Question 1 [15 points] {15 minutes}

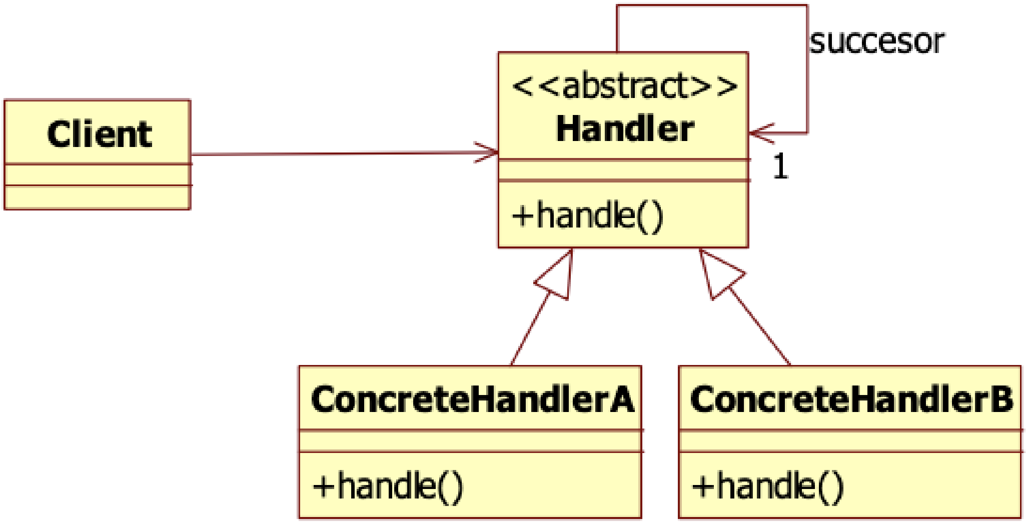
a) Explain clearly when you would use the Chain of Responsibility pattern. What problem does this pattern solve?

b) Draw the class diagram of the Chain of Responsibility pattern.

RESULT 1

a) With the Chain of Responsibility pattern a request is send to a chain of multiple handlers. The handlers themselves decide if they handle the particular request and if they pass the request to the next handler.
You use this pattern when a certain request should be handled by a certain handler class and you have more than one handler class. You need flexibility in the chain of handler classes.

b) .



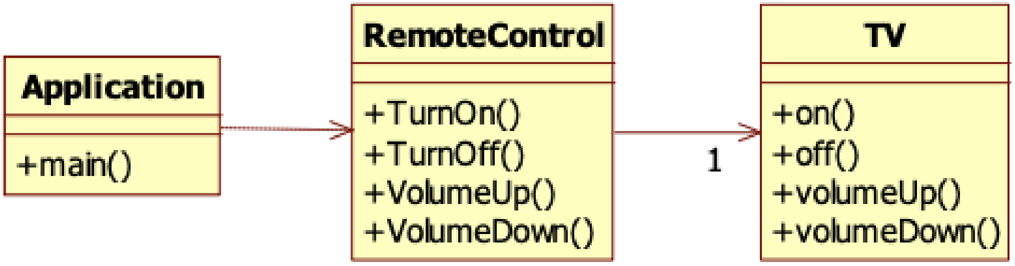
Question 2 [10 points] {15 minutes}

Suppose you have to design and implement a document editor like Microsoft Word. Explain for each of the following design patterns what problem this particular pattern can solve in a document editor like MS Word. Watch out, you have only 15 minutes for this question so your answer should be short and can be just 1 or 2 sentences.

Pattern	Document editor problem that this pattern solves
Strategy	Different formatting algorithms
Iterator	Spell checking
Composite	Document structure
Command	Undo/redo functionality
Mediator	Mediate between the different buttons and menu items on the GUI
State	Different view states (1 page per screen, 2 pages per screen, full screen, web layout, etc)

Question 3 [40 points] {50 minutes}

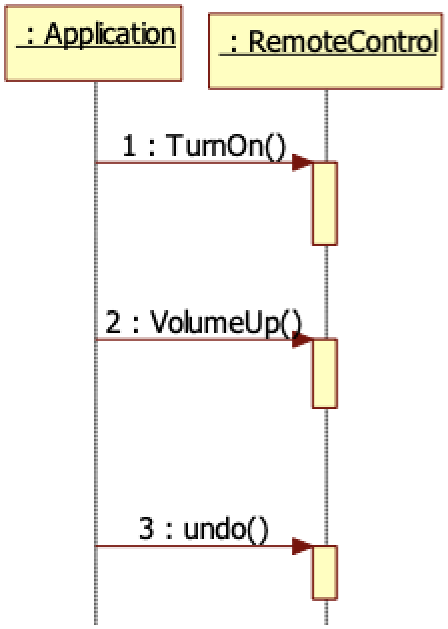
Suppose we have to develop in Java a remote control that operates on a TV. The remote control can perform the following actions on the TV: TurnOn, TurnOff, VolumeUp and VolumeDown. We make the following simple design:



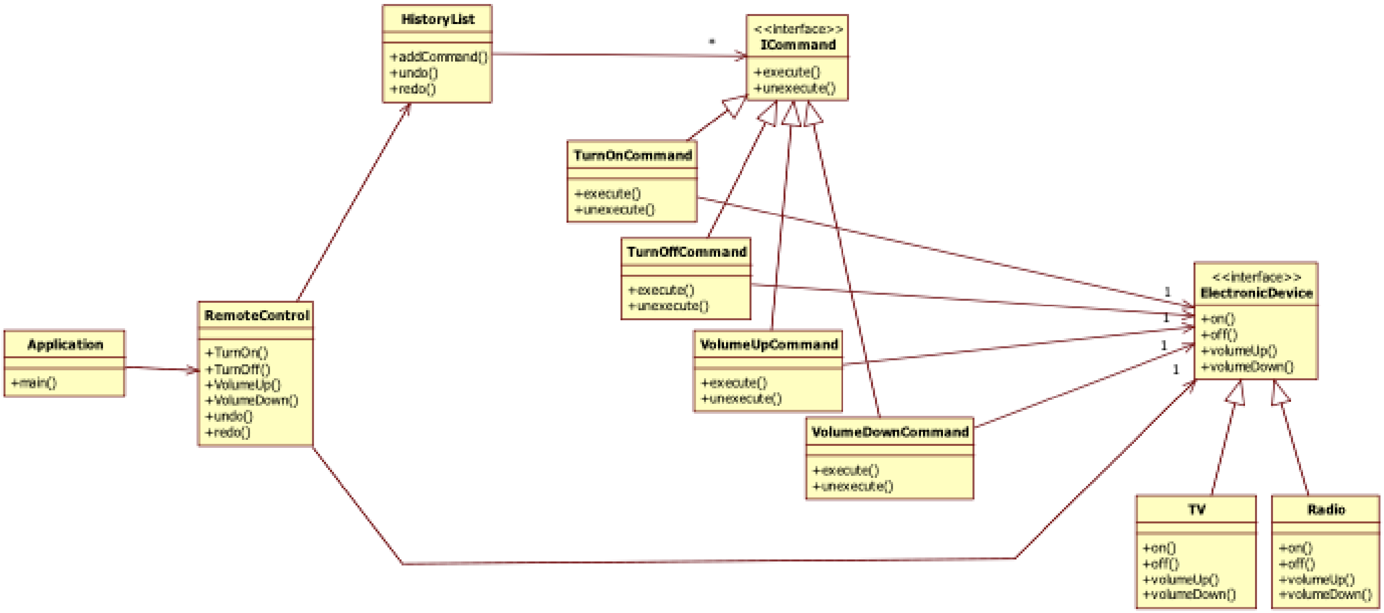
When we discuss this design with our client and our senior software architect, we discover that we have to modify our current design with the following requirements:

- The RemoteControl should not only work for a TV, but also work on other electronic devices like a Radio or a DVDPlayer. The RemoteControl can operate on only 1 electronic device (let's say a TV), but it should be very easy to support another electronic device (let's say a Radio) without changing the code of the RemoteControl. So our RemoteControl should be independent of the particular electronic device like TV or Radio.
- The RemoteControl should also support undo/redo actions. For example, if we call the VolumeUp() method on the RemoteControl, the volumeUp() method on the TV should be called. If we then call the undo method on the RemoteControl, the volumeDown() method on the TV should be called.

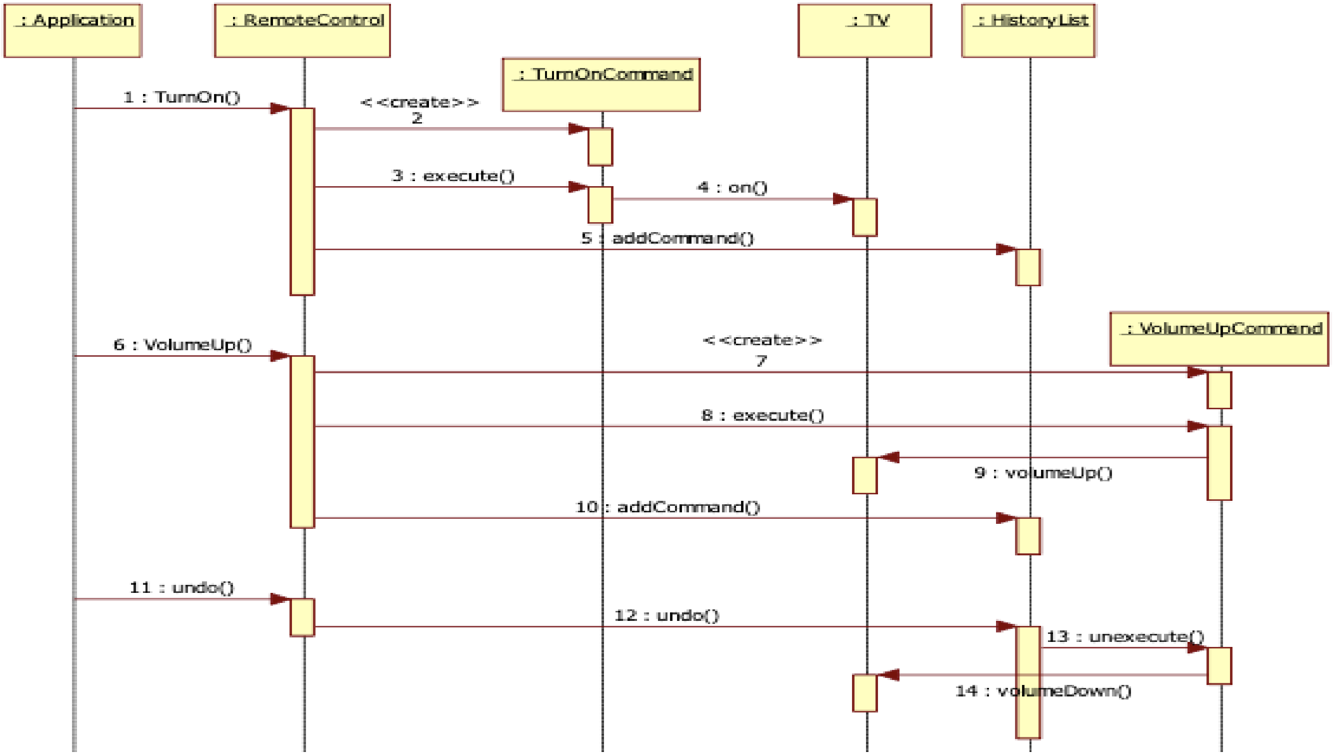
a. Draw the UML class diagram that shows how your design works. Make sure your class diagram contains all the important information to communicate your design.
b. Draw the UML sequence diagram that shows how your design works. This sequence diagram should show the following sequence: (see next page)


: Application
: RemoteControl
1 : TurnOn()
2 : VolumeUp()
3 : undo()

RESULT 3


HistoryList
+addCommand()
+undo()
+redo()
*
<<interface>>
ICommand
+execute()
+unexecute()
TurnOnCommand
+execute()
+unexecute()
TurnOffCommand
+execute()
+unexecute()
VolumeUpCommand
+execute()
+unexecute()
VolumeDownCommand
+execute()
+unexecute()
Application
+main()
RemoteControl
+TurnOn()
+TurnOff()
+VolumeUp()
+VolumeDown()
+undo()
+redo()
1
1
1
1
<<interface>>
ElectronicDevice
+on()
+off()
+volumeUp()
+volumeDown()
TV
+on()
+off()
+volumeUp()
+volumeDown()
Radio
+on()
+off()
+volumeUp()
+volumeDown()

b.



: Application
: RemoteControl
: TurnOnCommand
: TV
: HistoryList
: VolumeUpCommand
1 : TurnOn()
<<create>>
2
3 : execute()
4 : on()
5 : addCommand()
6 : VolumeUp()
<<create>>
7
8 : execute()
9 : volumeUp()
10 : addCommand()
11 : undo()
12 : undo()
13 : unexecute()
14 : volumeDown()

Question 4 [30 points] {40 minutes}

Suppose that you have to write an order fulfillment application according to the following requirements:

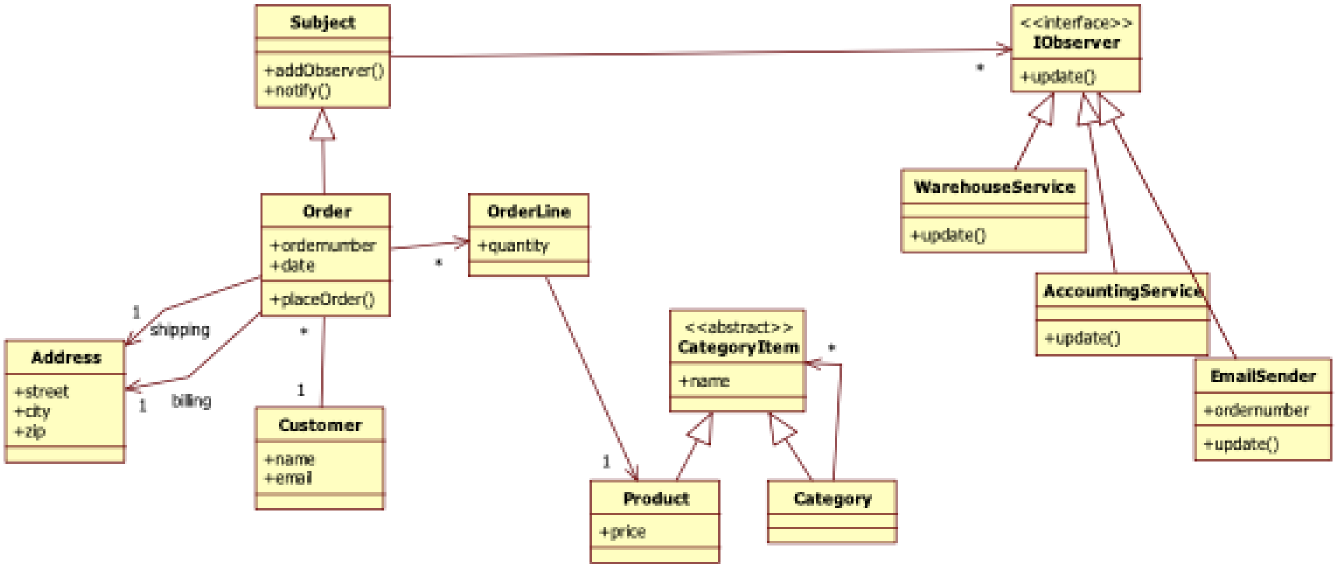
1. You can browse through product categories to find the right product. The application should support all kind of categories. For example when you select the "clothing" categories you will see all categories of "clothing" like "shoes", "t-shirts", etc. When you select "shoes" you see all categories of shoes like "men", "woman", "children", etc.
2. You can create orders. An order contains the following data:
   - order number
   - order date
   - multiple products with product name, product price
   - quantity of certain products
   - customer with name, email, street, city, zip
   - shipping address with name, street, city, zip
   - billing address with name, street, city, zip
3. Whenever a customer places an order the following actions need to be done:
   - The warehouse service needs to handle the order fulfillment
   - The accounting service needs to handle the payment for the order
   - The customer needs to get an email

   It should be easy to add more actions without changing the Order class

Draw the partial class diagram that shows how your design works. Do NOT draw the whole class diagram of the order fulfillment application, but only the class diagram that shows how your design works. You do not need to add GUI or DAO classes to your design.

Make sure your class diagram contains all the important information to communicate your design. Your class diagram should also show the classes that contain the data of an order given in requirement number 2 above.

RESULT 4


Subject
+addObserver()
+notify()
<<interface>>
IObserver
+update()
*
Order
+ordernumber
+date
+placeOrder()
OrderLine
+quantity
*
WarehouseService
+update()
AccountingService
+update()
EmailSender
+ordernumber
+update()
Address
+street
+city
+zip
1
shipping
1
billing
*
1
Customer
+name
+email
<<abstract>>
CategoryItem
+name
*
1
Product
+price
Category