今週の一コマ画像

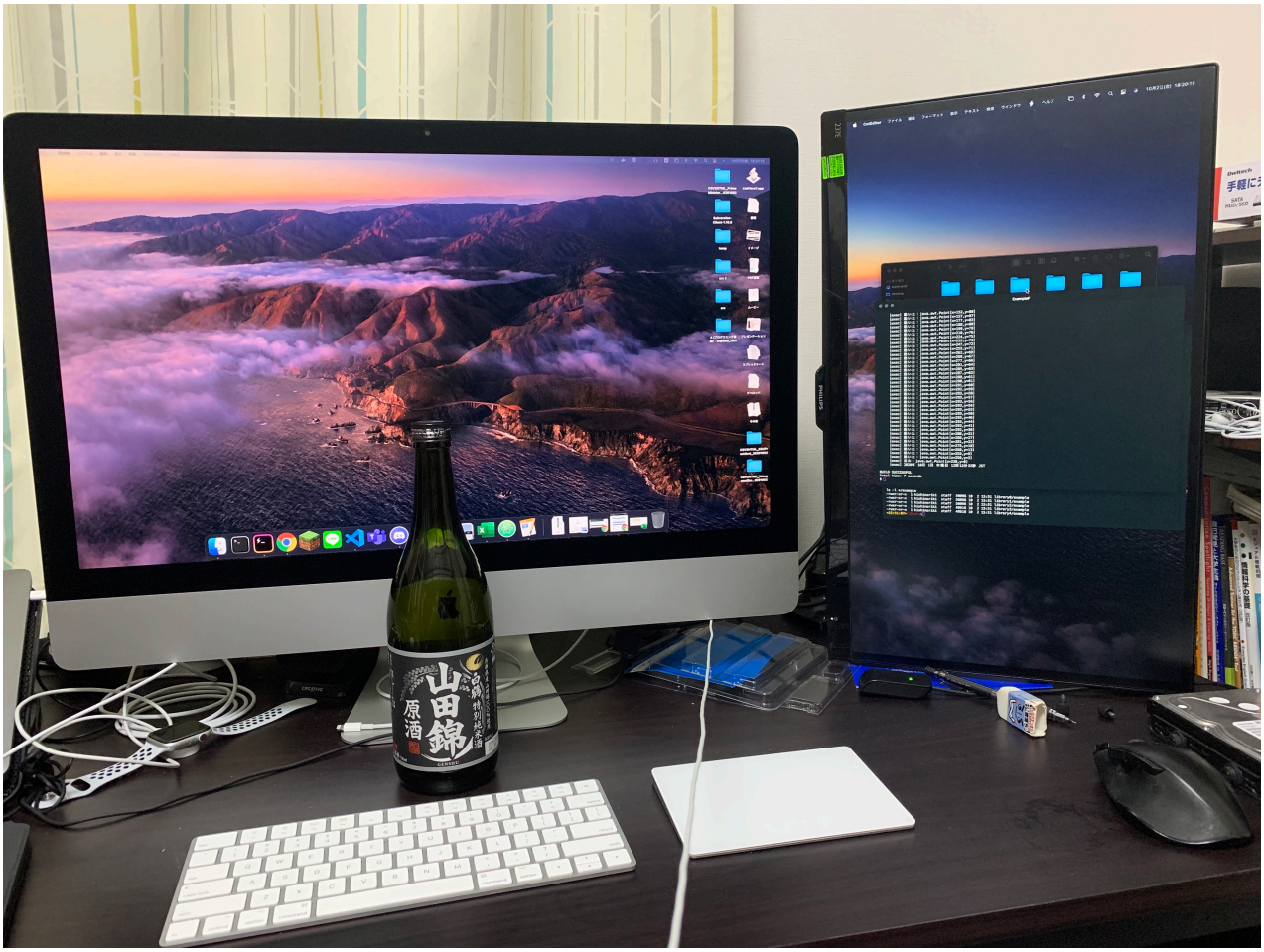

図0…酒とMacとオイラと課題(酒と泪と男と女 風)

APJ02

学籍番号: 744366

氏名: 岸 典樹

2020年10月2日

第1章 EXAMPLEのリファクタリングの変遷

本章ではレポジトリからチェックアウトしたExampleについて、Example{A..F}とリファクタリングを重ねたことで得られたことをまとめる。

1.1 プログラムの共通の目的

Example, Example{A..F}は共通して、java.awtパッケージを使用したアプリケーションである。起動すると灰色塗りの600x400のウィンドウが起動し、黒の48pxのSerif体で”京都産業大学”という文字が出力される。また、本プログラムはウィンドウのサイズを調整することができる。これらのプログラムをclang-format-for-java.yaml, build.xml, example.mf, Makefile, Apache Antを使用してexample.jar, Classes/example/Example.classを作り、実行している。

1.2 Example

Exampleは1.1節で説明したプログラムをこれ以上拡張することを考慮せず記述されている。JavaDocが記述されていたりpaintComponentクラスをオーバーライドしている点、voidもreturn;を入力している点はなど、javaプログラム上の様々な点を見ると非常に美しいコードではあるが、ファイル分割やメソッドの分割にまだまだ改良の余地があるため、小規模開発限定の書き方と言えるだろう。

1.3 ExampleA

ExampleAはExampleからView周りのコードを切り分けた、差し詰めVモデルといった書き方だろうか。UMLを見た上でも、変更点はこの辺りだけであり、Exampleに比べ、オブザーバモデル導入の準備を始めたものだと考察できる。

1.4 ExampleB

ExampleBはExampleに比べ、フィールドとコンストラクタの概念が追加された。インスタンス化を行う際にフォント関連の処理を既にしているから、画面が立ち上がってから文字が表示されるまでの遅延が限りなく0に近づいた。

1.5 ExampleC

ExampleCはCondition.java、Interval.java、ValueHolderを呼び出し、よりオブジェクト指向らしい記述になった。さらに、画面のリサイズに伴い適切なフォントサイズに切り替わるよう調整されていることが伺える。この辺りからif, else if, else, while, for文を使わずlambdaを積極的に使うようになったため、一気にコードのレベルが上がり、美しくなった。さらに、Condition.java, Interval.java, ValueHolder.javaをCondition.jarとしてコンパイルすることで多少セキュアなシステムになったものだとも考えられる。

ExampleC直下に置かれているcondition.jarの中身を知るためでコンパイルしてみたが、発展プログラミング演習や第二回までの応用プログラミングjavaでは教わらないかなり高等なスタイルでコーディングされていたので紐解くのに少々時間を要したが、おそらく各ファイルの用途は以下の通りである。

表1.5.1 Condition.jarをでコンパイルして取得したファイルの役割対応表

ファイル名	役割
Condition.java	条件分岐、繰り返し、周りの処理をオブジェクト化
ConditionException.java	toStringの上書き及びbreakやreturn周りのオブジェクト化 / 例外処理
Example.java	作成したクラスをテストするプログラム
Interval.java	forEachを使用したインターバルをクラス化したもの
IntervalException.java	Intervalクラスで使用する例外的メソッド群
ValueHolder.java	マルチスレッドを使用し、原子レベルで値を読み書きするクラス

しかし、個人的にCondition.javaコーディングスタイルが青木先生らしくないと考える。授業で提示していただくエクセレントなコードは我々に教えていただくために非常にわかりやすく記述したものであり、本当はCondition.javaのような記述がjavaを扱うプログラマには求められているのか、どこかのチームが開発したプログラムなのか、その点について疑問が残る。
少なくとも、Objectを継承していない点やjavaDocが記述されていない点はでコンパイルした際に復元できなかったとも考えにくいので、正解を知りたいのである。

1.6 ExampleD

ExampleDからはMVモデルとなった。ウィンドウの起動やフォントサイズ、表示する文字列の取得などはModelで行い、Viewではそれらを使用して画面表示とリサイズに応じたフォントサイズの変更などを行っている。また、Example.javaではModelの呼び出しを行なっている。

1.7 ExampleE

ExampleEでMVCモデルが導入される。さらに、Pointクラスを使用することで画面内にマウスカーソルがあるかどうか、ドラッグされたか、移動されたかなどのいくつかのアクションが検出できるようになった。

1.8 ExampleF

ExampleFはExampleEに比べ、特に機能の追加はみられなかったが、以下のような一部の記述が更新されている。

・Viewのフィールドのインスタンス化がコンストラクタ内に移動
・明示的な型定義
・引数となる一部のlambdaをダブルコロンに変更

まとめると、ExampleEの美しくない点をさらにブラッシュアップしたプログラムであった。

1.9 ためになったこと・これからしなければならないこと

全体を通して、アーキテクチャレベルでのファイル分割やオブジェクト指向の全面活用について学べるものが多かったが、個人的に一番の学びはデコンパイルしたConditionの記述方法である。JavaDocが記載されていない点など、いくつか教えに反して参考にならない点も多々あるが、全体的にJavaに関する知識不足を痛感したため、Javaプログラミングをする時間はしっかりとオブジェクト指向を学び、さらに高度なJavaへ入門したく思う。

第2章 EXAMPLEの設計図

本章ではレポジトリからチェックアウトしたExampleについて、それぞれの設計図について何を物語るのかをまとめる。

2.1 ExampleとExampleAの差

ExampleとExampleAの差は、Viewを切り分けたことにある。Web界隈のモダンなフロントエンドフレームワークは特に、エントリーポイントとなるファイルに重要なプログラムは書こうとしない。それは、エントリーポイントに書く”べき”処理は限られていて、必要な役割を果たすファイル(クラス)に処理を委託することで可読性が向上するからだと考える。同時に、UML上のクラスやパッケージを整理することでエンジニアがより構造をより理解しやすくなっている。違うサイズのネジを同じ引き出しにしまう現象と少々似ている。

2.2 ExampleAとExampleBの差

ExampleAとExampleBの差分はフィールドとコンストラクタを定義したことにある。コンストラクタを定義することでクラスが呼び出された際の処理のエントリーポイントがプログラム上でわかるほか、UML上ではそのクラスがどのようなフィールドを持つのか把握できるため、拡張がしやすくなる。

2.3 ExampleBとExampleCの差

ExampleBとExampleCの差はフィールドの型にジェネリクス型が投入されたことである。ExampleCからは画面のリサイズに伴いフォントサイズを変更する処理が組み込まれているが、fonts, width, heightがジェネリクス型になったことで、およそどのような処理を行なっているのか推測できるようになる。

2.4 ExampleCとExampleDの差

ExampleCとExampleDの差はModelが追加されたことである。ウィンドウの起動処理をViewからModelに移動させることでMVモデルが実現できていることがわかる。また、Viewのフィールドにあるprotected型を採用していることがわかるため、ViewにとってModelが保護されていることがわかる。

2.5 ExampleDとExampleEの差

ExampleEの差はjava.awt.eventとjava.swing.eventを呼び出してイベント処理を行なっているほか、example内でオブザーバモデルが実現できていることがわかる。Controller内のメソッドおよび選文の関係性を考えるとevent処理はModelではなくController側で処理されていることがわかるため、中で綺麗に役割分担ができていることが評価される。しかし、ここにきて少々疑問に思うのがExampleCからジェネリクス型を導入していたにもかかわらず、ここにきてUMLにjava.utilが追加された点である。本プログラムは我々

にリファクタリングの礎を教えていただくものだから、ジェネリクス型を使用した際にはjava.utilをUML上に書く必要があるという強いメッセージ性があるのかもしれない。また、javax.swing.JComponent, java.awt.event.MouseListener, java.awt.event.MouseMotionListener, java.awt.event.MouseWheelListenerに関してはjava本来のパッケージの中でも瓶詰で記述されているため、それらの中でどの処理を使用しているのかがわかる。

2.6 ExampleEとExampleFの差

ExampleEとExampleFの差は以下の3点である。

①example.Modelのsizesフィールドのジェネリクス型の採用
②Viewとjava.util.ArrayList<Font>, java.util.ArrayList<Integer>の依存関係の修正
③java.util.ArrayList<Integer>の多重度の明示

プログラム上の変更点は微々たるものであるが、

2.7 UMLをつぶさに観た上で変だと感じた点

僭越ながら、以下に、ExampleFまでのUMLの中で違和感を感じた点をまとめます。

①”Java.~”と”java.~”の混在

ExampleFにおいて、java.lang, java.awt.event, java.swing.eventのjavaが大文字から始まっていました。これについては意図的なものなのか、誤操作なのかが気になりました。

②プログラムとUMLの差分

UML上のExampleFのexample.Modelにおいて、getSizes()メソッドはInteger[]型を返しますが、プログラム上ではリファクタリングされているためList<Integer>を返します。フィールドとメソッドの間で型に違和感を感じたため、調査しました。

あくまで推測にはなりますが、これらも”つぶさに観る”ことの評価ポイントではないかと考えていますが、言葉の選択肢が少なく恐縮ですがもし失礼にあたればこの場を借りてお詫び申し上げます。

2.8 会得したもの

以前、某リフォーム密着番組を見ていた際に、日本人の建築士がフランスのマンションをリノベーションする企画がありました。その中で、フランスの製材所の仕事は少々いい加減で、設計図よりサイズが前後する木材が送られてきて、作業が止まることがしばしばありました。一方日本国内で加工された木材は寸分狂わず精密に作られているため、四方金輪などのとても複雑な構造をした組み木を用意できたりします。近年ではその圧倒的強度や素材の耐火性などから住友林業などが木造ビルの建築を可能にしたり、津波があっても家が原型を留めたまま流されていくなど、不謹慎ながら世界的に異例の実績や現象が起きています。これらの耐久性は当然設計図がないと実現できないものであり、設計図のない家は国に登録されている建造物の中には存在していないと考えています。

ソフトウェアの開発もまた同じで、アジャイル開発であっても設計図は存在します。設計図を一枚もくれないクライアント様も過去には存在しましたが、設計図がないと時間と労力だけかけて良いものは決して完成しません。UML、外観図、状態遷移図、アローダイアグラム、ガンとチャート、PART図、事態図、アーキテクチャ図など、ソフトウェアを開発する上で必要不可欠と言っても過言ではない設計図は多々存在するため、設計図の重要性について本課題で再確認できました。

第3章 自由記述

3.1 課題が増えてきました。。

各授業の課題数が少々多くなってきました。今学期、特別研究1を含め18単位を習得することができれば来年は特別研究「のみ」に専念する準備が整うため、雨ニモ負ケズ風邪ニモ負ケズ青木先生ノレポートニモ負ケズ頑張っていきたいと思います。コンピュータ理工学部の最後の世代ということもあって単位習得が容易な科目がどんどん閉講し、数学・物理・脳神経科学・機械学習系科目が多く残るようになってきました。残りは選択科目のみにはなりますが、なんとしてでも今年で単位を揃えたいと考えています。

3.2 Ethernetが。。。

新しいMacにEthernetケーブルを接続していますが、どうもソフトウェア側が認識してくれません。我が家(戸建て)のルータは屋根裏にあるため、最近父が交換したルータが家(壁についてるソケット)につながっているかチェックする必要がありそうですが、かなり大掛かりな作業になるためあまりやりたくない自分もいます。でもやった方が圧倒的に通信ラグが減るんだろうなという葛藤に駆られています。いずれやらねば…

3.3 大阪都構想の住民投票まで1ヶ月を切りましたが。。。

大阪都構想の住民投票まで1ヶ月を切りました。私としては行政を整理して大阪をさらに豊かな大きな都市にしようという考えは賛成なのですが、居住区によって反対意見が多い行政区もまだ半数くらいあるみたいです。21世紀は都市の時代と言われており、20世紀に比べ圧倒的に交通インフラの整備や超高層ビルの充実が進みました。大阪にはまだ大阪万博、なにわ筋線建設、うめきた2期開発、私鉄や地下鉄の延伸・新設計画、アリーナ設置計画などが山のように残っています。圧倒的な東京を除き、46道府県のなかで圧倒的な大阪(京阪神ネットワーク)の地位を取り戻すべく、ぜひ成功してほしいなぁというのが個人的な意見です。つまり、選挙権のない(大阪市の隣の)豊中市民の私は賛成派です。

以前府知事と市長をになった橋本さんは当時「ワン大阪」という名で二重行政を撤廃するいわゆる都構想を言われていましたが、同じ頃、2つくらい隣の市の市議会議員さんも「ワン西宮」とおっしゃられていたのを思い出しました。ワン大阪は大阪府と大阪市の二重行政を1つにまとめることからワン大阪と言われていましたが、もともと西宮は1つではないかと記者会見で橋本さんにツッコミを入れられているその議員さんは時の人として日本の見事に日本の恥さらしになってしまいましたが、今思うと当時中学生だった自分はそれのおかげで政治に興味を持てるようになったので、ある意味感謝しないといけないのかもしれません。

3.4 参考文献

・Java スレッド処理でsynchronizedを利用する際の注意点(マルチスレッド)

http://glasses-se-note.com/java-synchronized/

・Javaチュートリアル 同期されたメソッド

https://docs.oracle.com/javase/tutorial/essential/concurrency/syncmeth.html

・AtomicReference<V>

https://docs.oracle.com/javase/jp/9/docs/api/java/util/concurrent/atomic/AtomicReference.html

・クラス Class<T>

https://docs.oracle.com/javase/jp/6/api/java/lang/Class.html

・