

Handle-Body Pattern Solutions

Interface/Implementation Separation

- Why is separating the interface of a class from its implementation important in object-oriented programming?
 - It prevents users of the class from accessing the implementation of the class
 - This reduces coupling between the users' code and the implementation code
 - e.g. if an implementation detail changes, the users do not need to modify their code

C++ class organization

- Why is it difficult to completely separate the interface and implementation of a C++ class?
 - Users of a C++ class need to include a header file which contains the class definition
 - This exposes the names of private and protected data members, the signatures of private and protected member functions, and the definition of inline functions
- What problems can this cause when compiling large applications?
 - If any changes are made to the header, all the users of the class need to recompile their code
 - This applies even if the change does not affect the client

The Handle-Body pattern

- What is the the Handle-Body pattern?
 - The class is divided into two parts
 - The Handle contains the interface (public member functions)
 - The Body contains the implementation of the class
 - The member functions of the Handle call the corresponding member functions of the Body