How DynamicClass.cpp got divided for separate compilation:

1. Made new console application, cut/pasted original .cpp into it, did build, OK.
2. Solution Explorer window -> Right-click “Header Files” -> “Add” -> “New Item” ->”Header”
3. Chose name “fifo” leading to new file “fifo.h” appearing within the solution.
4. In that new .h file, replaced “#pragma once” with the method described in Savitch (p. 716), which is #ifndef <token> followed by #define <token>, with #endif at very end of .h file. Preventing the class declaration code being included multiple times, which compiler would reject.
5. CUT out of the original .cpp file the lines that included the class ***declaration*** only. Pasted them into the new .h file.
6. In the original .cpp file, then added #include “fifo.h” as the last file included, before the “using” statement.
7. Back in .h, noticed that there were now issues with the use of the “string” class. Needed to add “#include <string>” and “using namespace std;” to that file. (*Alternatively, could have made do with the statement “using std::string” see Savitch p. 724*)
8. Build now worked fine!
9. The fifo.h file is the “Interface Definition” file as discussed in Savitch Ch 12. Much commenting is placed in this file, because it is what other people will see in order to use the class “fifo”
10. Solution Explorer window -> Right-click “Source Files” -> “Add” -> “New Item” ->”C++ File”
11. Chose name “fifo” leading to new file “fifo.cpp” appearing within solution
12. Did COPY of all “includes” and the “using” declaration from original to fifo.cpp
13. Did Cut/Paste of all member functions from original to fifo.cpp
14. Builds OK. Have “implementation” file for fifo class, as well as new, very short main() user code.
15. In both fifo.h and fifo.cpp, placed my code (but not the includes/usings) into a namespace fifo_david (p. 721 Savitch)
16. In main() file, added a “using namespace fifo_david;” statement after the other “using” statement
17. Voila…