第2章 ①コメント追加，②状態変数の見直し，③ラムダ式の導入で改善

論理的文章でリファクタリング③…GUIアプリの機能分離

村井 和夫

システム設計やプログラム設計の段階で機能分解が十分にされていないと読みにくくなります．このようなことがよく分かる例として，通信処理とGUI機能を題材に，リファクタリングを行います．

GUIプログラムの機能分離

● 題材…機能分解とインターフェースが不明確なプログラム

PC用の電話プログラムをChatGPT o3-mini-highに作らせたところ，典型的な問題を抱えるプログラムを出力しました．WindowsのGUIベースのプログラムによくある問題です．これを例に機能分解点とインターフェースの重要性，カプセル化の例を見ていきます．

C++でQtを使った電話画面のプログラムです．この場合大きく電話画面のGUIと，電話通信処理をつかさどる2つのプログラム・スレッドに機能分離できます．この2つの機能は，インターフェースを介して極力独立にプログラムを書かなければなりません．この例では明らかな機能の重複が生じています．

リスト1に電話画面のGUI部分のプログラムを，分かりやすく説明に必要なところだけ切り出して示します．ダイヤルした番号の表示BOX，ダイヤル・パッド，発信・応答ボタンがあります．

● 問題点と修正方法

電話画面のGUIプログラムの中に電話通信状態が深く入り込んでしまっています．このプログラムの問題点は次の通りです．

▶①コメントが少ない

コメントが非常に限られているのでQtを知らないとどこで何をやっているのかさえ分かりません．

ここでは必要な部分だけ切り出していますので，かなり分かりやすくなっていますが実際のプログラムは非常に分かりにくいものです．

▶②状態変数を重複して持っている

GUIのWindow初期化を見れば分かりますが，通信処理のためのオブジェクトを全て抱え込んでいて，しかも発信ボタンを発信処理にするか応答処理にするか判断したり，ボタンの表示を変更したりするため，中途半端な`incomingCallWaiting`というフラグまで持っています．

これは通信プログラムのスレッドで当然分かっているもので，このような他で管理されている意味のない状態変数を重複して持つこと自体重大なバグの原因となります．

▶③コールバック処理を関数化している

ダイヤル・ボタンが押されたときの，数字をダイヤル・ボックスに反映させるコールバック処理を関数化しています．これそのものは決して悪いことではありませんが，関数化すると誰が呼ぶのか気にしなければなりません．たとえ`private`関数としても，間違ってオブジェクト内の他のところから呼ばれてしまう可能性もあります．また関数の実態の場所も分かれていて分かりにくくなっています．

このように再利用性がなく，しかも単にダイヤル・ボックスに数字を加えるだけのコールバック処理は，後述するラムダ式で処理した方がすっきりします．

技1 コメントを入れてリファクタリングする

これらの点をふまえて，修正した結果が**リスト2**です．これも，分かりやすさのため必要なところだけ切り出しています．

仕様書の章に相当する処理の場所には，章のタイトルと同じように先頭に見出しコメントが入っているのでブロックの区切りが分かり，そこで何を処理するかが一目瞭然となります．

当然のことながら，仕様書の章，節，段落に相当する処理は，インデント・レベルが同じになっているので処理の流れが明確になります．