Notes part 11

Professional C++. Ch. 11. Modules

- module interface files (pp. 399-401
- module implementation files (pp. 401-402)
- splitting interface and implementation (pp. 402-403)
- submodules (pp. 404-405)
- module partitions (pp. 405-408)
- old-fashioned ”Modules” (pp. 408-410)

A Tour of C++. Ch. 3. Modularity

- declare module with *export module*
- import instead of include – `import module foo;`
- module in `.cppm` *module definition files* eliminates need of separation in `.h` *header files* and `.cpp` *source code files*

– may be called `.cxx`, `.mpp` etc. depending on compiler
- error handling
- contracts
- assertions
- structured binding

Exploring C++20. Ch. 42. Modules

- hiding implementation
- compiling modules

Clean C++20. Ch. 6. Modularization

- information hiding
- strong cohesion
- loose coupling
- SRP – *Single Responsibility Principle* – similar to *Separation of Concerns* in database design
- SLA – *Single Level of Abstraction*
- OCP – *Open-Closed Principle*
- type erasure
- duck-typing
- LSP – *Liskov Substitution Principle*
- `final` specifier
- RTTI – *run-time type information/identification*
- ISP – *Interface Segregation Principle*
- DIP – *Dependency Inversion Principle*
- Law of Demeter – *don’t talk to strangers*

– *aspect-oriented software development*

– separate *interface* from hidden *implementatiton*
- avoid anemic classes
- avoid `static` class members
- BMI – *Built Module Interface* file