

# CS249 Final Exam: Closed Book, 180 Minutes

Instructor: David Cheriton, Monday, March 17, 2003

December 3, 2007

This is a CLOSED-BOOK exam. You are expected to work on your own to complete this exam, not using other people or any materials for your assistance.

- Please answer all questions; read each question carefully. You waste time by writing more than required by the question, and you lose points by answering the wrong question.
- Please give precise, specific answers that demonstrate the depth of your knowledge of the area being examined by the question, rather than vague and general comments. Point form answers are acceptable. You can use sketch code that is not necessarily complete AS LONG AS IT CLEARLY INDICATES you understand the approach or technique we are after.
- Please put your name and student number on each exam booklet you use and sign the honor code statement.

1. **(18 Points)** Define each of the following terms as used in the course and briefly describe the key reason this term/concept is useful
  - (a) generic accessory type
  - (b) notification
  - (c) value type
2. **18 Points** Cheriton claims that the attribute-only interface significantly reduces the *slice* that a new programmer has to learn in a software system designed using this approach.
  - (a) Describe why this claim is valid.
  - (b) How does the notification model of events benefit from the attribute interface model, if it does.
  - (c) Smart-boy Fred claims to have developed a major improvement in what he calls "generic attributes" to further reduce software complexity, i.e. attributes that are used across many different classes. Describe whether this is a useful concept and to what degree this course already presents and uses a number of generic attributes.
3. **18 Points** In C++, it seems necessary, at first, to enclose every function call in a try block, and even each catch statement in a try block yet Cheriton claims it is feasible to write many functions with no try blocks, and in fact, a try block may be selectively added later as an optimization.
  - (a) Describe the techniques/approach involved to allow many functions to be written with no try blocks.
  - (b) Why is it not feasible to say "write all functions with no try blocks". I.e. in which functions does having a try block seem essential?
  - (c) Briefly describe a function in which one might add a try block as an optimization.
4. **18 Points** Cheriton makes the outrageous claim that the `Ptr` class template is basically the only form of smart pointer required, yet others have designed smart pointer classes specifically for pointing at persistent and remote objects.

- (a) Describe how to deal with these cases using `Ptr`.
  - (b) Briefly illustrate the problems with having extra smart pointer classes.
  - (c) If the C++ community agreed that `Ptr` is all you basically need, what are the key benefits of it still being defined in a class template as opposed to being directly built into the language/compiler.
5. **18 Points** JLo received a copy of the CS249 reader from Ben Affleck and laughed hysterically after reading the chapter on manager, directories and modules. "How did this clown reason his way from good old C++ constructors all the way to using an abstract framework interface class `Directory` and calling a generic function `Directory::newInterface` with a couple of string parameters to create objects?", she exclaimed? "You don't even get the type of the created object returned to you". Describe the reasoning behind this approach to JLo, focusing on the key points.
6. **18 Points** It's nice to think about shared descriptions, but if you were really trying to write a flight simulator, no two airplanes would have the same description, simply because they would be in different places, going in different directions, with different rudder angles, etc.
- (a) Describe how we can share a description with another airplane in this world where they disagree on some of the attributes of the description that they supposedly share?
  - (b) Why are there complications with allowing writing to a Named Description?
  - (c) Describe two techniques which may be used to fix the problems from part (b).
7. **18 Points** In class we suggested that all inter-object relationships could be modeled as if they were dynamic collisions between objects of some geometry (for a fairly liberal definition of geometry).
- (a) Why does the collision model of inter-object relationships make reference counting unnecessary (or at least, much simpler)?
  - (b) What is the main advantage that the collision model of inter-object relationships gives us?
  - (c) What is the most substantial penalty?
8. **18 Points** We argued that you should break objects of substantial size down into smaller "component" objects. Your friend (who eats assembly code for breakfast) tells you that this is all very well and nice for little toy programming assignments, but that in a real program:
- (a) Calling down into component objects induces unnecessary runtime overhead, making your program run slowly.
  - (b) Each component of an object needs a separate set of "wiring code" to tie it into the larger whole, leading to code bloat.
  - (c) Having components makes it impossible for new programmers to learn the code because they have to dig down into component after component to get to the code which is actually doing the interesting work.

Refute these arguments by giving three reasons why component-oriented design is the right way to do things. (Describing a technique that we use which invalidates one of his arguments may also be used as a reason).

9. **18 points** Saddam Hussein states that Oil "isAKindOf" Currency (because he can exchange it for weapons). He also states that Oil "isAKindOf" Fuel. Therefore, he has the Republican guard using not just inheritance but multiple inheritance with these classes in their C++ programs. As part of the reconstruction of Iraq, your job is to get them to reduce their dependency on inheritance. Describe how you would structure the relationship between Oil, Currency and Fuel to avoid inheritance (sketching the relevant portions of these classes if necessary) and how you would convince the guard that this approach was better.

10. **18 points** List three key benefits of application-specific value types (ADTs) and describe how generic framework mechanisms can support defining these types while minimizing the amount of code required. (You can illustrate with a C++ type definition or describe in words.)

The End, Have a good spring break!