CSE687 Midterm #2

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 answer first those questions you believe to be the easiest.

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

   Answer:

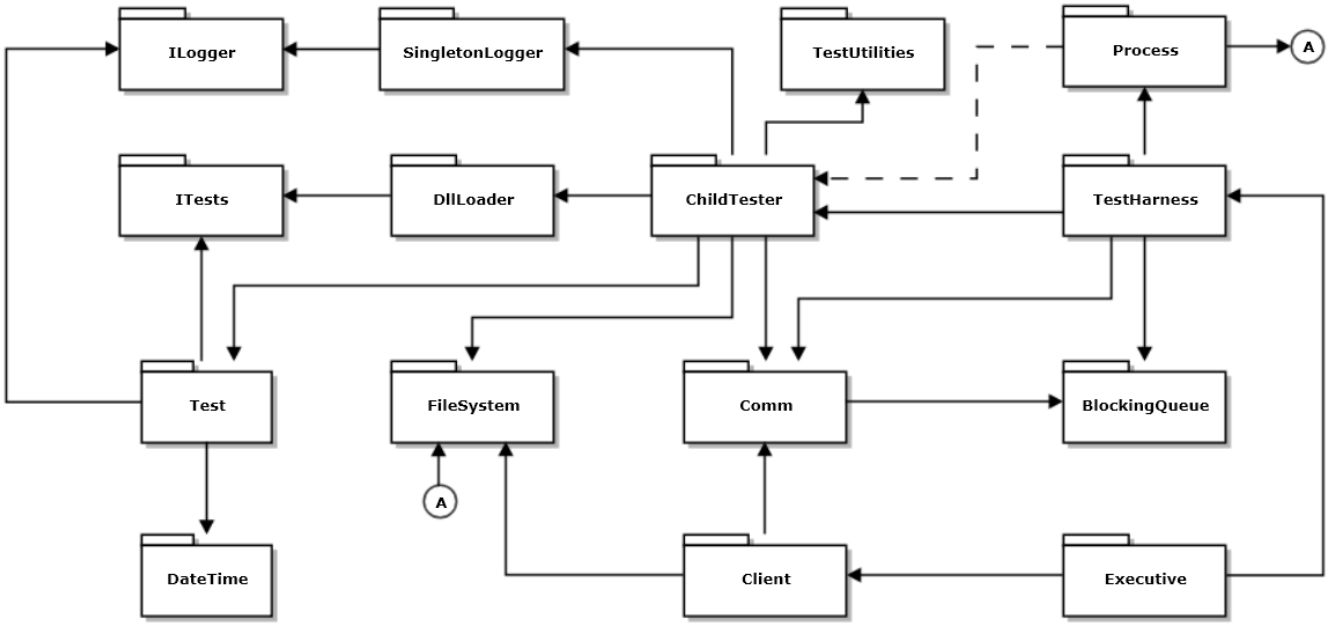
   Each software entity: function, class, package, and module should have one responsibility. If you can't state the responsibility in a few words, you probably are not satisfying the principle.

   Yes.

   a. Client creates TestRequest and stores in requestPath.
   b. DllLoader loads dlls and extracts their tests.
   c. Tester executes tests extracted by DllLoader
   d. TestRequest creates and parses test requests
   e. TestDriver1 and TestDriver2 define tests to be run by tester.
   f. SingletonLogger sends log messages to possibly several streams
   g. FileSystem makes directory information accessible to C++ programs.
   h. DateTime represents clock time.
   i. CodeUtilities provides helpers for low-level code operations.
   j. StringUtilities provides helpers for handling strings.
   k. TestUtilities provides framework for single user testing.

2. Draw a package diagram for your design of Project #3.

   Answer:



3. 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.

Answer:

```
Pseudo Code:  From Process Pool Diagram
while(true)
  deQ test request message
  deQ ready message
  send request message to from_address
  of ready message
```

```
///////////////////////////////////////////
// MT2Q3: Part A of answer

void TestHarness::dispatchMessages()
{
  while (true)
  {
    Message trMsg = requestQ_.deQ();
    std::cout << "\n    TestHnDsp deQ msg: " + trMsg.command();

    Message rdyMsg = readyQ_.deQ();
    std::cout << "\n    TestHnDsp deQ msg: " + rdyMsg.command();

    trMsg.to(rdyMsg.from());  // send request to ready child
    std::cout << "\n    TestHnDsp sending msg: " + trMsg.command();
    comm_.postMessage(trMsg);
  }
}
// MT2Q3: End of Part A of answer
///////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////
// MT2Q3: Part B of answer

std::cout << "\n    starting TestHarness dispatch thread";
dspat = std::thread([&]() { dispatchMessages(); });
dspat.detach();

// MT2Q3: End of Part B of answer
///////////////////////////////////////////////////////////////////
```

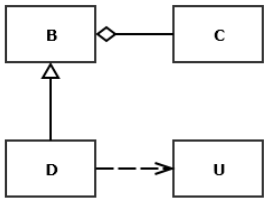
4. What are the main elements of Object Oriented Software Design, and how did you use them for Project #2?

   Answer:

   Design of software using classes and class relationships: inheritance, composition, aggregation, and using, to provide a software structure.
   a. Inheritance supports sharing of code between base and derived classes, and most important, Liskov Substitution "functions that accept pointers and references to base classes must accept pointers and references bound to derived class objects without using any knnowledge specialized to those derived classes.
   b. Composition allows a composing class to contain instances of other types that provide help in implementing the classes' responsibilities. This decomposes classes into parts each of which should support the Single Responsiblity Principle.
   c. Aggregation allows an aggregating class to create instances of other classes, as needed, and bind them to member pointers. It serves a purpose similar to composition, but supporting dynamic creation and lazy loading.
   d. Using allows a using class to access and employ instances of classes created by some other entity.

   In Project #2:
   a. Test drivers implemented the ITest interface, allowing the DllLoader to contain a collection of tests via Liskov Substitution.
   b. The tester composed an instance of DllLoader.
   c. TestCollection holds a vector of pointers to tests created on the native heap, e.g., aggregation.
   d. Interface ITest's acceptHostedResource(ILog* pRes) supports use of the Tester's logger.

5. 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?



Answer:

```
D(D&& d) : B(std::move(d))
{
  std::cout << "\n  D moved constructed";
}
```

Note this uses the B move constructor to take care of C. U is simply used by D, so should not be moved.

D's move constructor will be called when an instance of D is return by value from a function that created the local instance of returned D or when we explicitly invoke it as here:

```
D d1;
D d2 = std::move(d1);  // move construction may invalidate d1
```

6. 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 string[1] for display purposes.

   Answer:

```
template<typename AppType>
class TestFormatter {
public:
  void name(const std::string& name);
  void author(const std::string& author);
  void date(const std::string& date);
  void result(bool rslt);
  void appType(const AppType& appType);
  std::string format();
  std::string format(ITestResult<AppType>* pTestResult);
private:
  std::string name_;
  std::string author_;
  std::string date_;
  bool result_;
  AppType appType_;  // will use to hold log items
};
```

[1] You may assume that the unspecified type has a toString method. The details of the format are not important for this question.

7. 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?

   Answer:

   You choose to implement copy, assignment, and destruction operations only if the class bases and members do not have correct copy, assignment, and destruction semantics.

   You may also choose not to provide them for the case cited above, but must then qualify the copy and assignment operations with = delete;

   If you don't provide those operations:

   a. If copy and assignment are not marked delete, then the compiler will generate them if needed by doing base-wise and member-wise copy and assignment.
   b. If copy and assignment are marked delete, implied copy and/or assignment operations will fail to compile.

   Examples of these rules:

   c. If the class has no bases and its data members are all primitive types and/or STL containers, then their copy, assignment, and destruction semantics are correct. So you should choose to let the compiler generate these operations.
   d. If the class has a member pointer to an instance allocated on the heap, then you must provide them or qualify copy and assignment as = delete. In this case you have to define the destructor to avoid a resource leak.