INFO8240 Assignment 2

Issued: January 28

Due: February 11

1. Study the Proxy and Bridge design patterns at `http://dofactory.com/net/proxy-design-pattern` and `http://dofactory.com/net/bridge-design-pattern`, respectively, as well as the program examples at `http://patterns.cs.up.ac.za/examples/ch2/proxy-spacebook.cs`, `http://patterns.cs.up.ac.za/examples/ch2/bridge-openbook.cs`, and `http://patterns.cs.up.ac.za/examples/appendix/adapter-pluggable-coolbook.cs`. Make the appropriate updates to *CoolBook* such that a user is able to send his or her friend a virtual gift. Such virtual gifts would normally be in the form of small graphics, but in this case let us simplify and make them textual. e.g. If Tom sends Judith a *heart* virtual gift then the textual output would be: `Tom: sent virtual heart`. You may also read *Structural Patterns: Adapter and Façade* at `https://msdn.microsoft.com/en-us/library/orm-9780596527730-01-04`, which was referenced in lecture a few weeks ago, for an explanation of CoolBook and example outputs of the *poke* operation.

2. Write the classes for the most recent update to the Monopoly case study using C#. That is, we now have the classes `MonopolyGame`, `Board`, `Square`, `PropertySquare`, `RegularSquare`, `GoSquare`, `IncomeTaxSquare`, `LotSquare`, `RailroadSquare`, `UtilitySquare`, `Die`, `Piece`, and `Player`. Ensure to reference the used design patterns in the updated code, including making `Board` a singleton class.

3. Refactor the SeaBird program example using your notes from lecture. Write comments in your code to document the parts that have been refactored.

4. Using the refactored SeaBird program from the previous task, consider the following demonstration of the program. An aircraft's engines are turned on and it then accelerates to a reasonably slow speed in order to align itself to the takeoff area. It then stops and waits for a moment, and then accelerates once more to a high speed, raises its nose to some degree, and takes off. It then continues to fly up until it reaches some height. The aircraft then continues to fly at this height for a little while and then increases its height once more. It continues to fly at the new height until it is close to its destination, at which time it lowers its nose to some degree and begins the landing process. Once it lands, the plane decelerates to a reasonably low and constant speed. It then continues at this constant speed until it decelerates once more, stops, and turns of its engines. *(continued on next page)*

How well does your program support the above simulation? Rewrite it such that it clearly contains the appropriate methods that allow the operations described above, noting the above is a rather simplified account of what actually occurs in flight simulations. Test your program by running it using the above description.

5. Using one of your past projects as an example, describe any five anti design patterns that appeared in that project. How may you use the lessons learned in that project such that these do not occur again in another project such as your upcoming capstone project work? Why do you think it is worthwhile for us to be aware of common anti design patterns?