Dr Kevan Buckley

Advanced Software Engineering Topics

Portfolio Task 2 Hints and Advice

Some Quick Wins

- Look at the marking grid to find out what your targets are.
- The trivial refactorings are worth as much as the complex ones.
- Consult the refactoring checklist and prioritise what you are going to attempt.
- Make a note that you need to demonstrate a competency of unit testing.
  - The only way you can do this is to extract some methods to test
- Make sure that you make regular commits/pushes. There are 3 reasons for this.
  - When you totally break your application you can roll back
  - Your tutors will be monitoring your progress
  - When you ask a tutor for advice they can see your code and documents
    - Please don't contact tutors through Canvas. They cannot reply directly and have to look up your email address.
    - The first 2 tasks are set and marked by Kevan. The other tasks are set by and marked by John. It is best to ask the person marking your work for advice, then there won't be any confusion.
    - For general advice John is the module leader and will be in the best place to help you.

Focussing on a Good Grade

- Assess your own work to estimate what your grade will be. Take into account that you might get some things wrong.
- There is one class that many students try to get rid of because it might be lazy or a data class.
  - One of the principles of OO is that you keep the methods that work on data should be close to the data.
  - If a class has no methods of its own but there are methods elsewhere that work on that data, the SRP principle would be violated if you moved the data. Move the methods.
- A quotation class and a quotation factory would enable you to demonstrate that you can test classes that have random numbers and test extracted methods.
  - Have a deep think about what other smells could be in those classes.
- There are opportunities to extract methods from human interface code that could be used for general testing. See the notes on TriangleTutor.

Be Pragmatic

- It is better to attempt a lot of small things than a small number difficult things.
- Some of the smells like shotgun surgery and divergent change require you to have a deep understanding of the code, which probably means renaming a lot of things and doing a lot of extracts.
- Doing 1000 extract methods will only get you one tick. You may have to do some replications to address some of the more difficult smells/refactoring.
- Make sure you document what you have done. You need to get credit for your hard work. If you don't signpost what you have done something might not be recognised in marking.