イントロダクション

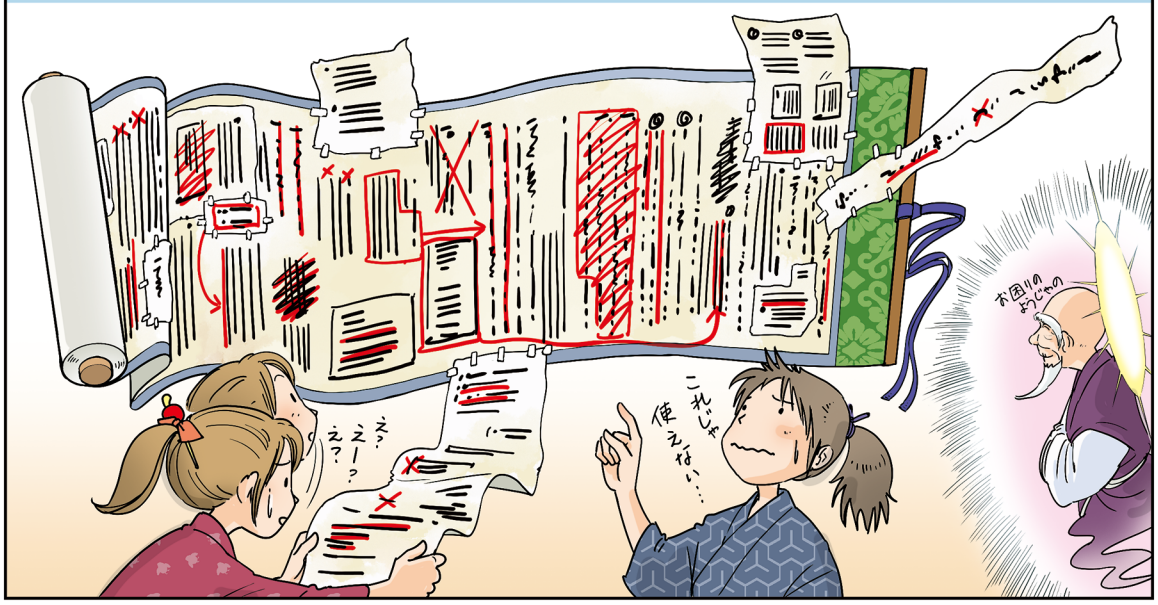
開発にスピードが求められる時代だからこそ重要…！

リファクタリング＆移植

編集部

先人の知恵である貴重な設計資産…活用しない手はない！

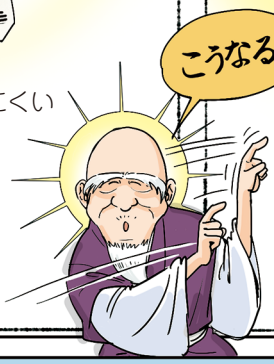
しかし…修正や機能の追加の積み重ねで解読困難



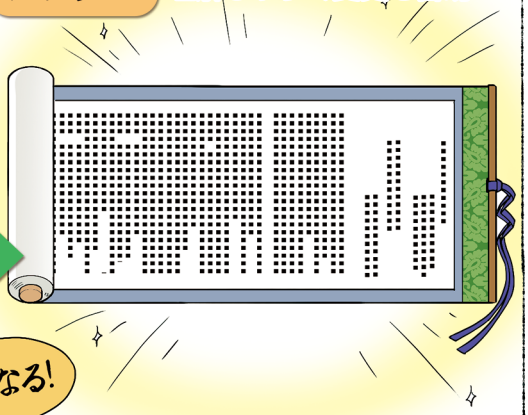
解読困難になる前に…リファクタリングのススメ

ビフォー 解読や検証に時間がかかる

- 未使用コードが残っていて見にくい
- 処理が分かりにくい
- 書き方が人によりバラバラ
- 関係する処理があっちこっち
- 意味不明の定数
- 関数が長くて読みにくい
- 何の関数なのか不明



アフター 理解しやすく変更も容易



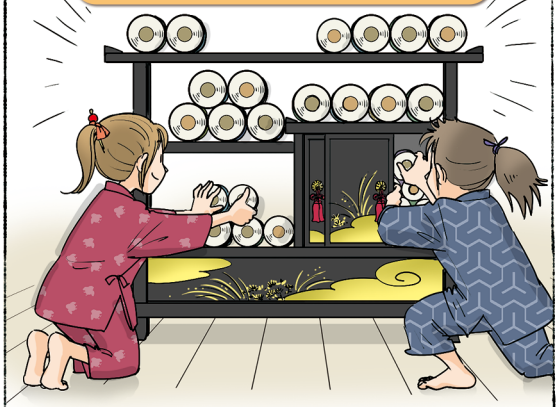
- スッキリして見やすい
- 処理が分かりやすい
- 書き方が統一されて読みやすい
- 処理がまとまって理解しやすい
- マジック・ナンバで意味が分かる
- 短い関数で読みやすい
- 関数の意味が理解しやすい

特集ではリファクタリング＆移植の技100を紹介

リファクタリング

- 体験しながら学ぶリファクタリング
- 5分で終わるリファクタリング
- リファクタリング後のテスト
- リファクタリング事例集
- Pythonでもリファクタリング

移植



- 長く使うコードこそ移植が重要
- 移植しやすいソフトウェアの事例
- Pico 2やLinuxなどの移植事例
- PC用Pythonプログラムをマイコン用MicroPythonへ移植する時の勘所