Assessments

The programming solutions for each chapters' questions can be found in our GitHub repository at the following URL: `https://github.com/PacktPublishing/Demystified-Object-Oriented-Programming-with-CPP/tree/master`. Each full program solution can be found in the GitHub under the appropriate chapter heading (subdirectory, such as `Chapter01`) in the subdirectory `Assessments`, in a file that corresponds to the chapter number, followed by a dash, followed by the solution number in the chapter at hand. For example, the solution for question 3 in chapter 1 can be found in the subdirectory `Chapter01/Assessments` in a file named `Chp1-Q3.cpp` under the aforementioned GitHub directory.

The written responses for non-programming questions can be found in the following sections. Should an exercise have a programming portion and a follow-up question, the answer to the follow-up question may be found both in the next sections and in a comment at the top of the programming solution on GitHub (as it may be appropriate to review the solution in order to fully understand the answer to the question).

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.