

# CS 249 MidTerm Exam - Closed Book In-Class Exam

Time allotted: 75 minutes February 8, 2000

Answer all questions. Each question is worth the points as indicated. You can answer in point form if you prefer when English is called for. 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. We are not looking for a full essay on each question, but rather a short concise set of points or sketch code that responds to the question and indicates you understand the point being explored.

- (10 Points)** Describe how each of the following contribute to the SOS methodology:
  - Modeling and simulation perspective.
  - Separating interface from implementation
  - Attribute-only interface design
- (10 Points)** With interfaces classes,
  - Give three reasons why we derive them from the class `Interface`.
  - Describe why you should minimize, but not preclude, implementation in interface classes. Illustrate with an example.
  - Why should you minimize in general the height of the interface class type hierarchy?
- (10 Points)** Al Gore, in a campaign speech, surprisingly proposed to add a conversion operator to the `Ptr`, supporting automatic conversion to the raw pointer. George W. Bush immediately issued a press release saying this is a bad move because it allows ordinary client code to get at the raw pointer.
  - Explain the disadvantages of Gore's proposal, with a compelling example.
  - Why is Bush's objection not valid.
- (10 Points)** Consider an manager class for some application entities that is implementing and using the `Notifieee` interface of the `MemMan` instance it is accessing for memory allocation.
  - What is the sequence of actions that occurs when the application-specific manager is destroyed, focusing on the notification interface and its coupling to the memory manager, and how are they implemented.
  - What are the problems that have arisen with handling callback interfaces of this nature in other implementations?
- (10 Points)** Britney Spears wanted a new challenge, and joined your programming team, writing the following interface class for a simulation of a computer that your team is doing.

```
class Computer : public Interface {  
    . . .  
    virtual void reboot() = 0;  
    . . .  
};
```

Assuming you, as manager, buy completely into attribute-only interface design, how do you tell Britney to change this class and what are the reasons you give, in brief.

- (10 Points)** Consider the following sketch of some exception-using code.

```

class AirplaneException {
    ...
};
void AirplaneImpl::thrust( int i ) {

    if( ... ) throw AirplaneException;
    if( ... ) throw int( 37 );
    adjustThrust( i );
    updatePlaneState();
}

FlightSim::do() {
    try {
        . . .
        a->thrust( newThrust );
        . . .

    }
    catch( int& i ) {
        . . .
    }
    catch( AirplaneException& ae ) {
        . . .
    }
}

```

What is wrong with the exception-handling aspect of this code with respect to the approach described in the course? (You can make reasonable assumptions about missing aspects of the code that you; just state them.)

The End