し、操作場 所を選択する	#define PRODUCT_INSTALL_MODE "KERNEL"	#define PRODUCT_INSTALL_MODE "PASSIVE"	#define PRODUCT_INSTALL_MODE "QUIET"		
4		4 インストール完了後のメッセージ		2 ウィザードを表示し、操作場 所を選択しない	#define PRODUCT_INSTALL_MODE "QUIET"				
5		5 インストール完了後の再起動要求		1 表示しない (既定)	#ifndef _WIN32	[fxinstall.h]			
6				2 表示する	fxinstall.h	FinishTitleFxMetaInstaller	FinishMessage(インストールが完了しました)		
7				3 表示しない (既定)	-				
8				4 表示不可		WinRestart.h			
9									
10									
11									

関数 項目の見出し列 (B~C) のセルを結合する(行番号, 項目):
項目の選択肢の数が1以下だったら:
 何もせず終了
 そうでないなら (選択肢が2つ以上あるなら)、
 「見出し列の定義の配列」の全要素について、要素の番号 を使って:
 シートの範囲を結合(
 開始行 → 行番号,
 開始列 → 要素の番号,
 終了行 → 行番号 + 選択肢の数 - 1,
 終了列 → 要素の番号)

セルを結合する

A	B	C	D	E	F	G	H	I
	項目番号	カスタマイズ項目（目的）	選択肢番号	選択肢	設定内容の範囲 (E801)	反映した設定値 (E801)	E1070→E1091 での実装	確認手順書付 記番号
1	1	インストーラの表示名	1	1 Fx Meta Installer (既定)	-	-	-	-
2	2	インストーラの表示名	2	2 任意の名前	#define PRODUCT_FULL_NAME "（名前）"	#define PRODUCT_FULL_NAME "Demo Fx Installer"	-	-
3	3	インストーラのファイル名	1	1 FxMetaInstaller	-	-	-	-
4	4	インストーラの表示名	2	2 任意の名前	#define PRODUCT_NAME "（名前）"	#define PRODUCT_NAME "DemoInstaller"	-	-
5	5	インストーラの動作モード	1	1 ウィザードを表示し、操作場所を尋ねる	#define PRODUCT_INSTALL_MODE "KERNEL"	-	-	-
6	6	インストーラの動作モード	2	2 ウィザードを表示し、操作場所は選択しない（既定）	#define PRODUCT_INSTALL_MODE "PASSIVE"	#define PRODUCT_INSTALL_MODE "PASSIVE"	-	-
7	7	インストーラの動作モード	3	3 ウィザードを表示しない	#define PRODUCT_INSTALL_MODE "QUIET"	-	-	-
8	8	インストーラの表示名	1	1 表示しない（既定）	#ifndef _WIN32 #include <iostream.h>	-	-	-
9	9	インストーラの表示名	2	2 表示する	#ifndef _WIN32 #include <iostream.h> #endif #include <wx/string.h> #include <wx/meta_installer.h>	-	-	-
10	10	インストール完了後のメッセージ	1	1 表示しない（既定）	-	-	-	-
11	11	インストール完了後のメッセージ	2	2 表示する	#ifndef _WIN32 #include <iostream.h> #endif #include <wx/string.h> #include <wx/meta_installer.h> #include <wx/msgdlg.h>	Message:インストールが完了しました。	-	-
12	12	関数 項目の見出し列 (B~C) のセルを結合する(行番号, 項目):						
13	13	項目の選択肢の数が1以下だったら: 何もせず終了						
14	14	そうでないなら (選択肢が2つ以上あるなら)、 「見出し列の定義の配列」の全要素について、 要素の番号 を使って: シートの範囲を結合(
15	15	開始行 → 行番号, 開始列 → 要素の番号, 終了行 → 行番号 + 選択肢の数 - 1, 終了列 → 要素の番号)						
16	16	スタートメニュー、タスクバー上のショートカット						
17	17							
18	18							
19	19							
20	20							
21	21							

見出し列の定義の配列

	A	日 項目設定番号	○ カスタマイズ項目（目的）	D	E 選択肢 番号	F 状態表	G 設定内容の範囲 (E5801)	H 反映した設定値 (E5802)	I E1076→E1081 での実装 候補
1	カタゴリー								
2	Install		1 インストーラの表示名		1 Fx Meta			Installer"	
3			2 インストーラのファイル名		2 任意の名前			"ex"	
4			3 インストーラの動作モード		3 カタログ 生成モード			"M"	
5			4 インストール完了後のメッセージ		4 フォルダ 作成モード				
6			5 インストール完了後の再起動要求		5 表示しない				
7					6 表示する				
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									
22									
23									
24									
25									
26									
27									
28									
29									
30									
31									
32									
33									
34									
35									
36									
37									
38									
39									
40									
41									
42									
43									
44									
45									
46									
47									
48									
49									
50									
51									
52									
53									
54									
55									
56									
57									
58									
59									
60									
61									
62									
63									
64									
65									
66									
67									
68									
69									
70									
71									
72									
73									
74									
75									
76									
77									
78									
79									
80									
81									
82									
83									
84									
85									
86									
87									
88									
89									
90									
91									
92									
93									
94									
95									
96									
97									
98									
99									
100									
101									
102									
103									
104									
105									
106									
107									
108									
109									
110									
111									
112									
113									
114									
115									
116									
117									
118									
119									
120									
121									
122									
123									
124									
125									
126									
127									
128									
129									
130									
131									
132									
133									
134									
135									
136									
137									
138									
139									
140									
141									
142									
143									
144									
145									
146									
147									
148									
149									
150									
151									
152									
153									
154									
155									
156									
157									
158									
159									
160									
161									
162									
163									
164									
165									
166									
167									
168									
169									
170									
171									
172									
173									
174									
175									
176									
177									
178									
179									
180									
181									
182									
183									
184									
185									
186									
187									
188									
189									
190									
191									
192									
193									
194									
195									
196									
197									
198									
199									
200									
201									
202									
203									
204									
205									
206									
207									
208									
209									
210									
211									
212									
213									
214									
215									
216									
217									
218									
219									
220									
221									
222									
223									
224									
225									
226									
227									
228									
229									
230									
231									
232									
233									
234									
235									
236									
237									
238									
239									
240									
241									
242									
243									
244									
245									
246					</				

シートの範囲を指定して結合

A	日	E	F	G	H	I		
カタゴリー	項目番号	カスタマイズ項目(目的)	選択肢番号	選択肢	設定内容の範囲 (E801)	反映した設定値 (E801)	E1076→E1081 での実装	確認手順書付 記録番号
Install	1	インストーラの表示名	1	fx Meta Installer (既定) 2 任意の名前	#define PRODUCT_FULL_NAME "（名前）"	#define PRODUCT_FULL_NAME "Demo fx installer"		
	2	インストーラのファイル名	1	fxMetaInstaller	-			
	3	インストーラの動作モード	2	任意の名前	#define PRODUCT_NAME "（名前）"	#define PRODUCT_NAME "DemoInstaller"		
	4		1	クイックモード表示し、操作場所を表示する	#define PRODUCT_INSTALL_MODE "KERNEL"			
	5		2	クイックモード表示し、操作場所を表示しない(既定)	#define PRODUCT_INSTALL_MODE "PASSIVE"	#define PRODUCT_INSTALL_MODE "PASSIVE"		
	6		3	クイックモード表示しない	#define PRODUCT_INSTALL_MODE "QUIET"			
	7	インストール完了後のメッセージ	1	表示しない(既定)	#ifndef _WIN32			
	8		2	表示する	#include <iostream.h> #include <windows.h> #include <fxMetaInstaller.h> #include <fxMetaInstaller.h>			
	9		3	表示しない(既定)	-			
	10	インストール完了後の再起動要求	4	表示する	#ifndef _WIN32			
	11		5	表示しない(既定)	#endif			
	12		6					
	13		7					
	14		8					
	15		9					
	16	関数 項目の見出し列 (B~C) のセルを結合する(行番号, 項目):						
	17	項目の選択肢の数が1以下だったら:						
	18	何もせずに終了						
	19	そうでないなら (選択肢が2つ以上あるなら)、						
	20	「見出し列の定義の配列」の全要素について 要素の番号 を使って:						
	21	シートの範囲を結合(
	22	開始行 → 行番号,						
	23	開始列 → 要素の番号,						
	24	終了行 → 行番号 + 選択肢の数 - 1,						
	25	終了列 → 要素の番号)						

実際のコード

```
def m(s, n, i):
    if len(i['o']) <= 1:
        return
    s2 = s.s
    for i2, c in enumerate(hcol):
        s2.merge_range(n, i2,
                       n + len(i['o']) - 1, i2,
                       '')
```

書いた本人の視界

関数 項目の見出し列 (B~C) のセルを結合する(行番号, 項目):
項目の選択肢の数が1以下だったら:

何もせず終了

そうでないなら (選択肢が2つ以上あるなら)、
「見出し列の定義の配列」の全要素について、
要素の番号 を使って:

シートの範囲を結合(開始行番号, 開始列番号,

開始行番号 + 選択肢の数 - 1, 終了列番号)

```
def m(s, n, i):
    if len(i['o']) <= 1:
        return
    s2 = s.s
    for i2, c in enumerate(hcol):
        s2.merge_range(n, i2,
                       n + len(i['o']) - 1, i2,
                       '')
```

第三者の視界

```
def m(s, n, i):
    if len(i['o']) <= 1:
        return
    s2 = s.s
    for i2, c in enumerate(hcol):
        s2.merge_range(n, i2,
                       n + len(i['o']) - 1, i2, '')
```

関数 何か(何か, 何か, 何か):

何かの数が1以下だったら:

何もせず終了

そうでないなら、

何か = 何かが保持している何か

何かについて、何か を使って:

何かの範囲を結合(何か, 何か,

何か + 何か - 1, 何か)

前情報無しだと、意味ある情報を読み取るのは困難……

リーダブルなコードの場合

```
def try_merge_item_heading(self, row, item):
    if len(item['options']) <= 1:
        return
    sheet = self.sheet
    for index, _column in enumerate(HEADING_COLUMNS):
        sheet.merge_range(row, index,
                          row + len(item['options']) - 1, index,
                          '')
```

関数 項目の見出しセルを結合してみる(自分, 行, 項目):

項目の選択肢の数が1以下だったら:

何もせずに終了

そうでないなら、

シートは自分が保持している

見出しの列 の全項目について、 番号 を使って:

シートの、範囲を結合(行, 番号,

行 + 項目の選択肢の数 - 1, 番号,
,

先の例よりは意味のある情報を読み取れるはず

よく聞く話

- ✓ 手紙を書いたら一晩寝かせろ！
- ✓ 試験の回答文は、最後にもう一回読み返せ！

**前提を知らない読み手
の気持ちになろう**

実際のコードを読んでみよう

Excelのワークシートを 自動生成するPythonスクリプト

	A	B	C	D	E	F	G	H	I
	カテゴリー	項目設定番号	カスタマイズ項目(目的)	選択肢番号	選択肢	設定内容の履歴(E801)	反映した設定値(E801)	E807-E1891での変更	候補中候補対応番号
1	Install	1	インストールの表示名	1	Fn Meta Installer(既定)	-			
2		2	任意の名前	2	任意の名前	define PRODUCT_FULL_NAME "(名前)"	define PRODUCT_FULL_NAME "Demo Fn installer"		
3		3	インストーラーの動作モード	1	FnMetaInstaller	-			
4		4	1) ウィザードを表示し、操作画面を表示する	2	任意の名前	define PRODUCT_NAME "(名前)"	define PRODUCT_NAME "DemoInstaller"		
5		5	2) ウィザードを表示し、操作画面を表示しない(既定)	3	FnMetaInstaller	define PRODUCT_INSTALL_MODE "NORMAL"	define PRODUCT_INSTALL_MODE "NORMAL"		
6		6	3) ウィザードを表示しない	4	FnMetaInstaller	define PRODUCT_INSTALL_MODE "FRESHIE"	define PRODUCT_INSTALL_MODE "FRESHIE"		
7		7	4) インストール完了後のメッセージ	5	FnMetaInstaller	define PRODUCT_INSTALL_MODE "NONE"	define PRODUCT_INSTALL_MODE "NONE"		
8		8	1) 表示しない(既定)	6	表示する	fninstall1.ini: [Install] FinishInstallMetaInstaller FinishInstallMetaInstaller			
9		9	2) 表示する	7	FnMetaInstaller	[Install] [Install] FinishInstallMetaInstaller FinishInstallMetaInstaller			
10		10	5) インストール完了後の再起動要否	8	表示しない(既定)	fninstall1.ini: [Install] [Install] FinishInstallMetaInstaller FinishInstallMetaInstaller			
11		11	9) 表示する	9	FnMetaInstaller	[Install] [Install] FinishInstallMetaInstaller FinishInstallMetaInstaller			
12		12	6) インストールのウイザードの表示言語	10	日本語(既定)	define PRODUCT_LANGUAGE "Japanese"	define PRODUCT_LANGUAGE "Japanese"		
13		13	7) 英語	11	英語	define PRODUCT_LANGUAGE "English"	define PRODUCT_LANGUAGE "English"		
14		14	7) Firefox/Thunderbirdのバージョン	12	指定しない(既定)	-			
15		15	2) 指定バージョンを用意する	13	Firefox/Thunderbirdのバージョン	fninstall1.ini: [Install] MozillaVersion=91.0 MozillaVersion=91.99 ApvlID91tree ApvlID91tree ApvlID91tree ApvlID91tree	fninstall1.ini: [Install] MozillaVersion=91.0 MozillaVersion=91.99 ApvlID91tree ApvlID91tree ApvlID91tree ApvlID91tree		
16		16	8) Firefox/Thunderbirdのインストール先	14	1) 現在のインストール先(既定)	Firefox-setup.exe/Thunderbird-setup.exe: [Install] InstallDir=ProgramFilesMozilla Firefox InstallDir=extInstall InstallDir=extInstallPath			
17		17	2) 任意のインストール先	15	Firefox/Thunderbirdのインストール先	Firefox-setup.exe/Thunderbird-setup.exe: [Install] InstallDir=extInstall InstallDir=extInstallPath InstallDir=extInstallPath	Firefox-setup.exe/Thunderbird-setup.exe: [Install] InstallDir=extInstall InstallDir=extInstallPath InstallDir=extInstallPath		
18		18	9) フォントメニュー、タスクバー上のショートカット	16	1) 表示しない(既定)	-			
19		19	2) 指定する	17	フォントメニュー、タスクバー上のショートカット	fninstall1.ini: [Install] DisplayVersion=100	fninstall1.ini: [Install] DisplayVersion=100		
20		20	10) スタートメニュー、タスクバー上のショートカット	18	1) 表示しない(既定)	-			

ビジネス上の要件

- ✓ Firefoxを法人運用向けに設定したい
 - ✓ 設定を一覧で管理したい
 - ✓ 「自動更新：有効/無効」など
 - ✓ 顧客ごとに設定内容を変えたい
 - ✓ バージョンごとの変化を見たい
 - ✓ Firefox 78からFirefox 91の間で追加された設定、廃止された設定

元々はExcel|ワークシートを手作りしていた

- ✓ Gitでバージョン管理しにくい
- ✓ 同じ変更内容を複数顧客のワークシートに反映するのに手間がかかる

どうにかしたかった

自動化を図った

- ✓ 資料 자체をバージョン管理しやすく
 - ✓ プレーンテキスト形式のソースからExcelワークシートを自動生成
 - ✓ 1. 設定項目の定義
 - ✓ 2. 前バージョンのFirefox用の設定情報
 - ✓ 3. 今バージョンのFirefox用の設定情報
- ✓ 初版はPythonに慣れた人が実装

読んでみよう1

build-xlsx-1.py
ざっと見て概要を掴んでみよう

ポイント

- ✓ 少数の短い関数
- ✓ 分かりやすい名前付け
- ✓ 列番号はべた書き

試しに探してみよう1

- ✓ 「... → ... での変更」の列に
「新規」と出力する条件は
どこで判定している？

試しに読んでみよう2

build-xlsx-2.py
第三者が手を加えた物

改修の経緯

- ✓ 要件が増えた
- ✓ 比較対象の数が可変になった
 - ✓ 改修前：「Firefox 78」と
「Firefox 91」
 - ✓ 改修後：「Firefox 78」と
「Firefox 91(デスクトップPC)」
「Firefox 91(ノートPC)」…
- ✓ 行数は約1.2倍に増加

試しに探してみよう2

- ✓ 「検証手順書対応番号」の列に出力する内容はどこで決まっている？

さっきより大変なはず

リーダブルだった書き方がアシリーダブルに……

- ✓ 関数の数が少ない
- ✓ 個々の関数が肥大化し、全体像の把握が困難に
- ✓ if-elif/elseでの条件分岐
- ✓ 階層が複雑化して処理の実行条件を追いにくい

試しに読んでみよう3

build-xlsx-3.py
大幅に改修した物

改修の要旨

- ✓ 複雑化した問題に合わせて
「リーダブル」の基準を変えた
何がリーダブルかは状況に依存する
- ✓ 定数を使う部分を増やした
- ✓ メソッド・変数のスコープを小さく
- ✓ コードの行数は約1.5倍に増加

試しに探してみよう3

- ✓ 「検証済み」という列を
「検証手順書対応番号」
の列の右隣に追加したい
(セルの内容は空でよい) が、
どこに手を入れればいい?

試しに探してみよう3

- ✓ 「検証済み」という列を
「検証手順書対応番号」
の列の右隣に追加したい
(セルの内容は空でよい) が、
どこに手を入れればいい?
- ✓ →探し方のコツがある
 - ✓ 改修前：全体を読んで覚えて探す
 - ✓ 改修後：必要な部分だけ読んで探す

つまり、ここでの「リーダブル」とは

- ✓ 全体を一望するのが難しい複雑なコードについて
- ✓ 一度に読むことは諦めて
- ✓ 必要な部分だけ読む

という前提での「読みやすい」

ふつうのOSS開発者の日常

✓ ✗ イメージ

- ✓ コード全体を詳細・完璧に把握
- ✓ ノールックで修正

✓ ○ 実態

- ✓ 全体像はボンヤリ把握
- ✓ 都度必要な部分を読み直して調べ直しながら修正

自分で書いたコードでも、覚えてないのが当たり前！

- ✓ 覚えておかなくていいようにするのがリーダブルコード
- ✓ 書籍「リーダブルコード」巻末の「解説」も読んでみて！

リーダブルコードの実践

- ✓ 別の切り口でも考えてみよう
- ✓ なぜアンリーダブル化するのか？

アンリーダブル化する原因

考えてみよう

アンリーダブル化する原因 (例)

- ✓ コードが**状況の変化**に追従できていない
- ✓ コードの規模や前提は**徐々に変化**する
 - ✓ 気付かないうちに進行する！

状況の変化に気付けても、追従できない……

- ✓ 既存のコードを書き直せない
 - ✓ 書き直して動かなくなるのが怖い
 - ✓ 「とりあえず動くように」で焦って「書き足す」一辺倒になる
 - ✓ 冷静に見直す余裕がない

→ 良くない兆候を見て見ぬフリ

ベテランはどうする？

- ✓ 悪い兆候を見逃さない
- ✓ 既存のコードを書き直すのをためらわない
- ✓ 書き直さない方が、後で痛い目を見ると知っている

悪い兆候に敏感になろう

- ✓ マメに読み返していると、アンリーダブルのなり始めにすぐ気付ける
- ✓ 「アンリーダブルになり始めてる……？」
→書き直しを考えるタイミング
後回しにしない!
(すぐやらないなら、せめてコメントを残す)

ためらわずに書き直せるよう にするために(1)

- ✓ わけが分からないままにしない
- ✓ **自分が今何をしているか**を
正しく理解するよう努めてる
- ✓ 不安が少しでも生じたら
立ち止まって考える

ためらわずに書き直せるよう にするために(2)

- ✓ 便利な道具で補う
 - ✓ 自動テストで**結果の同値性**を保証
自動テストは大事！
 - ✓ Gitで**いつでも巻き戻せる**安心感
バージョン管理システムは大事！
 - ✓ コードフォーマッターで
安全にコードを整形
「リーダブルにする」はある程度自動化できる

早め早めのリカバリー

- ✓ なるべく早くリーダブルに直す
- ✓ 書く→直す のサイクルを回す

参考：良い分割の仕方

- ✓ "良いコードとは何か - エンジニア新卒研修 スライド公開"
- ✓ https://note.com/cyberz_cto/n/n26f535d6c575#E0aBe
 - ✓ サイバーエージェント社の新卒研修の資料
- ✓ 「凝集度と結合度」が参考になる

リーダブルコードの練習

- ✓ コードを書く→読む→直す
のサイクルを体験しよう

チームで開発してみよう

- ✓ まずは、チームの中でコードを読みあってみる
- ✓ 自分で書いた物を読むよりは客観的に読みやすい
- ✓ 交代で書く / 感想を述べ合う

注意点

- ✓ なるべくポジティブな提案を
 - ✓ ○ 「この書き方はリーダブルだね」
 - ✓ × 「これはアンリーダブルだね」
(粗をあげつらうだけなら誰でもできる)
 - ✓ ○ 「こうした方がリーダブルじゃない?」
- ✓ チーム内で皆が納得できる
リーダブルの基準を見つけよう
- ✓ 既存の基準をベースにするのはアリ
(例: 書籍「リーダブルコード」の内容など)

明日の後半戦

- ✓ (進捗次第では、
今日の後半の続きをやる)

明日の後半戦

- ✓ 他のチームが書いたコードを読んでみる
(自分のチームの物を読むよりは客観的に読みやすい)
- ✓ チーム間で感想を共有
- ✓ より多くの人が納得できるリーダブルの基準を見つけよう
- ✓ 開発を継続してみる

最終目標：コードを読む文化 を持ち帰ろう

- ✓ 「コードは読む物」
という認識を持つ
- ✓ 自分だけからチームへ
- ✓ チームだけから全体へ

この後の予定まとめ

- ✓ 本日の後半：課題を実装
 - ✓ リーダブルコードを書く体験
- ✓ 明日の前半：実装チェンジ
→開発継続
 - ✓ 既存のコードを読んで変更する体験
- ✓ 明日の後半：ふりかえり
 - ✓ リーダブルコードの基準を共有する体験

おさらい

- ✓ 講座の目的？
- ✓ リーダブルコードの必要性？
- ✓ 講座でやること？

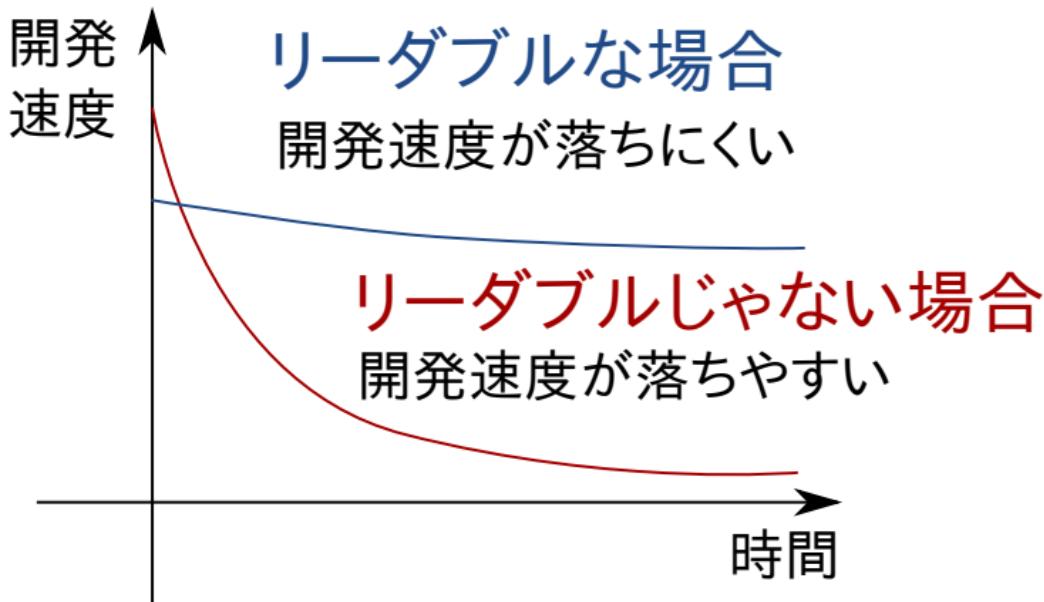
講座の目的

- ✓ リーダブルコードを日常的に書く上での基礎となる考え方を実践し、持ち帰る

リーダブルコードが必要な理由

- ✓ 既存のコードを読んで素早く内容を把握したい
- ✓ 既存のコードに素早く手を加えたい
- ✓ 開発速度を落としたくない

変更コストと開発速度



講座でやること

- ✓ コードを読む文化作りの体験
 - ✓ チームの中でコードを読みあってみる
 - ✓ チーム内でリーダブルコードの基準を共有する
 - ✓ 他のチームともリーダブルコードの基準を共有する

ここまで説明

腑に落ちました
か？

ここから先は、「まとめと次のステップ」の内容です

次のステップ

✓ もっともっとコードを読もう！

次のステップの例1

- ✓ ゼミの他の人や
同じ学部の人が書いたコードを
読んでみよう
- ✓ 研究の合間に
- ✓ 論文執筆の合間に

次のステップの例2

- ✓ 使うライブラリやツールのコードを読んでみよう
- ✓ OSSのコードはリーダブルであることが多い
- ✓ 「こういう結果を得るにはこう書くんだ！」と実感を伴って読める

次のステップの例3

- ✓ コミット単位で読んでみよう
- ✓ コード全体ではなく差分を読む
- ✓ コードの中身・設計の仕方ではなく
コードの書き方・開発の仕方に注目
する
- ✓ リーダブルなコードを
見つけるのに使いやすい

コードを読む文化を後輩につなごう

- ✓ ゼミの後輩にも「コードを読む文化」に馴染んでもらおう
- ✓ 皆さんのが書いたコードが資産として受け継がれる
- ✓ 口頭での詳しい説明なしでもコードを読んで伝わる情報が増える
後輩に手取り足取り教えずに済む→先輩も楽になる

新しい人との関わりが、リード ダブルの基準の見直し機会に なる

これまで大事にしてきたことを
後輩と共有



- ✓ 「もっとこっちの方が
リードダブルでは？」
- ✓ リードダブル基準の見直しのよい機会

サポート

- ✓ 今日の資料はすべて再利用可能
[https://github.com/clear-code/readable-code-workshop/
tree/master/20210913](https://github.com/clear-code/readable-code-workshop/tree/master/20210913)
(<https://slide.rabbit-shocker.org/authors/Piro/>)
- ✓ 迷ったら読み返せる

クリアコード

- ✓ クリアなコードが大切
 - ✓ クリア == clear == 意図が明確
 - ✓ クリアなコードはリーダブルコード

みんなのコーディングライフで
リーダブルコードが当たり前に
なることを応援します！

課題（宿題）

- ✓ 過去作成した何らかのプログラムについて
- ✓ リーダブルになるよう編集する
- ✓ リーダブルにした点を1つ以上メモに書き出す
- ✓ 以下の3つを併せて提出する
 - ✓ 変更前のプログラム
 - ✓ 変更後のプログラム
 - ✓ リーダブルにした点のメモ