ソフトウェア演習 IV 手順書（２週目）　　2020/12/08 梅村恭司

インターフェースとテストケースとセルフテスト

1: 先週の作業に問題があるかを下記に記載したので問題のある人は、TA と相談して問題に対処せよ。担当の TA は先週と同じである。

https://docs.google.com/spreadsheets/d/1CkHiWKqc9vuiDFhvk-21alGtQbQ7J4KAqkNQLvQIT68/edit?usp=sharing

2: 演習のリポジトリ https://github.com/TUT3Software4Project/2020informationQuantity.git が更新されたのでそれを取り込め。具体的には下記を参照せよ。

演習の大元の GitHub 上リポジトリ：upstream と呼ぶ
各自に fork した GitHub 上のリポジトリ: origin と呼ぶ
各自のマシン上にある git の保存領域：local と呼ぶ

先週は下記を行った。

(1) GitHub 上で upstream を fork した（Web ブラウザでの操作）
(2) fork したリポジトリを clone （git コマンドでの操作）

上記の続き：upstream の変更を origin に反映させる手順は

(1) git remote add upstream <演習のリポジトリ URL>
　　; これにより upstream が登録される。
(2) git pull upstream main --no-edit
　　; これにより local に upstream の変更が反映される。
　　; もし、自分の場所以外を変更しているとエラーがでる可能性がある。
　　; その場合の対処は後述する。
　　; GitHub の変更により”master”ではなく”main”を使う。
(3) git push
　　; local を origin に反映させる。

2:補足. git pull でエラーがでたときの対処
（変更してはいけないものを変更したが原因であることが多い）

git merge --abort
　　; これにより、中途半端なマージの状態を取り消す。
master を upstream の状態に合わせる。
（単純な方法)、upstream を現在の作業とは別の場所に clone して、
エラーが報告されているファイルを、upstream のものにする（自分の変更を消す）
pull したあとに、自分の変更を行った状態にする。
（上記の具体的な操作がわからない場合は TA に相談すること。

3: ソフトウェア演習ビデオ 3「外部仕様・プロトタイプ」を視聴せよ。
アクセスする URL は、default.lecture.ss.cs.tut.ac.jp であり、
ユーザ，パスワードは先週と同じである。

4: Javadoc によるソースコードからクラスのドキュメントの html ファイルを生成せよ。

cd s4/specification; make javadoc; cd ../..
; make で実行されるコマンドについて、その意味を確認しておくこと。

5: FrequencerInterface.html, InformationEstimation.html を探し、ブラウザで読め。

通常の Java のクラスの説明と同じフォーマットであることを確認せよ。
これが、クラスの外部仕様となる。

6:. ソフトウェア演習ビデオ 4「テスト」を視聴せよ。
7: 参考資料、code-c-22b-0001.pdf を classroom から入手して読め。
7:. TestCase.java のなかでの Frequencer のテストコードについて、specification の記述ごとに対応するテストケースを１つ以上つくり、TestCase.java に記述を追加せよ。(これは，ブラックボックステストに相当する。)

8: サンプルで提供されている Frequencer.java のコードを読み、問題を発見しそれを書き出せ。
9:. サンプルで提供されている Frequencer.java コードの問題が明らかになるようなテストコードを追加せよ。（ホワイトボックステストに
10:. 変更した TestCase.java を commit し各自の github 上のリポジトリに push せよ。
commit の文字列の案、"TestCase.java： for Frequencer"
11: ビデオ「時習館高校・圧縮類似度と作曲者判定を視聴せよ」これは次の論文の理解を助ける。
12: Dream Campus より、論文「Computing Information Quantity as Similarity Measure for Music Classification Task 」を入手して読め。ただし、情報量とはどのようなものかと、情報量をどのようなデータを対象に計算するのかということが目的である。
-- これ以下については，情報量の外部仕様がある時間以上かけても分からなければ作業できなくてもよいが、次回の冒頭に確認をすること。
13：InformationEstimatorInterface の記述ごとに、テストケースを TestCase.java に追記せよ。
14: 変更した TestCase.java を各自のリポジトリに push せよ。
commit の文字列の案, "TestCase.java: for InformationEstimator"