

CSE687 Midterm #1

Name: _____ Instructor's Solution _____

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 examinations 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 answer first those questions you believe to be easiest.

1. How would you modify AbstractXmlElement and its derived classes, from the Project #2 helper code, to return the tag for any derived Element with a tag and do something sensible for elements without tags? Please describe how you would implement that – you don't need to provide code, but may if you wish.

Answer:

Add a method, `virtual std::string tag();` to the AbstractXmlElement class which returns an empty string. Override that method in the TaggedElement class to return the element's tag, saved in member `tag_;`

You don't need to modify any other class derived from AbstractXmlElement.

Note:

- a. No `dynamic_casts` are required for this implementation.
- b. This question has nothing to do with parsing.
- c. You were required to modify the base class so don't try to avoid that. Modifying a base class interface is fine until you have a lot of code that uses the base.
- d. This question has nothing to do with templates.

2. Write all the code for a Log class that accepts a `std::string` representing a log message, saves that in an in-memory data structure capable of holding multiple messages along with a time-date stamp for each message. Please save in arrival order. It must provide the capability to append its data to a log file and clear its internal store with a "save" command. You may assume that a "getTimeDate()" is available to you¹. Are there other member functions you need to supply?

Answer:

```
class Log
{
public:
    void message(const std::string& time, const std::string& text);
    void show();
    bool save();
    // compiler generated methods are correct
private:
    using Msg = std::pair < std::string, std::string >;
    std::vector<Msg> log_; // std::list<Msg> may have better performance
};

void Log::message(const std::string& time, const std::string& text)
{
    Msg msg;
    msg.first = time;
    msg.second = text;
    log_.push_back(msg);
}

void Log::show()
{
    std::cout << "\n Message Log";
    std::cout << "\n -----";
    for (auto msg : log_)
        std::cout << "\n " << msg.first << " " << msg.second;
    std::cout << "\n";
}

bool Log::save()
{
    {
        std::ofstream out("logfile.txt", std::ios::app);
        if (out.good())
        {
            for (auto msg : log_)
                out << "\n " << msg.first << " " << msg.second;
            out.close();
            log_.clear();
            return true;
        }
        return false;
    }
}
```

¹ If you don't know how to open for appending to an existing file you may write pseudo code for that operation.

3. When is it appropriate to let the compiler generate default versions of the default constructor, copy constructor, assignment operator, and destructor? How do you do that?

Answer:

You should let the compiler generate default versions of the cited functions if, and only if, the class's bases and member data have correct semantics for those operations. In that case, the compiler will generate code to invoke the operations on each of the bases and members in declaration order and the operations will be correct.

This is the case if your class and its bases only compose primitive data types, `std::shared_ptr<T>s`, STL containers holding correct² class instances or `std::shared_ptr<T>s`.

It is almost never true when your class holds native pointers to resources on the native heap. If your class holds `std::unique_ptr<T>s` it cannot be copy constructed or copy assigned. If your class holds C++ references to instances you must explicitly initialize them in a constructor initializer sequence and your class can not be copied or assigned (C++ references cannot be reset). If your class may be the base for another class you must provide a virtual destructor that may have an empty body (if there is no resource to explicitly release). If the conditions of the first paragraph are true for your class then you should let the compiler generate the other methods.

The best way, since the release of C++11, to use the default operations is to append the `"=default"` post qualifier to a public declaration of each operation. For compilers that don't meet the new standard you simply don't declare the operations (see note c., below).

Note:

- a. The compiler generated copy will do deep copy if the copy operations of the bases and member data do deep copy. This is the case for STL containers.
- b. The operations won't be generated if there is no application code that requires them.
- c. The compiler won't generate a default constructor unless no constructors are declared or you declare it using the `=default` post qualifier.
- d. The errors cited in the deminherit code examples are not due to compiler generated copy operations but due the designer's incomplete implementation of the copy operation.

² Correct as stated in the first paragraph.

4. State the Open/Closed Principle and describe how you have or intend to use it in Project #2.

Answer:

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

In Project #2 the AbstractXmlElement class defines a protocol for using all of the derived Element classes. We can extend the behavior of the operations provided by each derived Element class by overriding virtual methods of the base.

We can also provide new functionality by defining callable objects passed to a Depth First Search global function, void DFS(XmlDocument& doc, CallObj& co). Essentially we add new behaviors simply by defining new callable objects to perform operations on the document. None of the element or document code needs to change to acquire this new behavior.

```

////////////////////////////////////
// Global Functions for Depth First Search
//
// These functions take a callable object to define processing on each
// element encountered on search traversal. They may be functions,
// functors, or lambdas - see XmlDocument.cpp for examples.

//-----< search subtree of XmlDocument >-----

template<typename CallObj>
void DFS(XmlDocument::sPtr pElem, CallObj& co)
{
    using sPtr = XmlDocument::sPtr;
    co(*pElem);
    for (auto pChild : pElem->children())
        DFS(pChild, co);
}

//-----< search entire XmlDocument >-----

template<typename CallObj>
void DFS(XmlDocument& doc, CallObj& co)
{
    using sPtr = XmlDocument::sPtr;

    sPtr pDocElem = doc.docElement();
    DFS(pDocElem, co);
}

```

5. Write all the code for an interface and an object factory for the log class you created in Question #2.

Answer:

```
struct ILog
{
    virtual ~ILog() {}
    virtual void message(
        const std::string& time,
        const std::string& text)=0;
    virtual void show()=0;
    virtual bool save()=0;
};

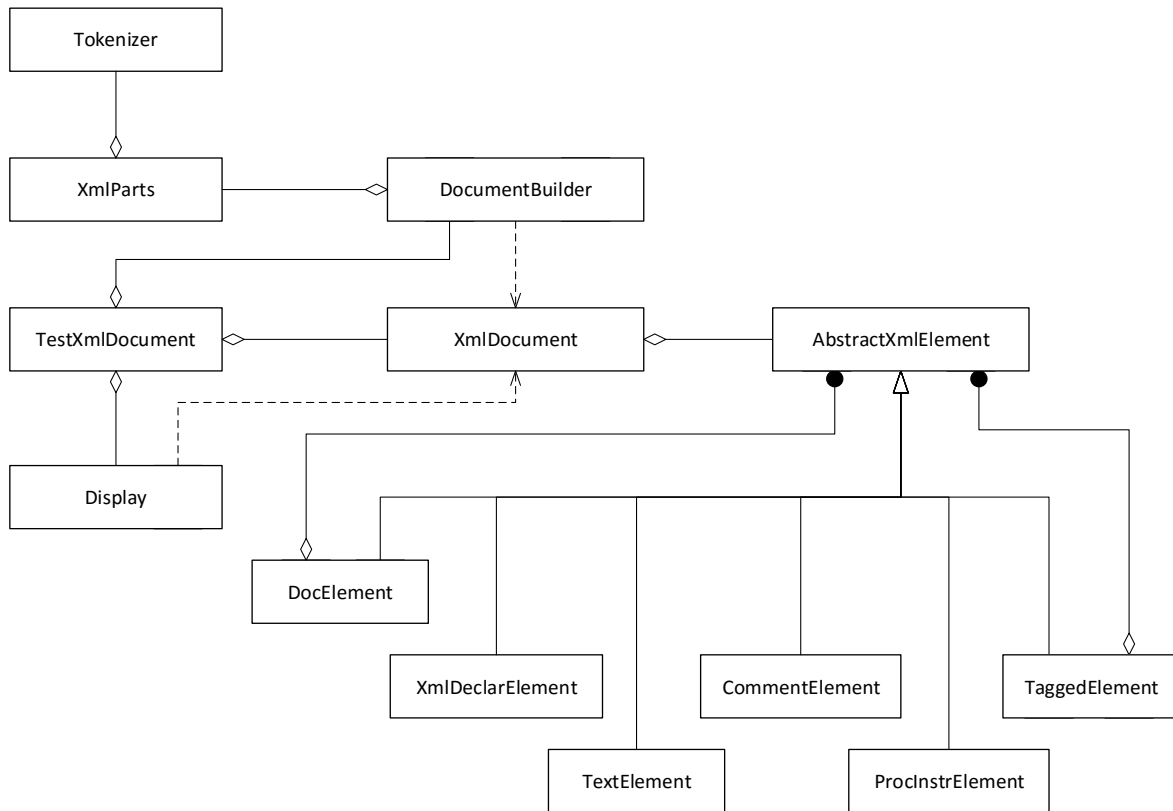
struct factory
{
    ILog* create();
};

class Log : public ILog    // derived class included to show context
{
    // not required for answer
public:
    void message(const std::string& time, const std::string& text);
    void show();
    bool save();
private:
    using Msg = std::pair < std::string, std::string >;
    std::vector<Msg> log_;
};

ILog* factory::create() { return new Log(); }
```

6. Draw a class diagram for your design of Project #2. Remember that you are obligated to demonstrate your project meets all the functional requirements in the project statement.

Answer:



7. List all the things you need to think about when designing something like project #2. Your answer should not be specific to Project #2 but should be consistent with the size and functionality of that project.

Answer:

Here are things to consider in essentially the order you would use to create the Project design³. This should focus on **Design** not **Implementation** details.

- a. **Tasks** required to meet requirements.
- b. The **order of processing** and any distinct phases needed to simplify its execution.
- c. **Structure of packages** to integrate all the tasks.
 - i. Here we think about the principles and patterns we may use, e.g., SRP⁴, LSP, OCP, DIP, ISP, and KISS⁵.
 - ii. We need to consider the boundry between application-side and solution-side processing⁶. For solution-side packages we think about reuse and flexibility. For application-side packages we want a domain model for each with language specific to its model⁷ and we want to ensure that the user does not need to know our solution-side details.
- d. How will packages **communicate** and what information will be passed?
 - i. Here's where we design the processing languages for each part of the system as defined by class interfaces.
- e. How will packages be populated with classes?
 - i. What class relationships are needed?
 - ii. How to manage object lifetimes?⁸ This is related to the order of processing, discussed above.
- f. What **data** must be processed and how that will be structured?
 - i. Now we're deep into the solution side, armed with the STL containers and algorithms, callable objects, and smart pointers.
- g. How to handle **errors** when using the file system and collecting user input?
 - i. Return errors or throw and catch exceptions or both.
- h. How to **format** the processed data to provide information to users about the processing and its results?
 - i. Here we decide on user control, filtering of data, and display formats.
 - ii. What data should be structured by the parts and what structured by the Display facility.

³ Using Principles discussed in class is a part of the answer, but a relatively small part.

⁴ Single Responsibility Principle

⁵ Keep It Small and Simple

⁶ Often only the Executive and Display may be application focused with the remaining being solution-side packages.

⁷ In Project #1 the domain model consists of commandline, text queries, patterns, and options. In Project #2 the domain model consists of Documents and Elements.

⁸ This is usually determined by ownership. An owned object can live as long as its owner.