MIDTERM EXAMINATIONS QUESTIONS.

1. **State the Single Responsibility Principle. How have you used it in project #2? Please be specific.**

Credits Received: 0.9

Mistakes:

Didn’t state the principle well as I tried to explain it my own words.

Corrections:

Each software entity: function, class, package, and module should only have one functionality. If an entity performs more than a single responsibility, it does not satisfy the principle.

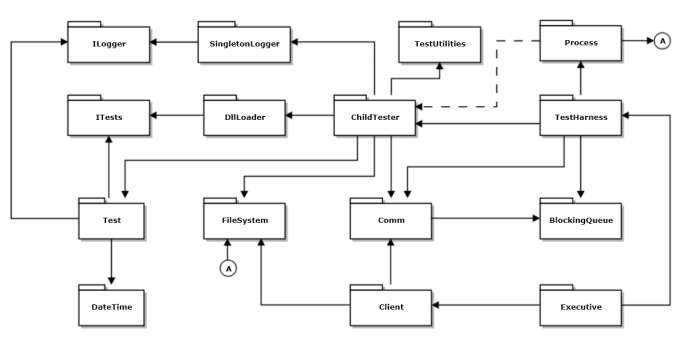
1. **Draw a package diagram for your design of project #3.**

Credits Received: 0.1

Mistakes:

Didn’t attempt the question. Understood the outline of Project 3 after the midterm examination.

Corrections:



This is the package diagram available from the Website. As I understood this is how the project 3 works:

* Comm Sends Test Requests that include the DLL’s to the TestHarness.
* TestHarness uses Blocking Queue to store the Requests.
* There is another Blocking queue of which maintains the TestExecutive Child Processes address.
* Every Request is then dequeued from the blocking queue and assigns the TestRequest to the TestExecutive which is dequeued from the second blocking queue.

1. **Assume that a project #3 test request consists of a list of one or more dynamic link libraries to load and execute, perhaps encoded in a string. Write a lambda that dispatches a test request message from one of the Process Pool’s Host queues to an appropriate child tester process. Now write a thread that, using the lambda, dispatches test requests messages as they arrive. You may treat the test request as a string without examining it’s details. For this question, I’ve attached a block diagram for Project #3, including a Process Pool, at the end of this exam packet.**

Credits Received: 0.3

Mistakes:

Didn’t answer relevantly to the question. Defined Threads, Priority Inversion, Deadlock, Lambdas and the functionality of Lambda with STL.

Corrections:

* Should define a lambda function which dispatches request Process pool’s host queues to a child tester process.
* Then a thread should be defined using the lambda and dispatch the test requests messages
* The whole process pool structure was clear from the diagram provided in the midterm examination.

1. **What are the main elements of Object Oriented Software Design, and how did you use them for Project #2?**

Credits Received: 0.89

Mistakes:

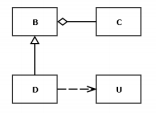
I got confused with the word Object Oriented Software Design in the question and gave just the elements of the software design.

Corrections:

Should have also mentioned the elements of Object Oriented design which are

* Encapsulation
* Class Relationships
* Principles

1. **Write a move constructor for the class D in the diagram given here. You may assume that all of the bases and members of each class are shown in the diagram. When will that method be called?**



Credits Received: 0.78

Mistakes:

Didn’t explain when the move constructor method will be called correctly.

Corrections:

The Move constructor will be called when any function returns the local instance of class D by value.

1. **Write a class declaration for a TestDataFormatter class. The formatter accepts a test driver name, author, date-time, test result, and a test specific instance of an unspecified type. It provides a method to return a formatted string1 for display purposes**.

Credits Received: 0.84

Mistakes:

Completely missed the term unspecified type which means a template from the question. Didn’t write the complete class declaration.

Corrections:

A method should be written to provide a formatted string for display purposes. This can be done by define a string as a formatted one first and then include an argument of a generic type which displays the string result.

Template<typename T>

Class TestDataFormatter

{

public:

void name(const std::string& testName); author(const std::string& authorName); timedate(const std::string& timeStamp); result(bool tResult);

private:

std::string namee;

std::string author;

std::string timedate;

bool result;

T t;

}

1. **When you define a class, under what conditions will you chose to implement copy, assignment, and destruction operations? When would you decide not to provide those? If you don’t, what happens if code using your class attempts to copy or assign instances?**

Credits Received: 0.82

Mistakes:

Failed to state correctly where we choose not to provide Copy Constructors and assignment operators.

Corrections:

Copy and assignment constructors cannot be defined when they do not match the correct construction semantics and also when they are not marked to be deleted.

MIDTERM MAKEUP EXAMINATION QUESTIONS

1. **Implement a thread safe unordered\_map that provides the same functionality as the std::unordered\_map, but supports safe read and write access by multiple concurrent threads.**

Solution:

#include <condition\_variable>

#include <mutex>

#include <thread>

#include <unordered\_map>

template<typename T, typename U>

class SafeUnorderedMap

{

public:

SafeUnorderedMap() { }

SafeUnorderedMap(const SafeUnorderedMap<T, U> && sumap) = delete;

SafeUnorderedMap<T, U>& operator=(const SafeUnorderedMap<T, U> && sumap) = delete;

SafeUnorderedMap(SafeUnorderedMap<T, U> && sumap)

{

std::lock\_guard<std::mutex> l(mtx\_);

umap\_ = sumap.umap\_;

//Clear the original umap

sumap.umap\_.clear();

}

SafeUnorderedMap<T, U>& operator=(SafeUnorderedMap<T, U> && sumap)

{

if (this == &sumap) return \*this;

std::lock\_guard<std::mutex> l(mtx\_);

umap\_ = sumap.umap\_;

//Clear the original umap

sumap.umap\_.clear();

return \*this;

}

U& operator[](T& key)

{

std::unique\_lock<std::mutex> l(mtx\_);

return umap\_[key];

}

void insert(T key, U val)

{

{

std::unique\_lock<std::mutex> l(mtx\_);

auto pair = std::make\_pair<T, U>(key, val);

umap\_.insert(pair);

}

cv\_.notify\_one();

}

U get\_value(T key)

{

std::unique\_lock<std::mutex> l(mtx\_);

return umap\_[key];

}

void clear()

{

std::lock\_guard<std::mutex> l(mtx\_);

umap\_.clear();

}

size\_t size()

{

std::lock\_guard<std::mutex> l(mtx\_);

return umap\_.size();

}

private:

std::unordered\_map<T, U> umap\_;

std::mutex mtx\_;

std::condition\_variable cv\_;

};

1. **Describe where you have used the five main principles discussed in class, SRP, LSP, OCP, DIP, and ISP, in your design of Project #4.**

Solution:

**SRP(Single Responsibility Principle):**

The main use of the SRP is to make the comprehension and the design of the project easier. In project #4, we have designed different packages, classes and functions solely with a particular responsibility.

List of packages and their responsibilities:

TestDrivers: Defines the tests to be performed (two test drivers are included in the project again for the purpose of separating out the driver with a single test from the driver having multiple tests.)

DllLoader: For loading the TestDriver Dlls’ if the follow the ITest Protocol, else report an error.

TestRequest: Parsing the TestRequests received through the Comm Channel

Translater: Convert managed C# messages to native C++ messages. Message: Defines the structure of Message that is to be sent over the Comm Channel. All the native C++ message follows a particular structure that is defined in this package.

MessagePassingComm: Established a communication channel for passing the TestRequests and results between Client, TestHarnessCore and the Child Processes. Contains three classes: Sender (supports connecting to multiple endpoints by sending connection requests and sending messages), Receiver (returns a Socket when connection is established), Comm (defines methods for sending and receiving messages along with composing Sender and Receiver objects)

Sockets: Contains four classes that helps the Socket Programming: Sockets (for handling Server-Client Architecture), SocketConnecter (for connecting to a server), SocketListner (wait till the connection request comes to accept that request if the request is good), Socket System (loading and unloading WinSock library)

SingletonLogger: Supports logging facility

TestHarnessCore: contains TestUtilities class which defines the process of test execution on the TestRequests and Tester Class to load and extract tests and execute them following the TestUtilities infrastructure

CommPeers: two CommPeers are defined, one of which acts as a mother of the TestHarnessCore and other acts as a Child and follows the MessagePassingComm architecture for communicating.

GUI: Provides the interface to perform tests on selected Drivers.

Process: Handling the ProcessPool functionality

Thus, I’m illustrating the structure of my Project by the distributed functionality of it’s packages and classes which follows the SRP. Each package has it’s own independent functionality which makes the design less complex.

**LSP(Liskov Substitution Principle):**

It states that functions that use pointers or references statically typed to some base class must be able to use objects of classes derived from the base through those pointers or references, without any knowledge specialized to the derived classes.

**OCP(Open Closed Principle):**

It states that the Software Entities should be open for extension but closed for modification.

In my design of Project #4, this principle has been used at many spots: TestUtilities: Contains the infrastructure of executing the tests sequentially, then in the next project, extended it to make the tests follow the ITest interface. MessagePassingComm and Translater: follows native message and CSMessage respectively and in the previous design, there was only the native message which was managed by MessagePassingComm.

**DIP(Dependency Inversion Principle):**

It states that the higher level designs should not depend upon the lower level designs and instead both should depend upon some common abstractions. An example of this which is used in Project #4 is that the architecture of the TestDrivers does not depend upon the TestExecution, i.e. TestUtilities package and instead, both of them depend upon the common infrastructure of ITest Interface.

**ISP(Interface Segregation Principle):**

It states that the Client should not be dependent upon the interfaces that they do not use.

In Project #4, an interface for the client was built for him/her to select the Tests that he/she wants to be performed. Apart from that, the inner complexities were not revealed to him/her and simply by clicking on the name of the DLL, the test results were shown. The client does not have to know about the complex infrastructure.

Also, when many TestRequests pile up from different Clients, the TestHarnessCore selects every free Child process to execute the TestResults and return the results to the client. This design also took care of which client sends which requests and sending the result to the proper client back by ProcessPool, proper Message Passing Communication Channel and QueueManagement.

1. **Please provide the definition of a Message dispatcher used to process messages received by the remote analysis server of Project #4. Provide at least one lambda, bound to a message command by the dispatcher.**

Solution:

The concept of message dispatcher was not used in project #4 nor do we have a remote analysis server.

1. **Explain what each of the initialization sequences is doing in the Compound Objects demo in Handouts/Repository/Cpp/CompoundObjects. Please do that for all the classes with initialization sequences in their code.**

Solution:

Class C:

Class C has no initialization sequences.

Class B:

* B(const B& b) : c(b.c)

{

putMsg(“B copy construction”);

}

— Calls copy constructor of Class C when this is executed because B and C have Composition relationship.

* B(B&& b) : c(std::move(b.c))

{

putMsg(“B move construction”);

}

— Calls move constructor of Class C when executed for the same reason stated above as B is composed of C.

Class U:

Class U has no initialization sequences.

Class D:

* D(const D& d) : B(d)

{

putMsg(“D copy construction”);

}

* D(D&& d) : B(std::move(d))

{

putMsg(“B move construction”);

}

— Again, B(std::move(d)) can be referred as passing an object of class B, i.e. d, as an object of Class D is also an object of Class B. Hence, the move constructor of Class B is called which calls the move constructor of Class C.

This is because, when any construction of D is called, it immediately calls B’s construction because D is derived from B. D’s construction doesn’t complete until B’s does. And when B’s construction is called, it calls C’s construction and doesn’t get completed until B’s does.

1. **Why has the Cpp11-BlockingQueue made its copy operations =delete?. You can find that code in Handouts/Repository/Cpp/Cpp11-BlockingQueue.**

Solution:

Cpp11-BlockingQueue is a Thread-safe implementation which is essential as it shares its resources across multiple threads. Hence the copy constructor needs to be deleted so that the compiler does not create a default copy constructor.

This is because if a copy constructor is created by an compiler, then the nature of the Cpp11-BlockingQueue which is Thread-Safe is then compromised.

1. **Given the code fragments below, where the ellipsis (…) represents code that has not been disclosed to you:**

**class X { … }; class Y { public: explicit Y(const X& x); … };**

**What can you say about the compilation and execution of each of the following statements? Describe each of the operations that occur as this code executes.**

**Y func(Y y) { … }**

**X x;**

**Y y = func(Y(x));**

Solution:

* Here explicit keyword is used in the promotion constructor of Y, which is taking argument of constant reference type of X. Since the third statement is calling function Y which is expecting argument of type Y explicitly Y’s promotion constructor is required to be invoked. Otherwise passing x inside function will fail compilation because of explicit keyword.
* X x is creating default construction of X.
* Y(x) is creating a temporary object of type y which is passed by value to the function. Here Y’s copy and move construction code is not shown. So dependent on the rest of the code, either copy or move construction will be invoked. Finally a copy construction will be invoked of Y from the returned object of type Y returned from function with Y y=func.

1. **Suppose that objects of a class are required to correctly carry out copy and assignment operations. Support or refute the claim that the class must implement a copy constructor and assignment operator. You must consider all the cases (what are they?).**

Solution:

Copy constructor is used to construct an object with same property values of another object.

Certain type of class objects if copied two instances with same property values can cause error in the system they belong to. For example, socket class objects (once starts listening on certain port) if they are allowed to be copied new copy constructed socket object will try to listen on same port number. And that will result in an error or exception.

1. **Use the std::list class to hold a collection of strings. Use algorithm(s) from the STL and a lambda to concatenate the strings**.

Solution:

#include<list>

#include <string>

#include <iostream>

void main()

{

std::list<std::string> strings;

strings.push\_back("Raj");

strings.push\_back("R");

auto Final = [&](std::list<std::string>)

{

std::string tempStr;

for (auto& x : strings)

tempStr.append(x);

return tempStr;

}

std::cout << Final(strings) <<"\n\n";

}

1. **What is the purpose of defining both an interface and an abstract class in an inheritance hierarcy?**

Solution:

An abstract class should be used when some common implementation is required. An interface would be needed if you we want to specify a contract that parts of the program have to confirm to. By implementing an interface we are guaranteeing that we will implement certain methods while by extending an abstract class we are inheriting some of its implementation. Therefore an interface and abstract class both are required in an inheritance hierarchy.

1. **Given the declaration:**

**void X::mf(Y &y);**

**How would you change the declaration to declare that the operation of this function will not change the state of the object invoking it? What can you do with the re-declared function that you could not do in its original form?**

Solution:

void X : : mf(Y &y) const ;

This declaration implies that we do not intend to change the calling parameters of the function. Hence object used to call this function will not be modified.

The declaration having a const, ensures that all the parameters remain constant and are not modified.