CSE687 Midterm #1

Name: ____Instructor's Solution___________ SUID: _____________

This is a closed book examination. Please place all your books on the floor beside you. You may keep one page of notes on your desktop in addition to this exam package. All Exams will be collected promptly at the end of the class period. Please be prepared to quickly hand in your examination at that time.

If you have any questions, please do not leave your seat. Raise your hand and I will come to your desk to discuss your question. I will answer all questions about the meaning of the wording of any question. I may choose not to answer other questions.

You will find it helpful to review all questions before beginning. All questions are given equal weight for grading, but not all questions have the same difficulty. Therefore, it is very much to your advantage to an

swer first those questions you believe to be the easiest.

1. State the Open-Closed Principle. Describe how you used it (or should have used it) in Project #1. Please be specific.

   Answer:

   OCP: Software entities (classes, packages, functions) should be open for extension, but closed for modification.

   This project requires the services of DirExplorerN, FileSystem, and Process – all available on the college server for student use.

   Because of the way DirExplorerN is structured, it has to be modified for application specific needs, but we can at least minimize the changes by having its file and directory handling functions simply save file names and paths, to be used by the other project packages[1]. An alternative would be to use DirExplorerT or DirExplorerI which can be used without modification by providing a Template parameter or derived class for application file and directory handling, e.g., extensions.

   The FileSystem has an effective interface and needs no modification for application specifics. Its functionality is wrapped by DirExplorer.

   The process class can be used without change by wrapping with a display class[2].

   All the other parts are newly created for this Project, and so do not qualify for use of OCP.

   See the answer to MT1-Q4 for good ways to design of OCP in both Project #1 and Project #2.

[1] You can't derive from DirExplorerN usefully, as it has no virtual functions.

[2] This ignores the bug in process that required you to move the callback definition from Process.h to Process.cpp. That is not a violation of OCP, it is a fix that renders the Process package truly reusable.

2. Write all the code for a container of an unspecified type. The container accepts instances of the type and retains up to the last N[3]. Please define an iterator for the container that starts at the latest element entered into the container and moves toward the oldest element in the container.

   Answer:

```
template <typename T>
class LastN {
public:
  using iterator = typename std::list<T>::iterator;
  using const_iterator = typename std::list<T>::const_iterator;

  LastN(size_t N) : N_(N) {}
  void add(const T& t)
  {
    // maintains last N with first at front
    list_.push_front(t);
    if (list_.size() > N_)
      list_.pop_back();
  }
  iterator begin() { return list_.begin(); }
  iterator end() { return list_.end(); }
private:
  std::list<T> list_;
  size_t N_;
};
```

[3] It retains all inputs until the Nth is entered.

3. What does the following code do? You may assume that all code shown compiles.

```
template<typename T>
T fun(const std::string& msg)
{
  return T(msg);
}

Widget w = fun<Widget>(“Hello”);
```

Answer:

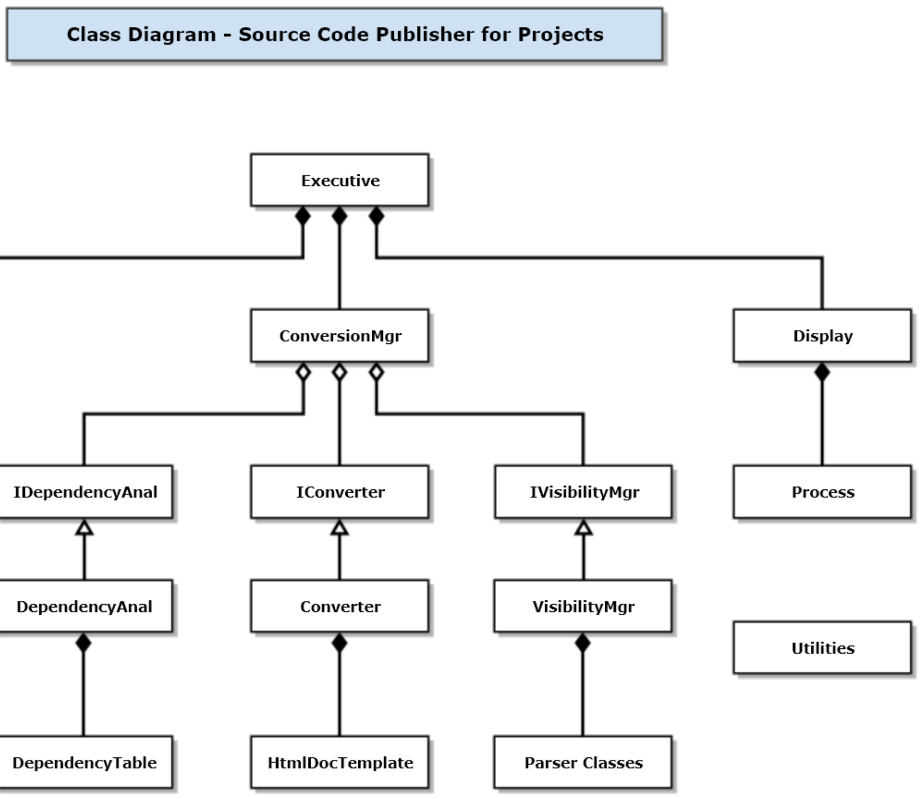
```
template<typename T>
T fun(const std::string& msg)
{
  return T(msg);
}


int main()
{
  std::cout << "\n  MT1-Q3 - what does this code do?";
  std::cout << "\n ----------------------------------\n";

  Widget d = fun<Widget>("Hello");
    // promotion constr of Widget, move construction of Widget
    // or copy construction if move not defined, or
    // return value optimization (RVO).
    // destruction of Widget
    // See MTCode/MT1-Q3 for details

  std::cout << "\n\n";
}
    // destruction of Widget
```

4. Draw a class diagram for your implementation of Project #2. You do not have to provide classes for the parser or any of its parts. Please guide your design by the Single Responsibility Principle.



5. Write all the code to create **a** child thread that accepts string message**s** safely from the main thread and writes **them** to the console.

Answer:

```
Note:
This question asks you to create ONE child thread and pass it MORE THAN ONE message.
```

```
void threadProc(BlockingQueue<std::string>& bQueue)
{
  while (true)  // no locking required as BQ is threadsafe
  {
    std::string msg = bQueue.deQ();
    std::cout << "\n  " << msg;
    if (msg == "quit")
    {
      std::cout << "\n  That's all folks\n\n";
      break;
    }
  }
}

int main()
{
  std::cout << "\n  MT1-Q5 - pass string messages between threads";
  std::cout << "\n -----------------------------------------------\n";

  BlockingQueue<std::string> bQueue;
  std::thread trd(threadProc, std::ref(bQueue));
  for (int i = 1; i < 11; ++i)
  {
    std::string msg = "msg #" + std::to_string(i);
    bQueue.enQ(msg);
  }
  bQueue.enQ("quit");
  trd.join();
  std::cout << "\n\n";
 }
```

6. What are the benefits of using inheritance? Identify places where you have used inheritance for your projects and state briefly why each was used (two or three places will suffice).

   Answer:

   Inheritance supports reuse of base class code and composed types in its derived classes. More important, though, is its use in helping us to build very flexible code through Liskov Substitution. Functions can accept parameters typed as base pointers or references, but will accept any pointer or reference to a class derived from base. This supports hiding distinctions between derived classes from clients, and allow the design to add new derived members without affecting clients.

   You probably did not use inheritance in Project #1 unless you used DirExplorerI or DirExplorerE. However, there are lots of opportunities to use Inheritance effectively for Project #2. Referring to my solution for MT4-Q4, above, we could use interfaces for the three principle activities: dependency analysis, conversion, and visibility control. If we decided later to webify C# or Java code, our #include-based analysis won't work and we would need to parse the code for its defined types. This is a major change, but with the IDependencyAnal interface, that won't affect anything else in the project. We simply substitute type-based analysis for #include-based analysis.

   Conversion of files is a fairly common implementation activity, and so the interface IConverter will allow us to take on other conversion processes without disturbing the rest of our code. Similar comments apply to the IVisibility interface.

   You also use Inheritance in a major way when you used the rule-based parser to implement visibility control.

7. Write a class declaration for a logger[4] that accepts string messages and asynchronously writes them to an output stream.

   Answer:
   I've generalized the solution by using templates, so one could log JSON or some structured message. That simply requires defining a suitable operator<< for type T. You were not required to do that.

```
template<typename T>
class Logger
{
public:

  Logger(std::ostream& out = std::cout);
  Logger(const Logger&) = delete;
  Logger& operator=(const Logger&) = delete;
  void log(const T& t);
  void wait();
private:
  std::ostream& out_;      // streams not copyable or assignable
  BlockingQueue<T> msgQ_; // BlockingQueue not copyable or assignable
  std::thread thrd_;
  void threadProc(BlockingQueue<T>& bQueue);
};
```

   See MTCode/MT1-Q7 for a complete implementation.

   Notes:
   - Writing a stream of messages asynchronously will get quite complicated if you don't use a thread-safe blocking queue.
   - Using async in write method is not a good idea. You do get asynchronous writes but spawn a new thread for each write. Better to use a single child thread to deQ messages and write them to the stream.

[4] This statement describes the logger. You don't need any further information.