Software Methods and Tools

Fall 2015
Instructor: Yongjie Zheng

Lab #3: UML Use Case Diagram, Sequence Diagram, and State Diagram

Description:

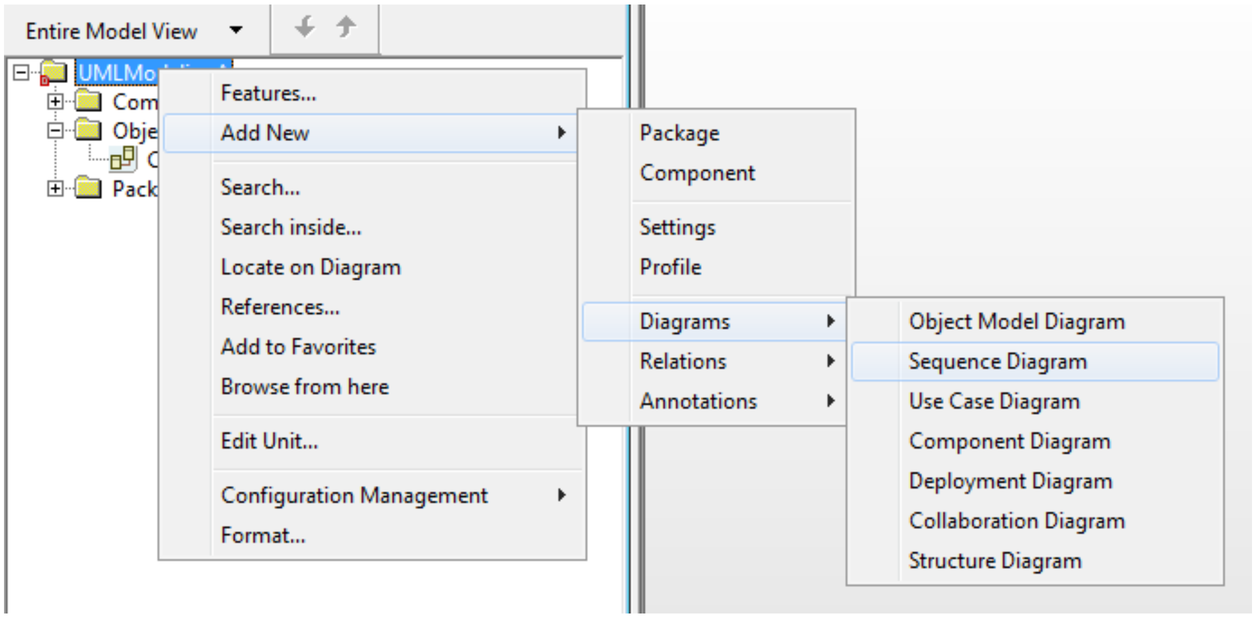
In this lab, we are going to use IBM Rational Software Modeler to create UML behavior diagrams, including use case diagrams, sequence diagrams, and state diagrams.

Note: when you run Rational Software Modeler, choose *Open Project* in the pop-up wizard panel and open the project that we created in Lab #2. We will need the classes we created last time.

Instructions:

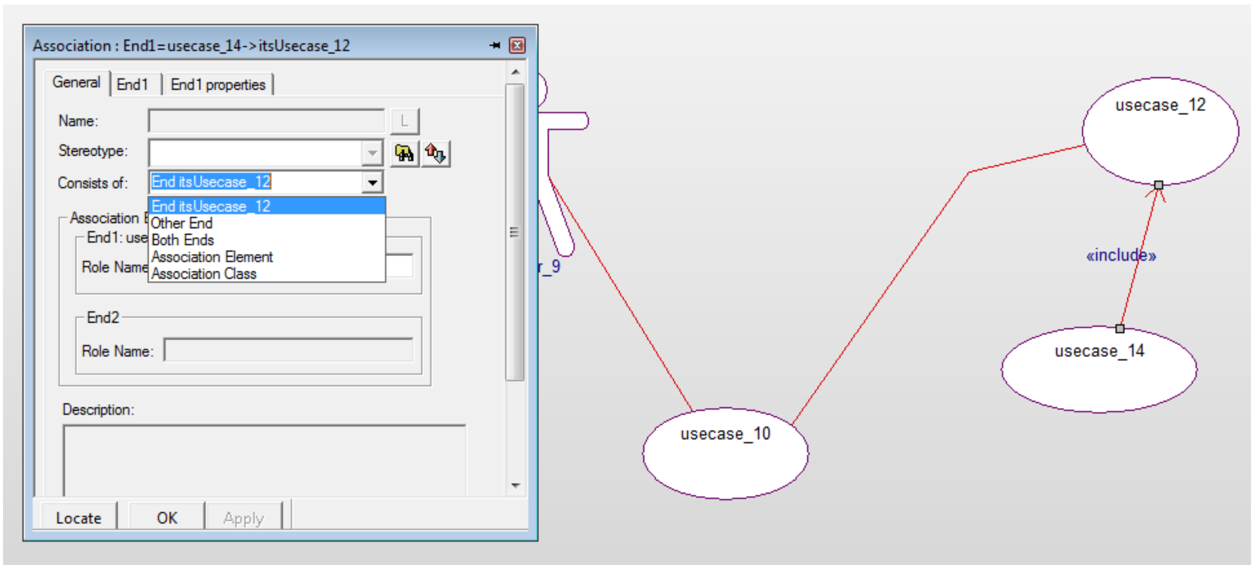
1. Creating a use case diagram

Right click your project, select *Add New* -> *Diagrams* --> *Use Case Diagram*. Look at the following figure for reference.



Similar to what we did in Lab #2, we will be primarily using Palette to create those basic elements of a use case diagram.

Note: you probably cannot find the **include** and **extend** relationships that we discussed in class. To create these relationships, choose *Associations* in the palette and you will see an undirected line. Next, double click the line and click the drop down menu that is labelled *Consist of*. Change its value from *Both Ends* to your target end, as shown in the following figure.



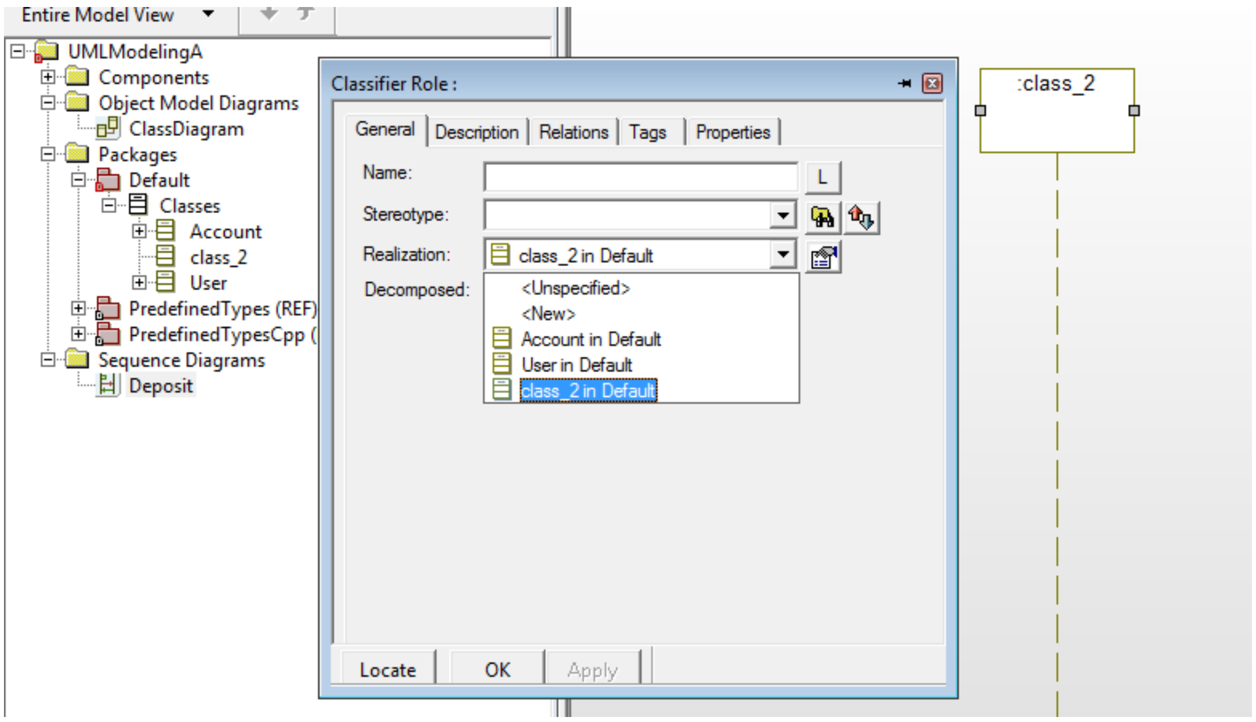
You need to figure out how to add either <<include>> or <<extend>> to your association. Hint: they can be seen as stereotype.

2. Creating a sequence diagram based on a class diagram.

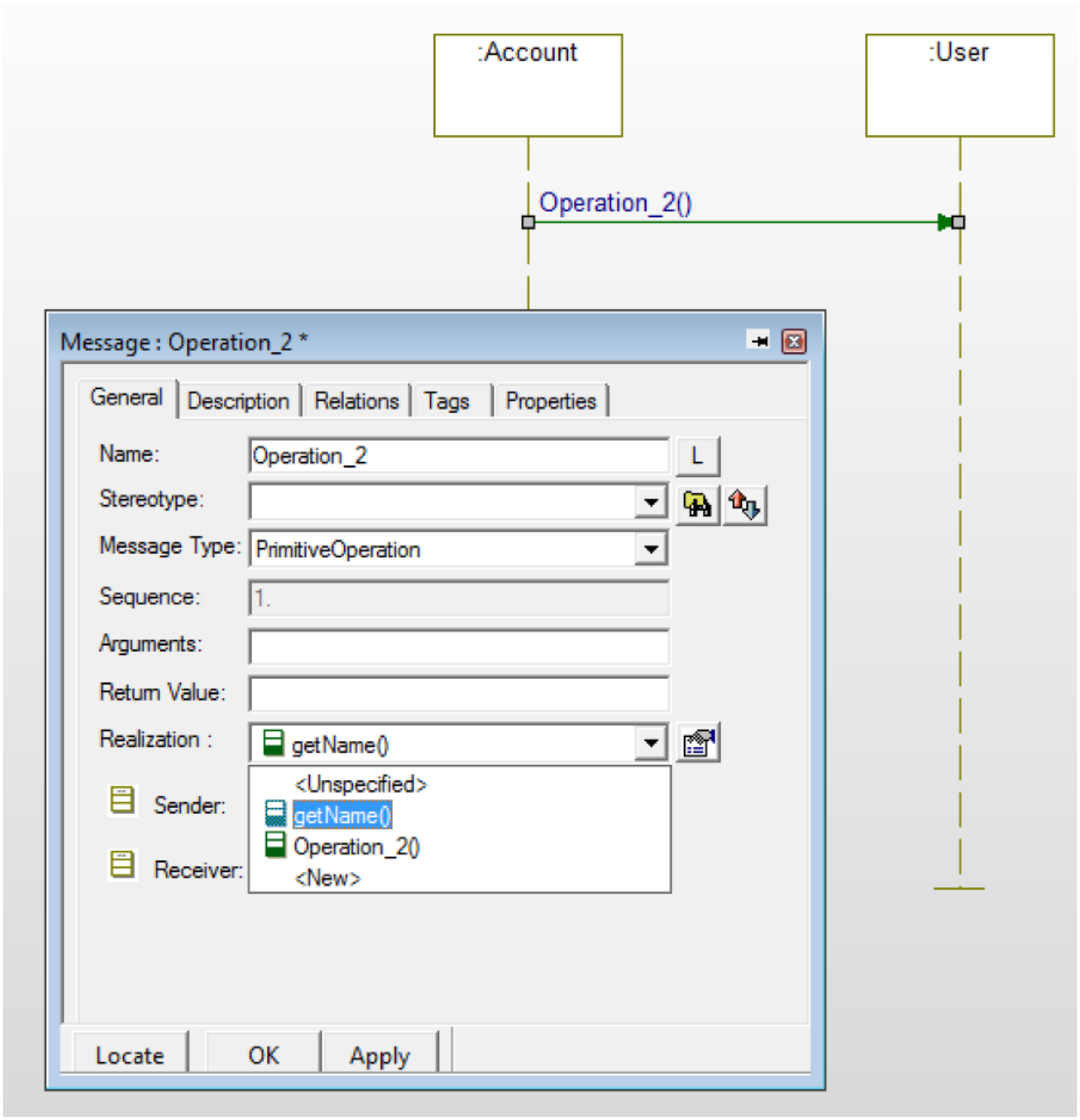
Similar to the process of creating a use case diagram, you can create a sequence diagram in the project.

To add a participating object, choose *Instance Line* from the palette. Double click the participating object you created, you can assign an existing class to it. Here you can use classes included in the class diagram that you created in Lab #2. Look at the figure on the next page for reference.

You can find the list of classes that you created last time from: *Packages -> Default -> Classes*.



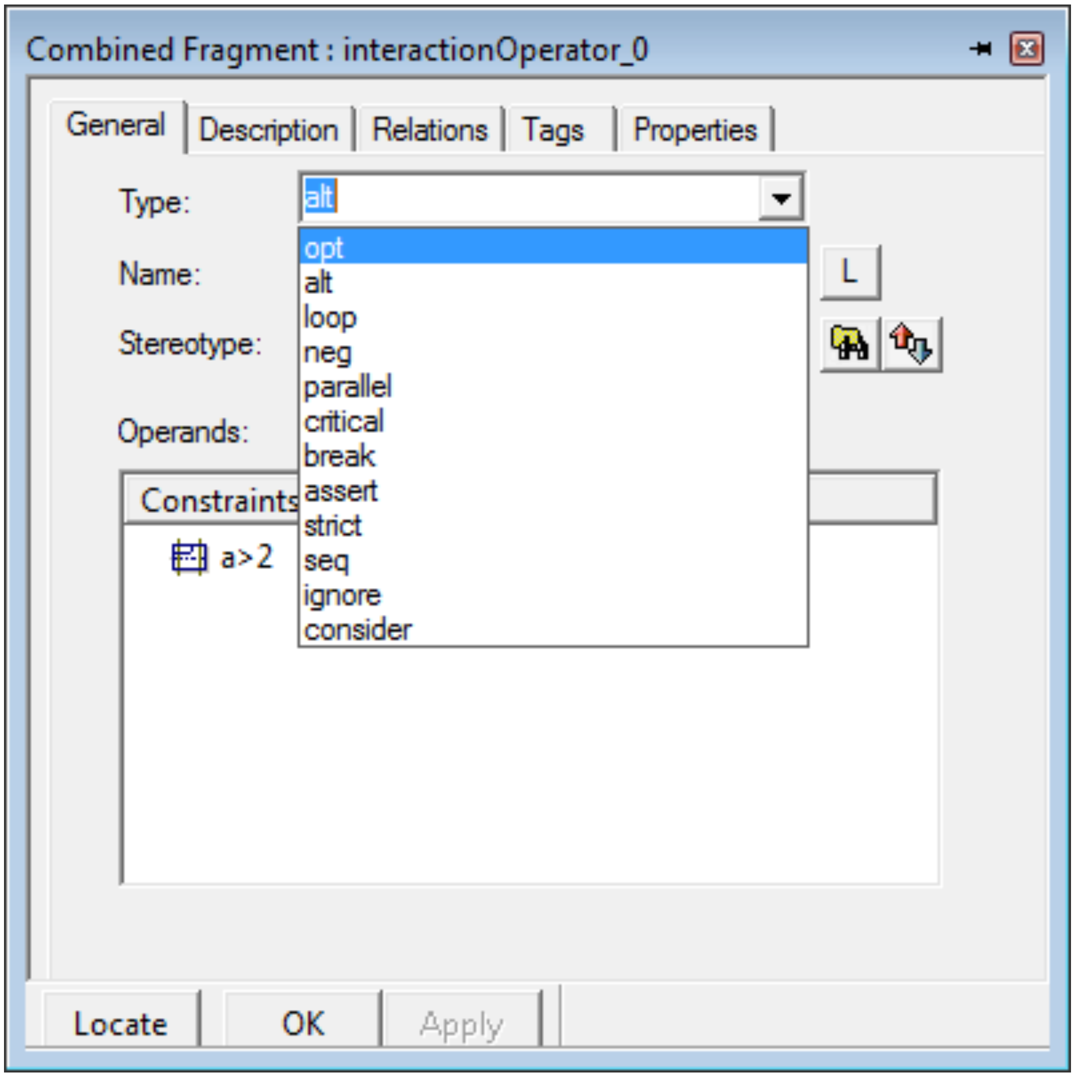
Similarly, you can explicitly specify which method to be associated when you create a message between two participants, as shown in the following figure.



Name:	Operation_2
Stereotype:	
Message Type:	PrimitiveOperation
Sequence:	1.
Arguments:	
Return Value:	
Realization :	getName()
Sender:	
Receiver:	

To add a frame into a sequence diagram, choose I*nteraction Operator* from the palette. Once you see the frame created in the canvas, you can double click the frame and select an operator as shown in the following figure.

For the alt operator, you will need multiple fragments. To add a fragment into the alt frame, choose *Operand Separator* and release it in the frame.



Type: alt

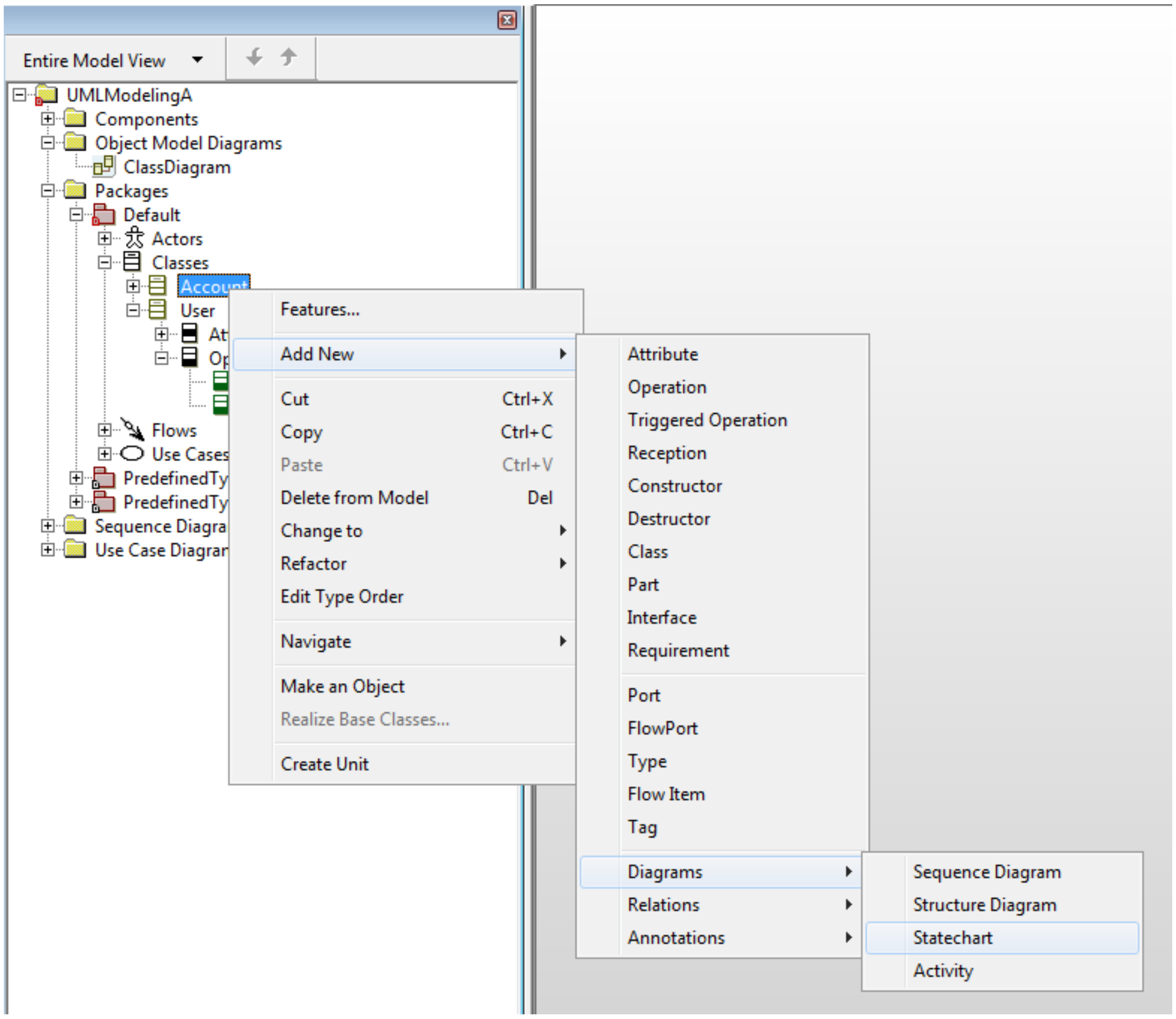
opt
alt
loop
neg
parallel
critical
break
assert
strict
seq
ignore
consider

Constraints: a>2

3. Creating a state diagram.

The only thing different about creating a state diagram is where to start. :-)

This time, you need to find the class that you want to create a state diagram for. Select Packages -> Classes -> [target class] -> Add New -> Diagrams -> Statechart. Look at the following figure.



Exit Exercise

Finish the following three tasks.

Task 1: Create a use case diagram for the online banking system based on the descriptions provided below.

The online banking system has two types of users: customer and system admin. Each of them can interact with the system in the following ways.

Customer:
- Open an account (a request will be sent to system admin)
- Close an account (a request will be sent to system admin)
- Check the balance of an account
- Print the statement of an account
- Transfer money between two accounts of the customer
- Transfer money between the customer’s account and another customer's account
- Report a problem if a transfer transaction fails.

System Admin:
- Handles the OpenAccount/CloseAccount request from the customer.
- Review the details of a transaction.

Note: Use the “include”, generalization, and “extend” relationships as needed.

Task 2: Create a sequence diagram for the online banking system based on the use case provided below.

Use Case Name	Transfer money from checking account to saving account
Scope	Online Bank
Primary Actor	Customer
Preconditions	The customer is logged in and both accounts exist
Guarantee (Postconditions)	The money is transferred from source account to target account.
Main Success Scenario	1. Customer selects “Transfer Fund” 2. Customer selects the checking account 3. Customer selects the saving account 4. Customer enters the amount to be transferred and clicks “Transfer”. 5. A transaction is started. 6. The transaction generates a log. 7. The transaction returns a confirmation message to the customer.
Extensions	5. If the amount customer entered is greater than the balance of the checking account, the system will ask customer to re-enter the amount.
Special Requirements	

Frequency	
Miscellaneous	

Hint: there should be at least four participants in your sequence diagram: Customer, CheckingAccount, SavingAccount, and Transaction.

Task 3: Create a state diagram for the SavingAccount object. Your diagram should include the following states:

- Initial State
- Open (maintenance fee will be charged)
- Free (no maintenance fee if the balance is over certain limit)
- Frozen (A do/warn() activity must be included)
- Closed
- Final State

Additional information:
- When a saving account is first created, it is in the “Open” state with 0 balance.
- Once the balance exceeds certain limit, the state is changed to “Free”.
- The account will become “Frozen” when illegal actions are detected (e.g. unrecognized transaction). It may return to its original state at customer’s request.
- An account can be closed at customer’s request.

Draw all the transitions based on the above description and label them appropriately.

Show the results to TA.