Student ID___________________Student Name______________________________________

Advanced Software Development DE Midterm Exam June 25, 2011

PRIVATE AND CONFIDENTIAL

1. **Allotted exam duration is 2 hours.**
2. **Closed book/notes or open book/notes.**
3. **No personal items including electronic devices (cell phones, computers, calculators, PDAs).**
4. **Cell phones must be turned in to your proctor before beginning exam.**
5. **No additional papers are allowed. Sufficient blank paper is included in the exam packet.**
6. **Exams are copyrighted and may not be copied or transferred.**
7. **Restroom and other personal breaks are not permitted.**
8. **Total exam including questions and scratch paper must be returned to the proctor.**

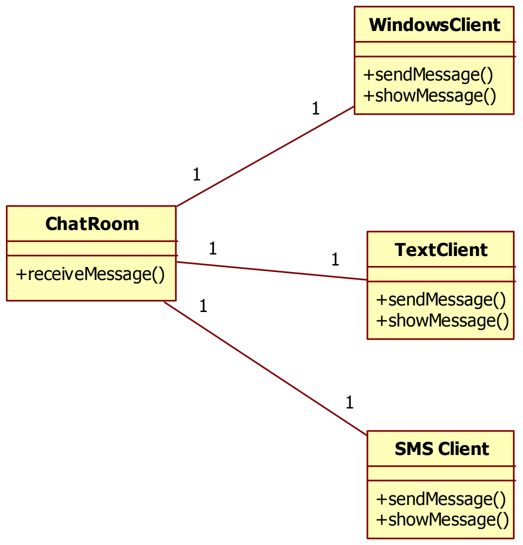
8 blank pages are provided for writing the solutions and/or scratch paper. All 8 pages must be handed in with the exam

BE VERY CAREFUL WITH THE GIVEN 2 HOURS AND USE YOUR TIME WISELY. THE ALLOTED TIME IS GIVEN FOR EVERY QUESTION.

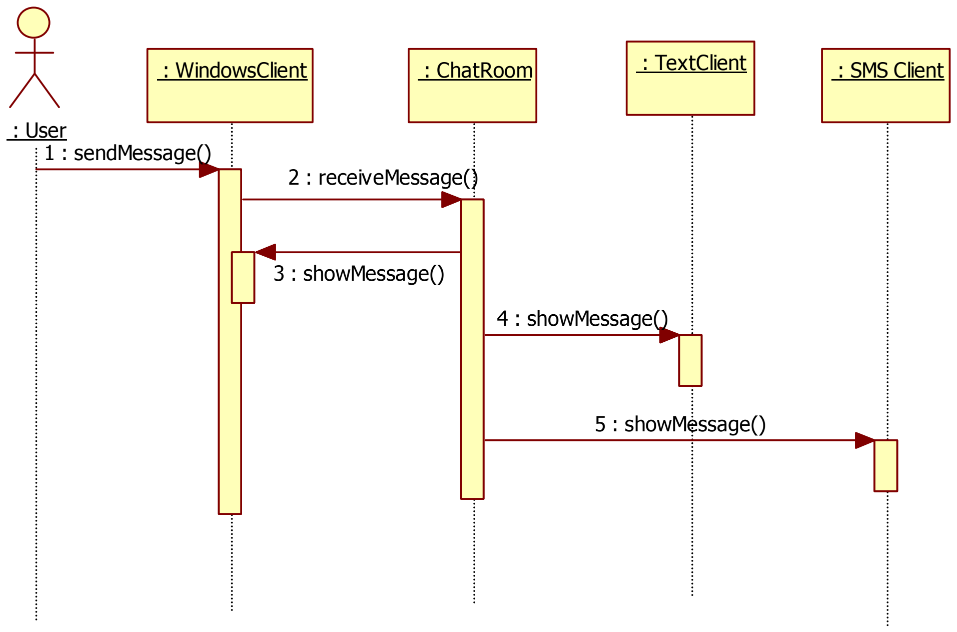
Write your name and student id at the top of this page.

Question 1 [15 points] {15 minutes}

Suppose our current chat application contains the following design:



We have 3 types of client user interfaces, a windows client UI, a text client UI and a SMS Client UI. When we send a message from one of the clients to the ChatRoom, then this message will be shown on all Clients.



Suppose we want to add more chat clients, for example a Flash chat client or a more advanced Windows chat client. If we want to add another chat client to our current chat application, then we have to modify the receiveMessage() method of the ChatRoom class so that it also sends the messages to this new chat client

Redesign the chat application so that we can easily add new chat clients without modifying the ChatRoom class.

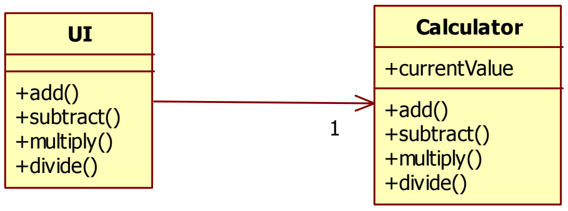
Draw a class diagram of your design. **Make sure you add all necessary UML elements (attributes, methods, multiplicity, etc) to communicate the important parts of your design.**

Question 2 [15 points] {15 minutes}

Suppose we have the following calculator class:

```
public class Calculator {
      private int currentValue;

      public int add(int x){
            currentValue=currentValue+x;
            return currentValue;
      }
      public int subtract(int x){
            currentValue=currentValue-x;
            return currentValue;
      }
      public int multiply(int x){
            currentValue=currentValue*x;
            return currentValue;
      }
      public int divide(int x){
            currentValue=currentValue/x;
            return currentValue;
      }
}
```



A Windows UI class allows us to use this calculator. The UI shows the currentValue of the Calculator. We can fill in the value x, and then click one of the buttons add, subtract, multiply or divide. The UI will then show the new currentValue of the Calculator.

Now we want to add an undo and a redo button to this application. Draw the class diagram of the design that adds undo or redo functionality to this application.

Question 3 [20 points] {25 minutes}

For a Hotel room reservation system we have to design the logic for calculating the room price. Every room in the hotel has a base price. The price that the customer has to pay is the base price minus a certain discount. The computation of the discount depends on a number of properties:

1. Every season you get a different discount
2. Every payment option gives different discount
3. Different types of customers get different discount
4. Customers who are member of the Hotel Rewards Program get a base discount plus an additional discount depending on their customer status

To give you a better understanding, suppose we give the following discount:

Season:

Summer	0.5%
Autumn	1.5%
Winter	1%
Spring	2%

Payment

Mastercard/Visacard	7%
American Express	3%
Check	0.5%

Customer type

Business	4%
Non-business	2%
Hotel reward member base discount	1%
Hotel reward member with status Silver additional discount	5%
Hotel reward member with status Gold additional discount	10%
Hotel reward member with status Platinum additional discount	20%

If a business person wants to have a room in the spring and he pays with Mastercard, he gets 4% (business) + 2% (spring) + 7% (Mastercard) = 13% discount.
Design the discount calculator module with the following requirements:

1. It should be easy to add more seasons, for example early summer and late summer
2. It should be easy to add more payment types
3. It should be easy to add more customer types
4. It should be easy to add more states for the members of the hotel rewards program
5. It should be easy to add other discount options.
6. In the given discount example I show a certain fixed percentage. In reality the discount is calculated with a certain algorithm. For example, if you pay with American Express, and the room base price is lower than $100, you get 2.5% discount, and if the room price is higher than $100, you get 3.5% discount.

a. Draw the class diagram of your design of the discount calculator.
b. Draw a sequence diagram that shows the correct working of your design. The sequence diagram should show the scenario where we calculate the discount of Hotel Reward Member with status Gold that wants a room in the Spring and pays with a check.

Make sure you add all necessary UML elements (attributes, methods, multiplicity, etc) to communicate the important parts of your design.

Question 4 [10 points] {10 minutes}

Suppose I have a list of accounts: CheckingAccounts, SavingAccounts and BusinessAccounts.
Twice a year I need to add interest to all accounts in the list. The calculation of the interest is a complex algorithm that is different for every different account type. The code for the current accounts does not contain the functionality to compute and add the interest. So I decided to write an InterestVisitor class that visits all Accounts in the list, and adds the correct interest to all accounts.
Once a year I also have to charge the accounts for the use of ATM cards. I write a new CardChargerVisitor that charges every account the correct amount. In time I extend the functionality of my bank account system with new Visitors: StatementPrinterVisistor, CheckAccountForUnusualBehaviorVisitor, etc. I think I have made the right choice to apply the Visitor pattern, because I can add behavior to my accounts without changing the account classes.

Explain clearly if I made the right choice to apply the Visitor pattern in this situation. Describe under what circumstance the Visitor pattern is a good choice, and under what circumstance it is not a good choice to use the Visitor pattern.

Question 5 [10 points] {15 minutes}

Suppose a certain device can send 45 different messages to an application that need to handle these messages. Every message contains a certain number between 1 and 45 that corresponds to the type of message. The receiver object that receives these messages has to make sure the correct logic is executed based on the message number. For every message with a different number, some different logic needs to be executed.
It is important that the design of the application allows us to easily add more messages (for example 100 different messages) with the least amount of code change.

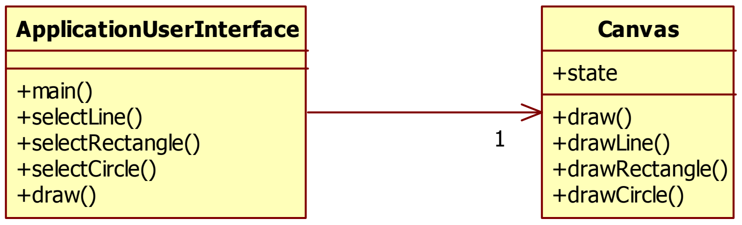
a. What pattern would you apply in your design?
b. Draw the class diagram of your design.

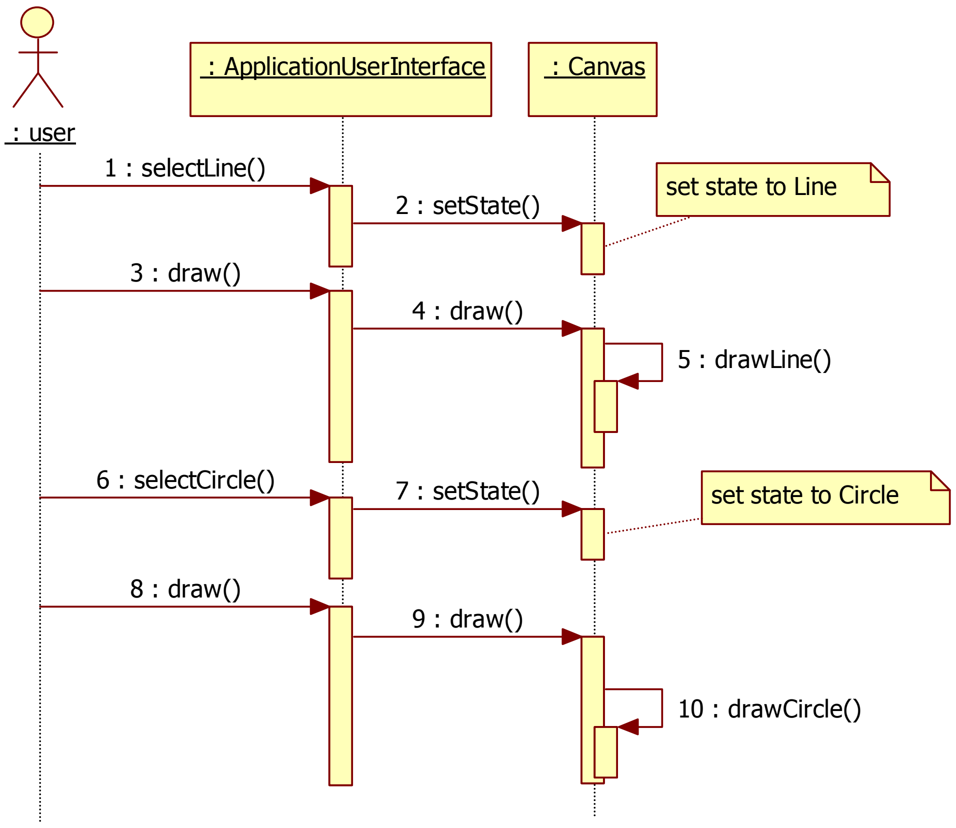
Question 6 [5 points] {10 minutes}

Describe how we can relate the Composite pattern to one or more principles of SCI. Your answer should be about half a page, but should not exceed one page (handwritten). The number of points you get for this questions depends how well you explain the relationship between the Composite pattern and one or more principles of SCI.

Question 7 [25 points] {30 minutes}

Suppose we have to design and write a simple drawing program. With this simple drawing program we can draw lines, rectangles and circles. The program we write has the following design:





The problem with our current application is that whenever we add new shapes to our drawing application, we have to modify the canvas class. Another issue is that the draw() method of the canvas class is a large if-then-else statement where it is easy to make mistakes.

a. Redesign our drawing application such that it will be easy to add new shapes to our application. Draw the new **class diagram**.
Make sure you add all necessary UML elements (attributes, methods, multiplicity, etc) to communicate the important parts of your design.

b. Now we also want to add undo/redo functionality. The ApplicationUserInterface class gets 2 more methods: undo() and redo(). Draw a new **class diagram** and a new **sequence diagram** with the undo/redo functionality. The sequence diagram should show the scenario where the user draws a line first, and then does an undo action.
Make sure you add all necessary UML elements.