Exercises:

1. Learn about the "pimpl idiom". – What is the "pimpl idiom"?
   a) When should it be used? – What is its benefit then?
   b) How does it work?
   c) Create an example program using the pimpl idiom.
2. Refactor the program (without the pimpl idiom) using the UDTs and hierarchy *Engine*, *Tyre*, *Car*, *VintageCar* and *Bus* in order to Optimize the compile time dependencies as far as possible with the means we've discussed in the lecture!
3. Create a program, in which two types depend on each other mutually.
4. Learn how the exported symbols of an o-file can be dumped on your compiler environment OS. Document your findings with any o-file from the recent examples.

Remarks:

- Everything that was left unspecified can be solved as you prefer.
- In order to solve the exercises, only use known constructs, esp. the stuff you have learned in the lectures!
- **Please obey these rules for the time being:**
  - **The usage of goto, C++11 extensions, as well as #pragmas is not allowed.**
  - **The usage of global variables is not allowed.**
  - **You mustn't use the STL, because we did not yet understood how it works!**
  - **But *std::string*, *std::cout*, *std::cin* and belonging to manipulators can be used.**
- Only use classes for your UDTs. The usage of public fields is not allowed! The definition of inline member functions is only allowed, if mandatory!
- Your types should apply const-ness as far as possible. They should be const-correct. Minimize the usage of non-const&!
- The results of the programming exercises need to be runnable applications! All programs have to be implemented as console programs.
- The programs need to be robust, i.e. they should cope with erroneous input from the user.
- You should be able to describe your programs after implementation. Comments are mandatory.
- In documentations as well as in comments, strings or user interfaces make correct use of language (spelling and grammar)!
- Don't send binary files (e.g. the contents of debug/release folders) with your solutions! Do only send source and project files.
- Don't panic: In programming multiple solutions are possible.
- If you have problems use the Visual Studio help (F1) or the Xcode help, books and the internet primarily.
- Of course you can also ask colleagues; but it is of course always better, if you find a solution yourself.