<<interface>>

**EmailItem**

\*

**EmailGUI**

+createEmail()

+sendEmail()

+receiveEmail()

+forwardEmail()

+replyEmail()

**EmailService**

+createEmail()

+sendEmail()

1 +receiveEmail()

+forwardEmail()

+replyEmail()

+getAccount()

**EmailAccount**

+name

+emailAddress

+incomingMailServer

+outgoingMailServer

1 +username

+password

**EmailFolder**

+name **Email**

\* +from

+to

+subject

+content

+received

**Attachment**

+name

\* +link

Question 1

**EmailSender**

1 +sendEmailToServer()

**EmailReceiver**

1 +receiveEmailsFromServer()

**FileDAO**

1 +loadAccount()

+saveAccount()

Question 2

**History List**

+addCommand()

+undo()

+redo()

<<interface>>

**ICommand**

\* +execute()

+unexecute()

1

**PushCommand**

**PopCommand**

**Subject**

+addObserver()

+notify()

<<interface>>

**IObserver**

\* +update()

+execute()

+unexecute()

+execute()

+unexecute()

**ButtonGUI**

+push()

+pop()

+undo()

+redo()

1

1

**Stack**

**StackDAO**

+update()

+updateStack()

**StackViewerGUI**

+push()

+pop()

+showStack()

+update()

: ButtonGUI

: Stack : HistoryList : StackDAO : StackViewerGUI

<<create>> : PushCommand

1

2 : execute() 3 : push()

4 : notify()

5 : update()

6 : updateStack()

7 : update()

8 : showStack()

9 : addCommand()

10 : undo()

11 : unexecute()

12 : pop()

13 : notify()

14 : update()

15 : updateStack()

16 : update()

17 : showStack()

**Question 3**

a.

|  |  |
| --- | --- |
| **Observer pattern** | **COR pattern** |
| Observers don’t know each other | Handler needs to know the next handler |
| All observers are called | A handler may decide not to call the next handler |
| We have a list of observers | We have a chain of handlers |

b.

You use COR when only one class needs to handle the action, and you don’t know which class. You use the observer pattern when all observers need to be notified