Assessment Solutions

Chapter 20 – Removing Implementation Details Using the pImpl Pattern

1. Please see `Chapter20/Assessments/Chp20-Q1.cpp` in the GitHub repository.
2. Please see `Chapter20/Assessments/Chp20-Q2.cpp` in the GitHub repository. (Follow-up question) Simply inheriting `Student` from the `Person` class in this chapter that embraces the pImpl pattern presents no logistical difficulties. Additionally, modifying the `Student` class to also employ the pImpl pattern and utilize a unique pointer is more challenging. Various approaches may run across various difficulties, including dealing with inline functions, down-casting, avoiding explicit calls to the underlying implementation, or requiring back pointers to help invoke virtual functions. Please see the online solution for details.
3. Other examples which may easily incorporate the pImpl pattern for relative implementation independence include creating generic GUI components, such as for `Window`, `Scrollbar`, `Textbox`, and so on, for various platforms (derived classes). The implementation details can easily be hidden. Other examples include proprietary commercial classes that the developer wishes to hide the implementation details that might otherwise be seen in a header file.