ABSTRACT

ATM stands for automatic teller machine, and DON WEZTEL invented the first successful and modern ATM in the USA. Software is to be designed for supporting a computerized ATM banking network. An ATM accepts a relevant cash card, interacts with the user, communicates with the central system to carry out the transactions, dispenses cash and prints receipts and it must include appropriate record keeping and security provisions. A session is started when a customer inserts an ATM card into the card reader slot machine. The ATM pulls the card into the machine and reads it. The customer is asked to enter his/her personal identification number (PIN) and is then allowed to perform one or more transactions. The transactions will then be sent to the bank, along with information from the customer's card and the PIN the customer entered. If the bank approves the transaction, the transaction will be performed, if the customer wishes. If the bank reports that the customer's PIN is invalid, the invalid PIN extensions will be performed and then as attempt will be made to continue the transaction. If the customer's card is returned due to too many invalid PIN's, the transaction will be aborted and the customer will not be offered the option of doing another. After dispensing the cash and printing the receipt, the customer is asked whether he/she would like to perform another transaction, if not the card is ejected from the machine and the session ends. The customer may abort the session by pressing the cancel key when entering a PIN or choosing a transaction type. In this project the software Rational Rose is used to generate code for the diagrams in the UML.

INTRODUCTION

Automatic teller machine is a machine which is used to get instant money round the clock. ATM's were the pioneers, and much of commercial cryptography was developed in the late 1970's and early 1980's in order to take the real or perceived security problems of ATM systems. This technology has since been applied to many others systems, such as lottery terminals, pre payment electricity meters, satellite and cable Tv decoders, and burglar alarms etc. Security systems which are to provide evidence must be designed and certified on the assumptions that they will be examined in the detail by a hostile expert.

Expect the real problems to come from blinders in the application design in the way the system is operated. Before setting out to build a computer security system, make sure you understand what its real purpose is. Understand how liability is transferred by any systems you build or relay on. Computer security legislation is highly likely suffer from the law of unexpected consequences. Don't relay on engineering standards to solve legal problems.

Facts about ATM:

- Through ATM we get money.
- If we enter invalid PIN number more than three times our debit or credit card will be blocked.
- We should not use our mobiles at the ATM counters because satellite link between ATM machines and the mobiles which we are using will be overlapped.
- At the ATM counters there will be cameras fixed to identify the customer, so the customers are not allowed with helmets.

DESIGN AND ANALYSIS

Class diagram for ATM system

Shown below is the class diagram for the ATM system. The basic structure of the class diagram arises from the responsibilities and relationships discovered when doing the CRC cards and Interaction Diagrams. (If a class uses another class as a collaborator, or sends a message to an object of that class during an interaction, then there must either be an association linking objects of those classes, or linking the “sending” classes to an object which provides access to an object of the “receiving” classes).

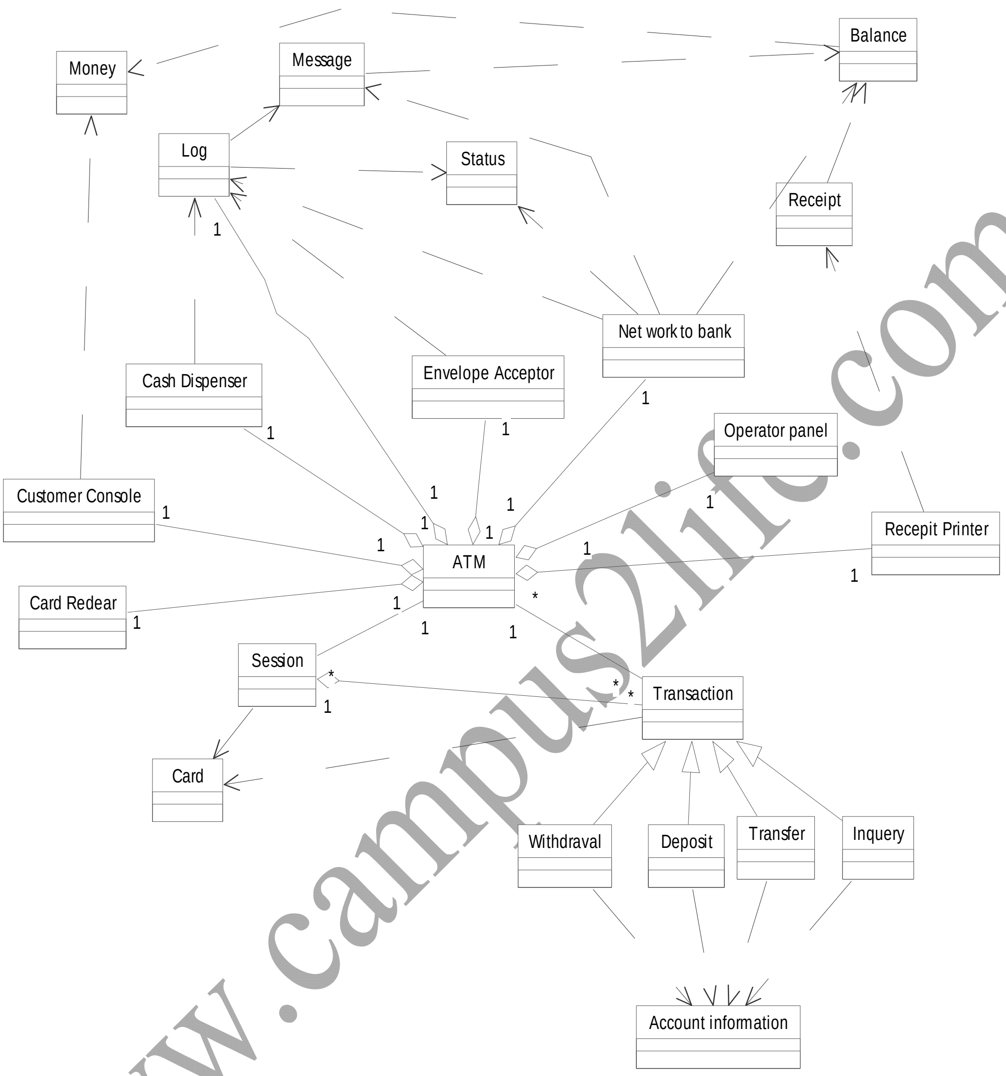
In the case of the ATM system, one of the responsibilities of the ATM is to provide access to its component parts for session and Transaction objects; thus, Session and transaction have associations to ATM, which in turn has associations to the classes representing the individual component parts. (Explicit “uses” links between session and transaction, on the one hand, and the component parts of the ATM, on the other hand have been omitted from the diagram to avoid making it excessively cluttered).

The need for the various classes in the diagram was discovered at various points in the design process.

- Some classes were discovered when doing analysis.
- Some classes were discovered when doing CRC cards
  - Message – used to represent a message to the bank.
  - Receipt – used to encapsulate information to be printed on a receipt.
  - Status – used to represent return value from message to the bank.
  - Balances – used to record balance information returned by the bank.

- Some classes were discovered when doing detailed design or writing code.
  - Money – used to represent money amounts, in numerous places.
  - Account Information – contains names of various types of accounts customer can choose from

Class Diagram for ATM System



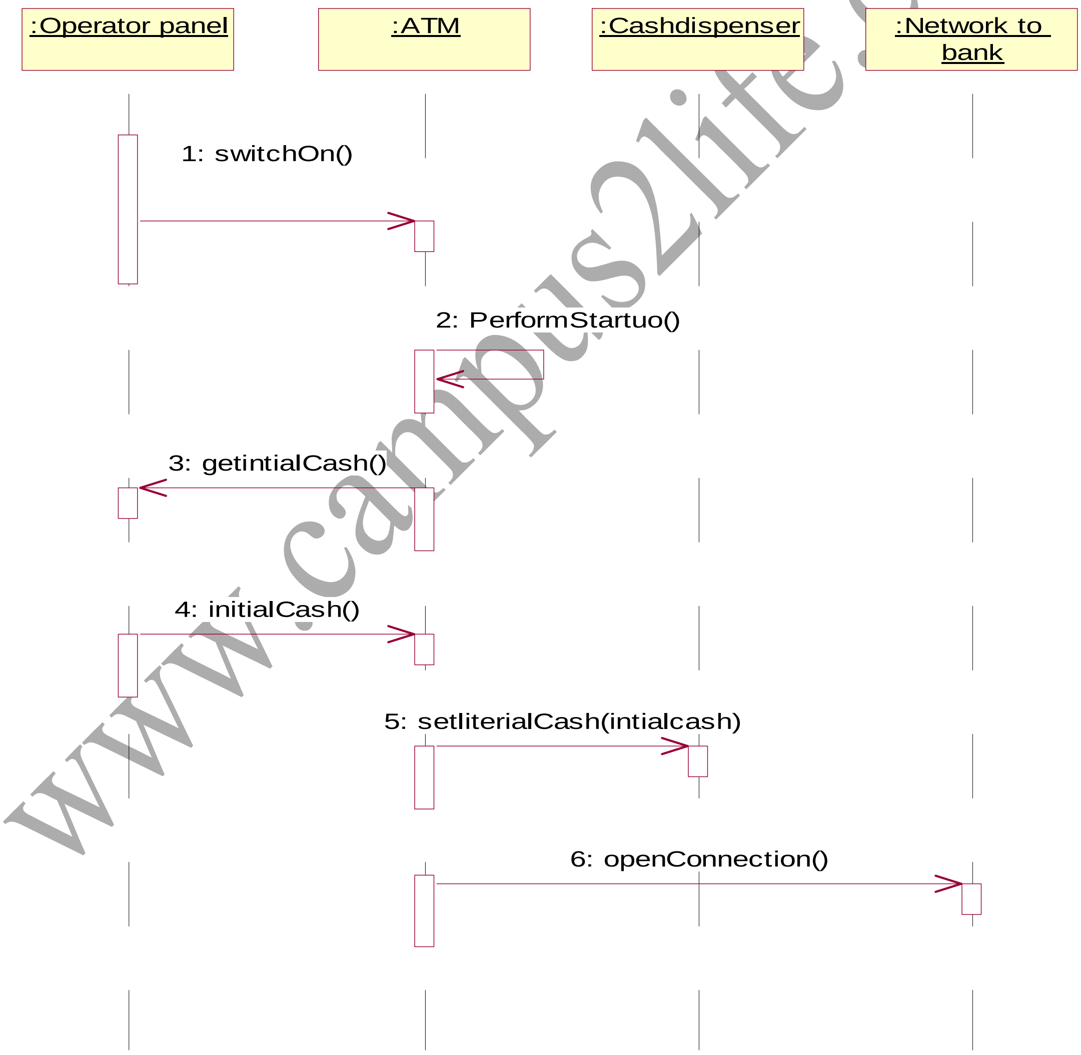
Object diagram for ATM system:

A generalization of one or more objects with a uniform set of attributes and services is object diagram.

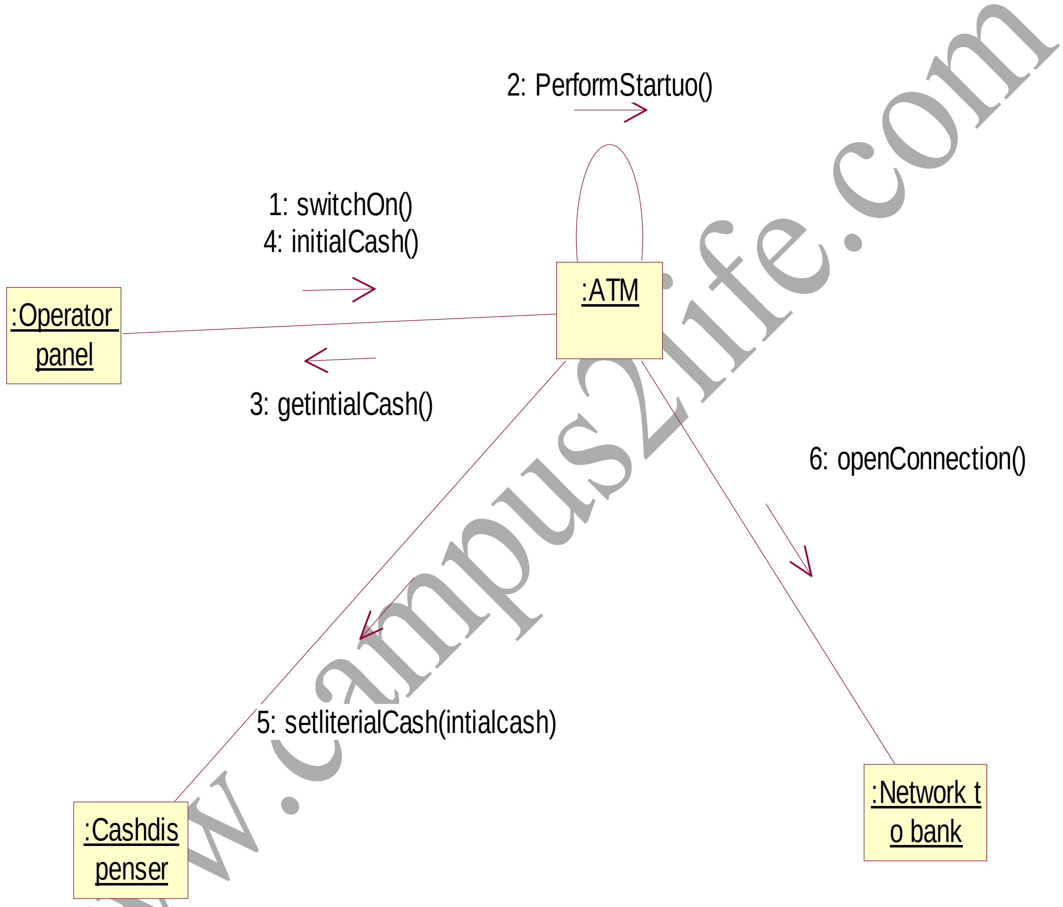
Interaction Diagrams for ATM System

Uml defines two types of interaction Diagram: The Sequence Diagram and the Collaboration Diagram. In order to illustrate both types, the major use cases are documented using Sequence Diagrams, and the specific sub cases of a transaction (withdrawal, etc…) and the Invalid PIN Extinction are documented using Collaboration Diagrams.

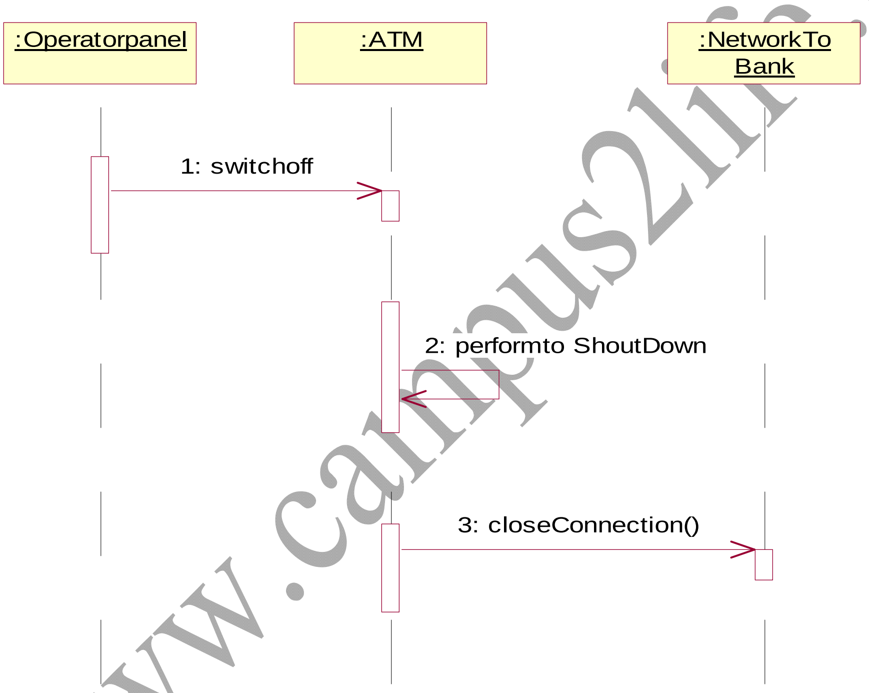
System Startup Sequence Diagram



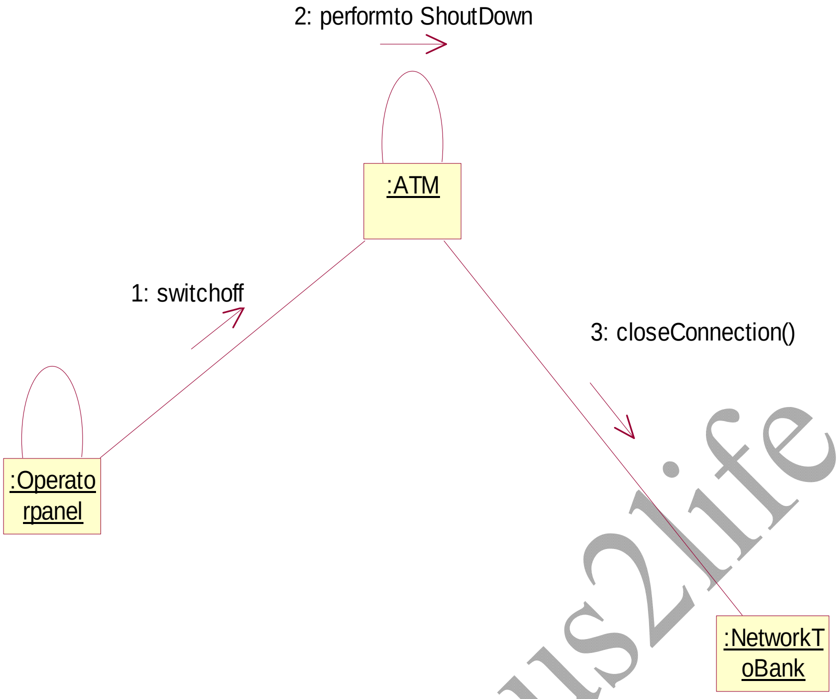
System Startup Collabration Diagram



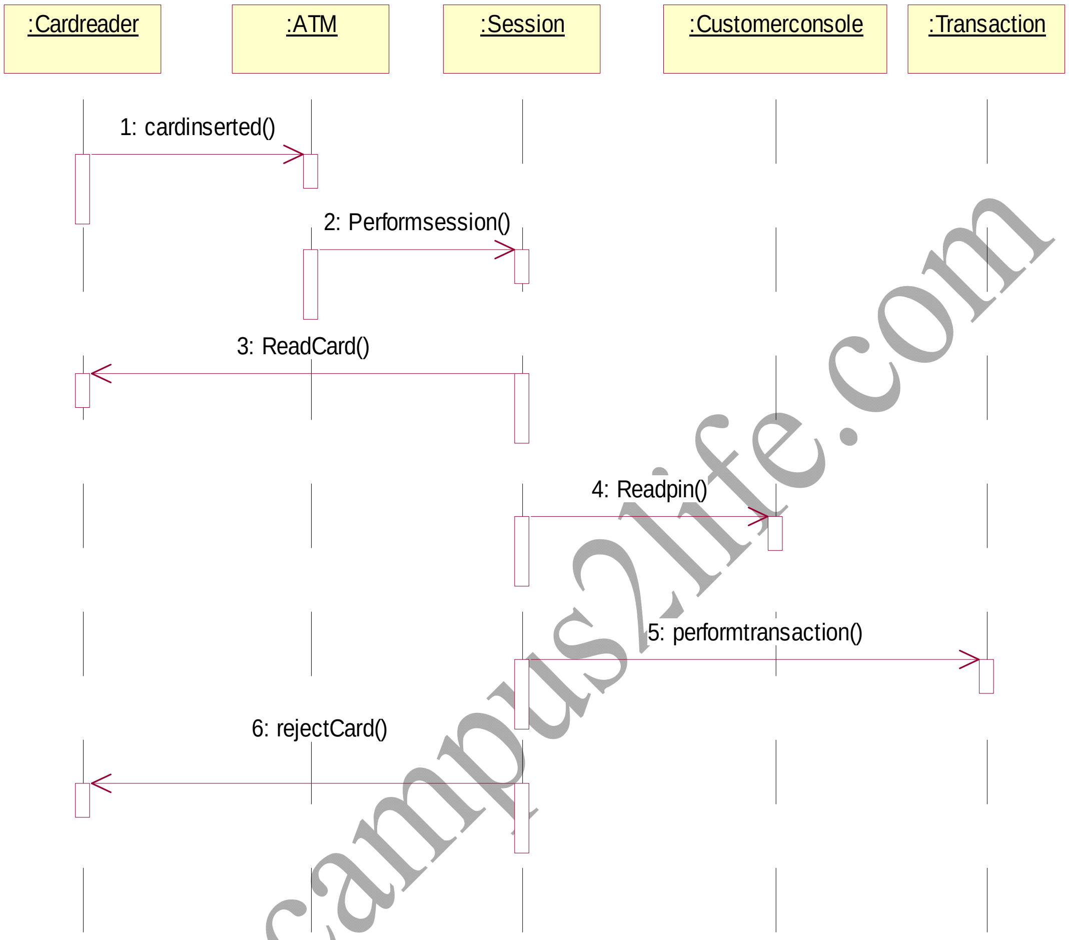
System Shutdown Sequence Diagram



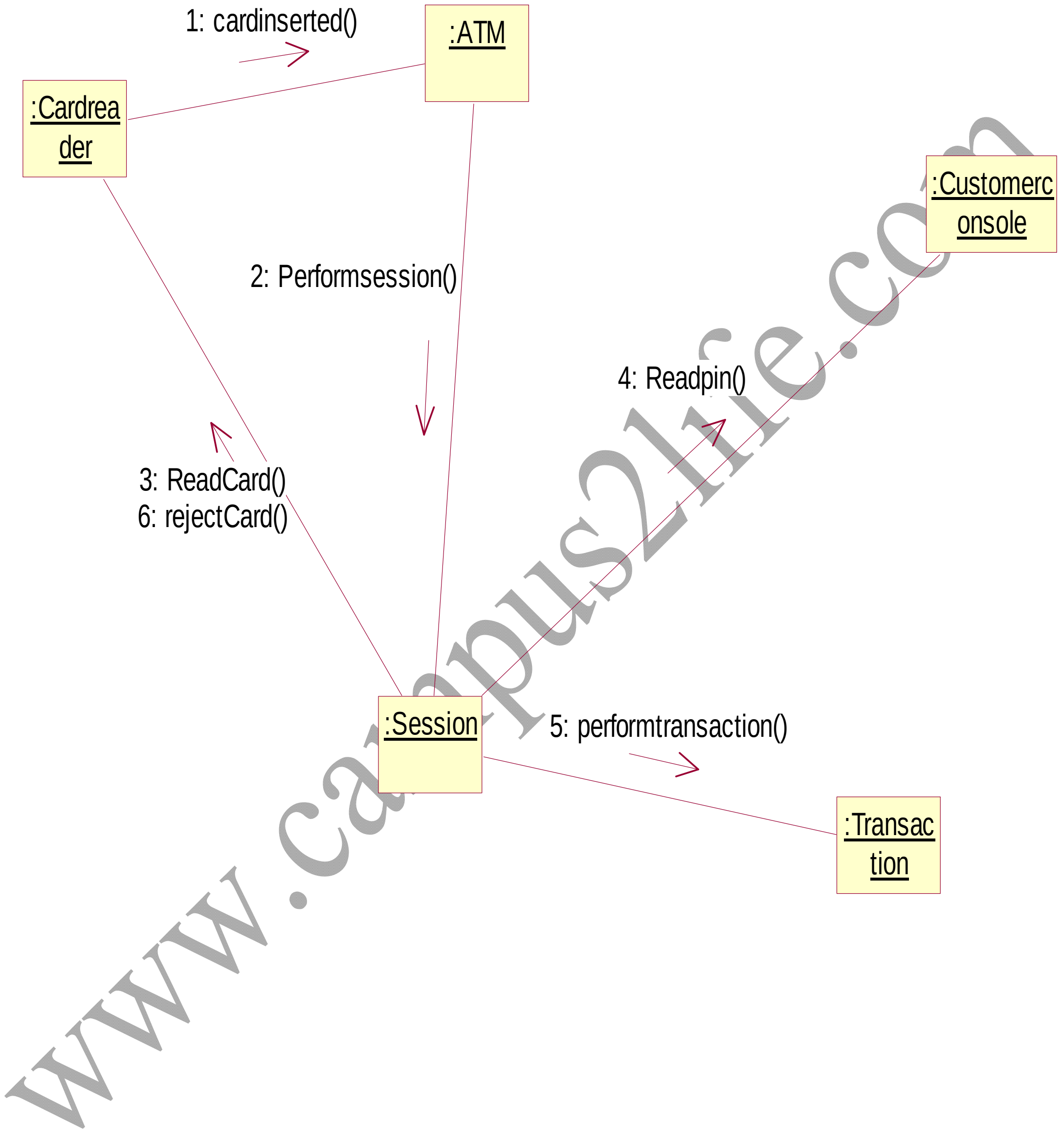
System Shutdown Collabration Diagram



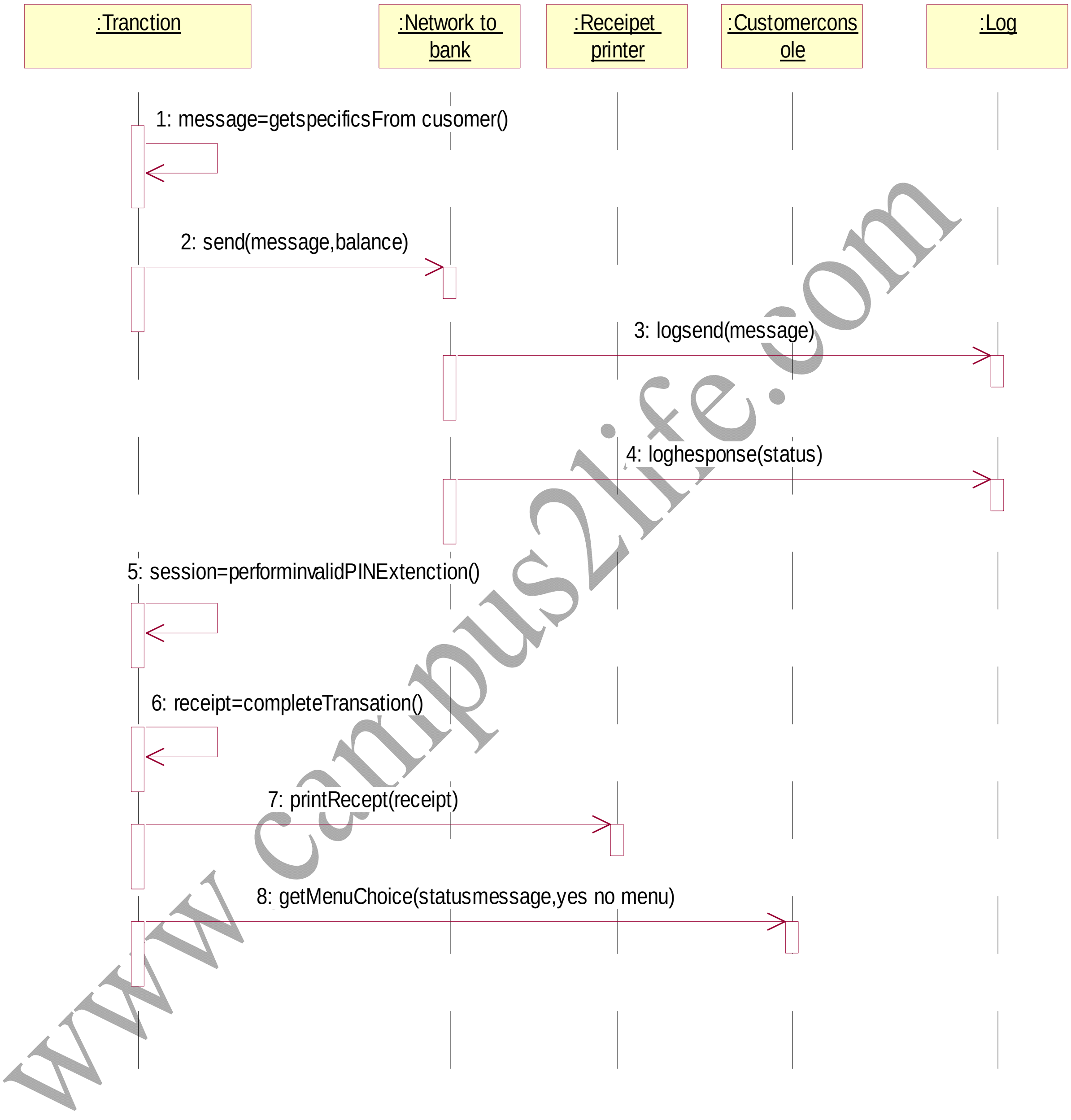
Session Sequence Diagram



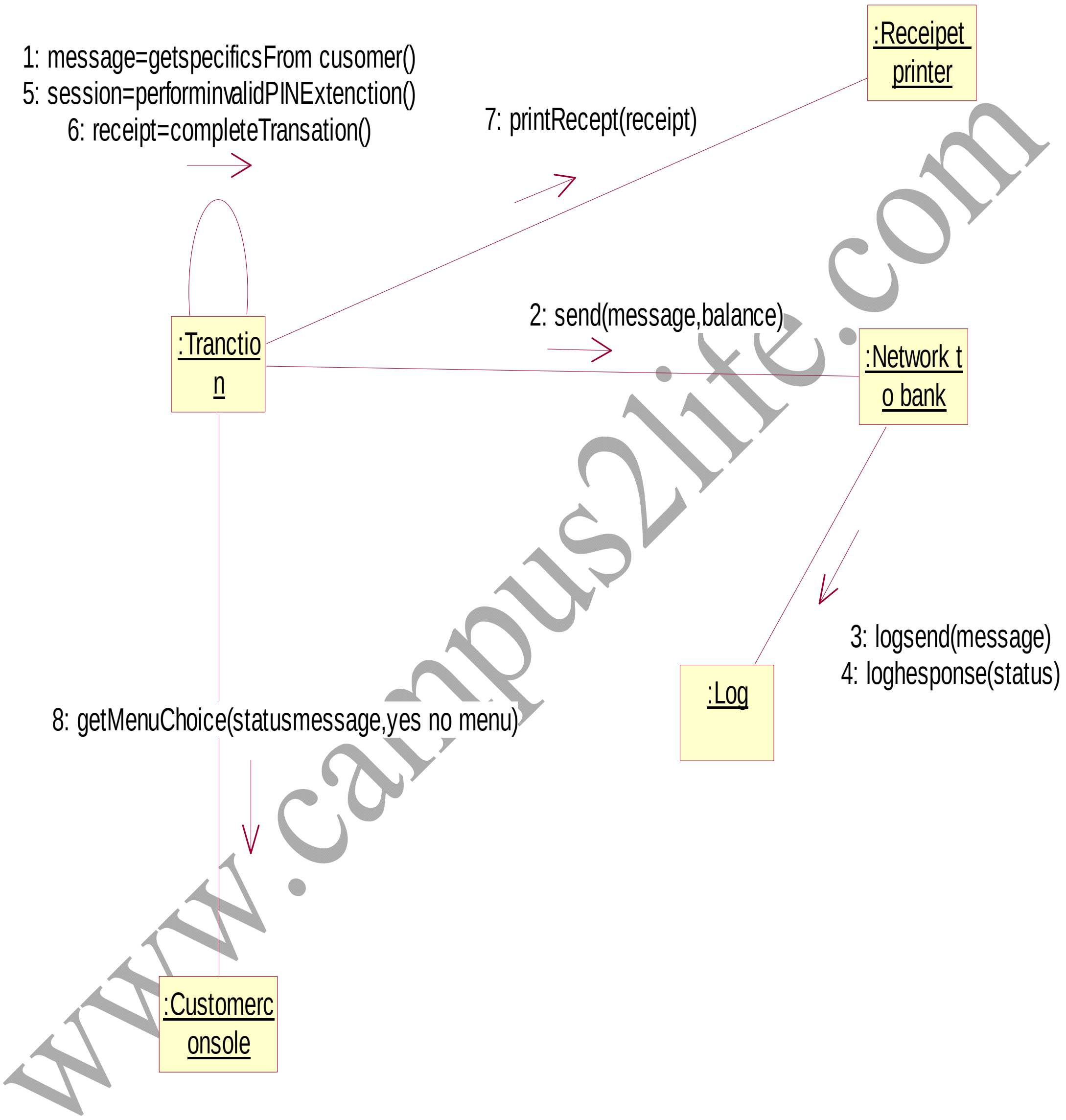
Session coallboration Diagram



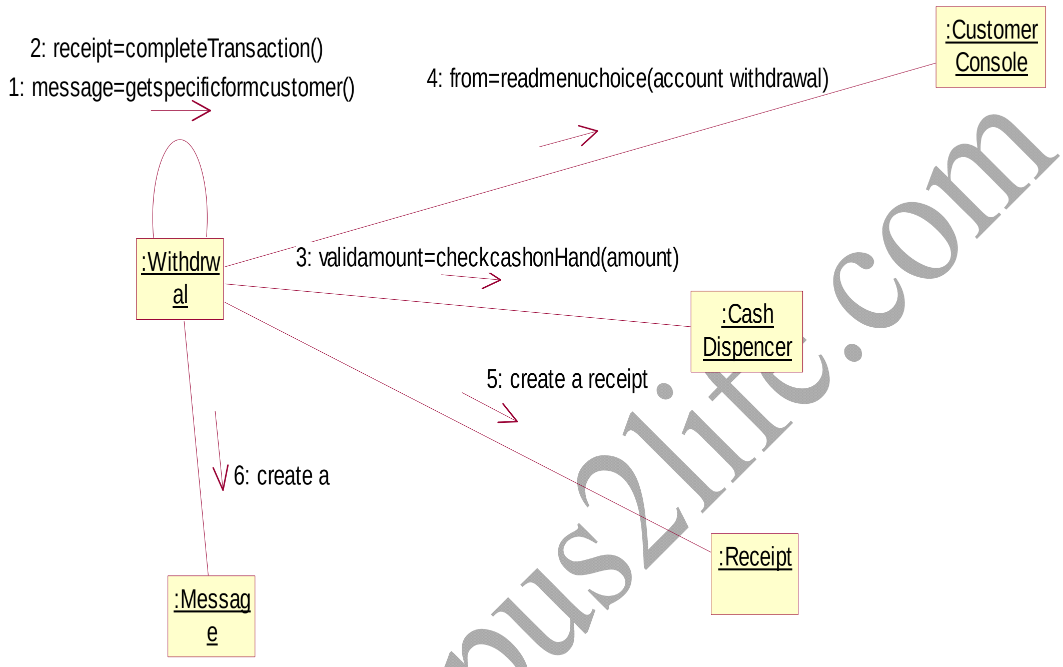
Transaction Sequence Digram


:Tranction
:Network to bank
:Receipet printer
:Customerconsole
:Log
1: message=getspecificsFrom cusomer()
2: send(message,balance)
3: logsend(message)
4: loghesponse(status)
5: session=performinvalidPINExtenction()
6: receipt=completeTransation()
7: printRecept(receipt)
8: getMenuChoice(statusmessage,yes no menu)

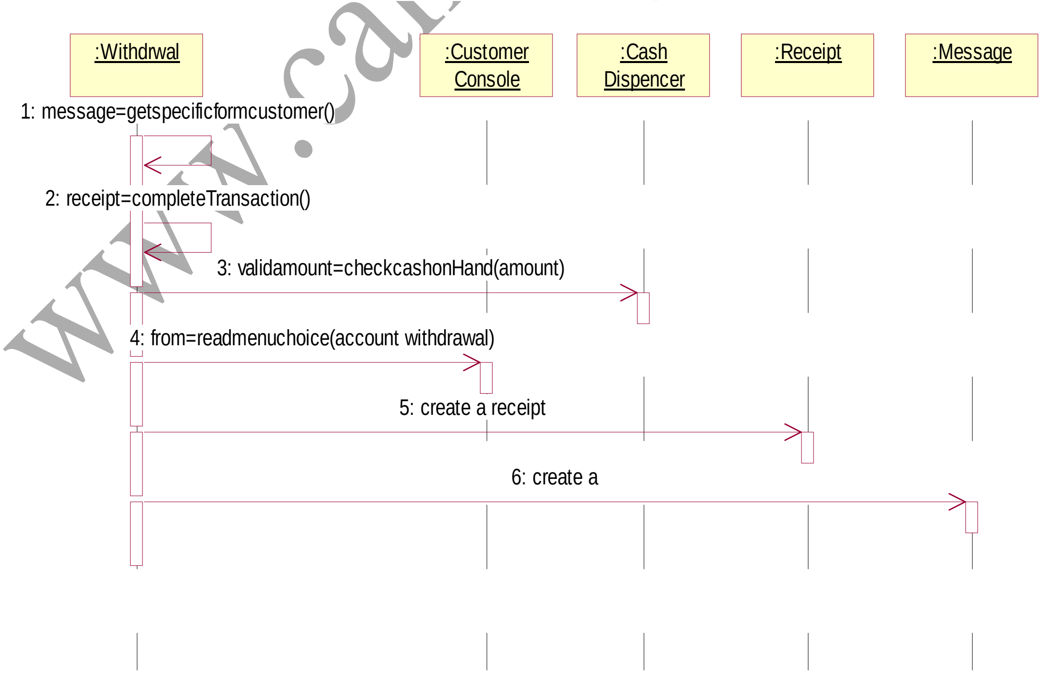
Transaction Collaboration Digram



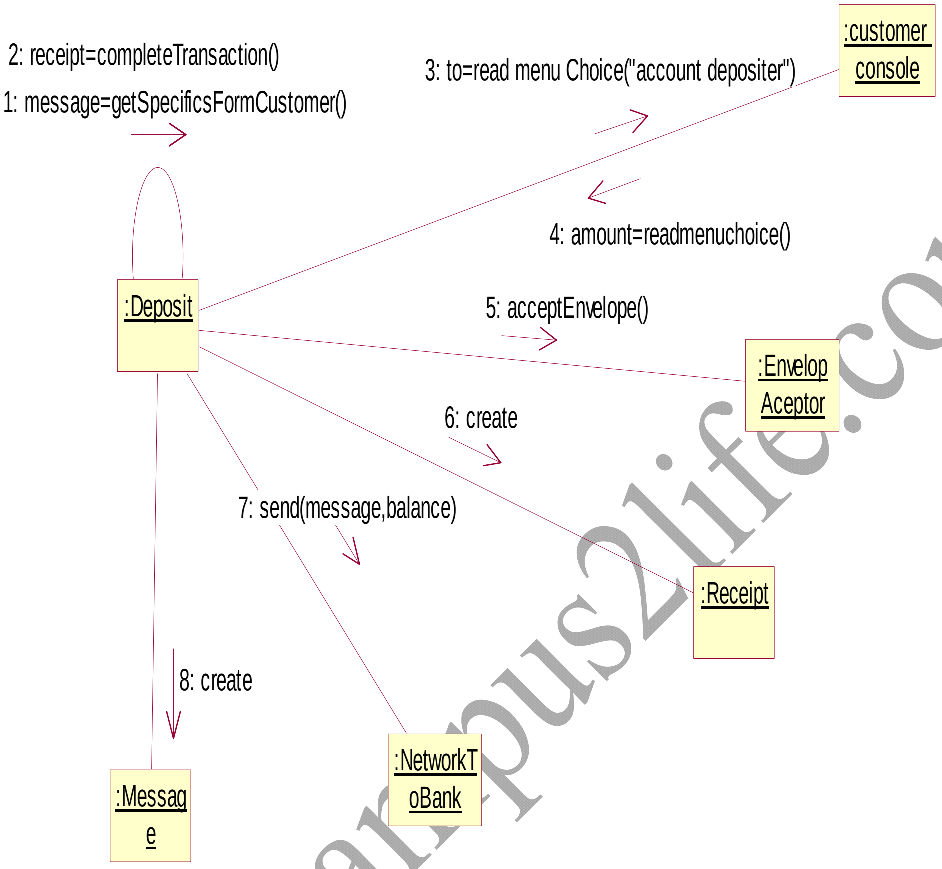
Withdrawal Transaction Collaboration


2: receipt=completeTransaction()
1: message=getspecificformcustomer()
4: from=readmenuchoice(account withdrawal)
:Customer Console
:Withdrwal
3: validamount=checkcashonHand(amount)
:Cash Dispencer
5: create a receipt
6: create a
:Receipt
:Message

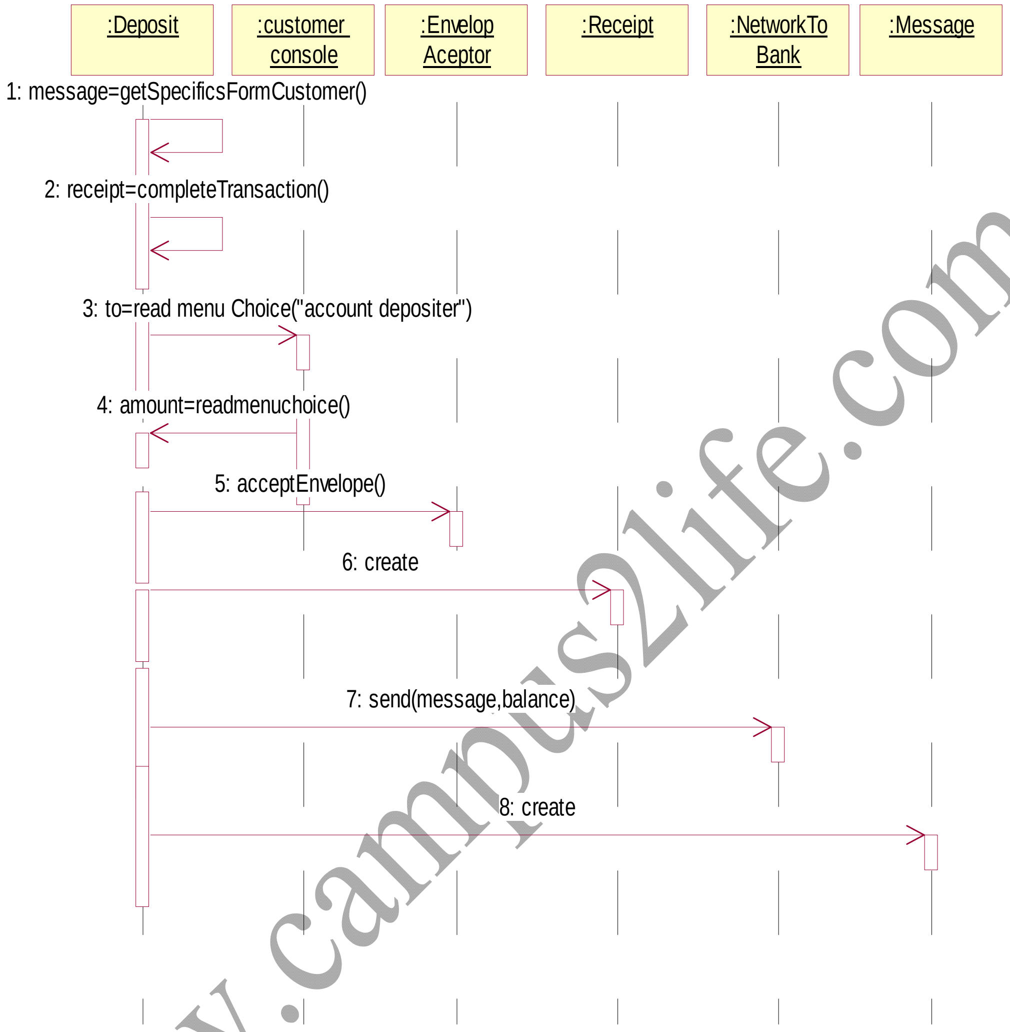
Withdrwal Transaction Sequence Digram


:Withdrwal
:Customer Console
:Cash Dispencer
:Receipt
:Message
1: message=getspecificformcustomer()
2: receipt=completeTransaction()
3: validamount=checkcashonHand(amount)
4: from=readmenuchoice(account withdrawal)
5: create a receipt
6: create a

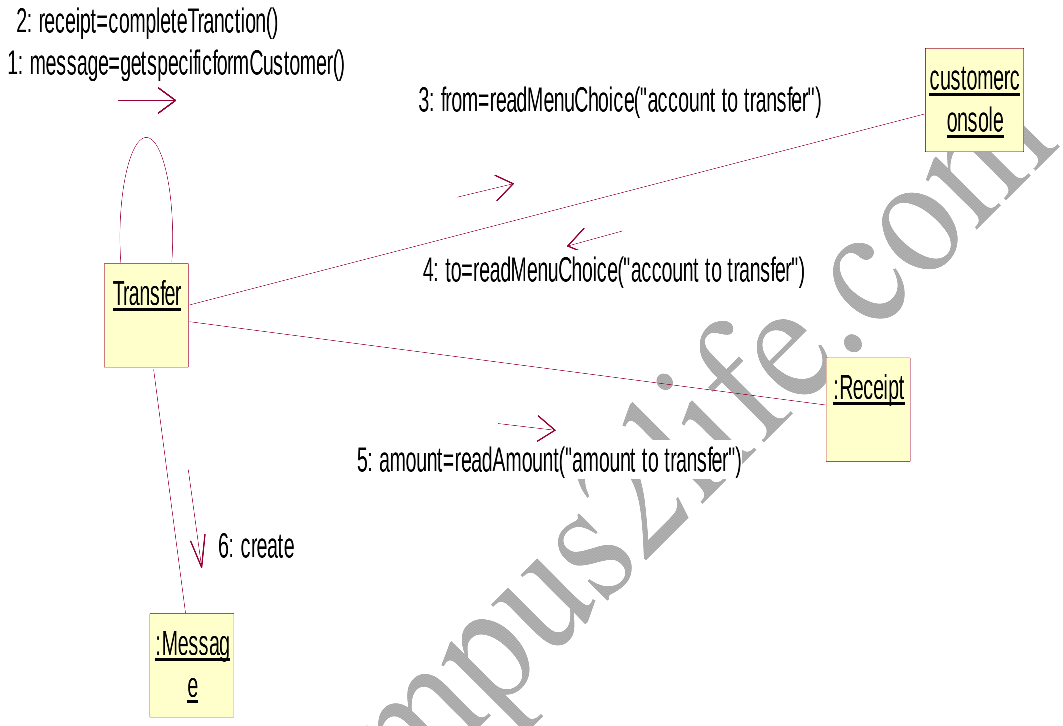
Deposit Tranction Collaboration


2: receipt=completeTransaction()
1: message=getSpecificsFormCustomer()
3: to=read menu Choice("account depositer")
:customer console
4: amount=readmenuchoice()
:Deposit
5: acceptEnvelope()
:Envelop Aceptor
6: create
7: send(message,balance)
:Receipt
8: create
:NetworkToBank
:Message

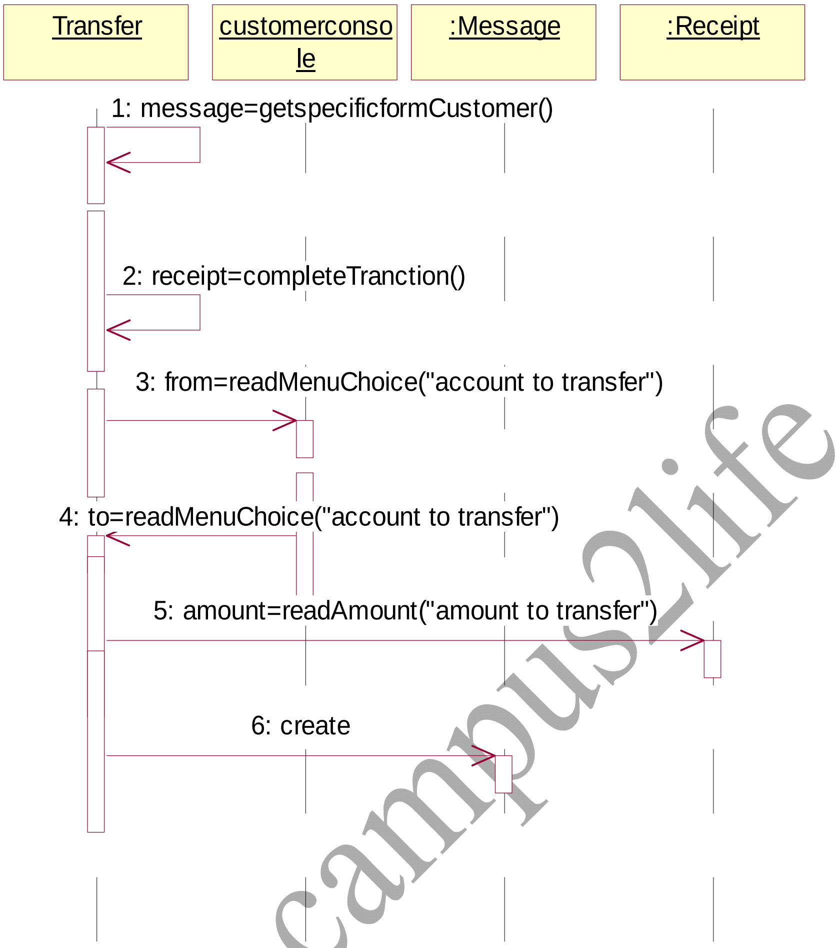
Deposit Transaction Sequence Digram


:Deposit
:customer console
:Envelop Aceptor
:Receipt
:NetworkTo Bank
:Message
1: message=getSpecificsFormCustomer()
2: receipt=completeTransaction()
3: to=read menu Choice("account depositer")
4: amount=readmenuchoice()
5: acceptEnvelope()
6: create
7: send(message,balance)
8: create

Transfer Transaction Collaboration Digram


2: receipt=completeTranction()
1: message=getspecificformCustomer()
3: from=readMenuChoice("account to transfer")
customerconsole
4: to=readMenuChoice("account to transfer")
Transfer
:Receipt
5: amount=readAmount("amount to transfer")
6: create
:Message

Transfer Transaction Sequence Digram


Transfer
customerconso
le
:Message
:Receipt
1: message=getspecificformCustomer()
2: receipt=completeTranction()
3: from=readMenuChoice("account to transfer")
4: to=readMenuChoice("account to transfer")
5: amount=readAmount("amount to transfer")
6: create

Use Cases for ATM System

System Startup Use Case

The system is started up when the operator turns the operator switch to the “on” position. The operator will be asked to enter the amount of money currently in the cash dispenser, and a connection to the bank will be established. Then the servicing of customers can begin.

System Shutdown Use Case

The system is shut down when the operator makes sure that no customer is using the machine, and then turns the operator switch to the “off” position. The connection to the bank will be shut down. Then the operator is free to remove deposited envelopes, replenish cash and paper, etc.

Session Use Case

A session is started when a customer inserts an ATM card into the card reader slot of the machine. The ATM pulls the card into the machine and reads it. (If the reader cannot read the card due to improper insertion or a damaged stripe, the card is rejected, an error screen is displayed, and the session is aborted). The customer is asked to enter his/her **PIN**, and is then allowed to perform one or more transactions, choosing from a menu of possible types of transaction in each case. After each transaction, the customer is asked whether he/she would like to perform another. When the customer is through performing transactions, the card is ejected from the machine and the session ends. If a transaction is aborted due to too many invalid **PIN** entries, the session is also aborted, with the card being retained in the machine.

The customer may abort the session by pressing the cancel key when entering a **PIN** or choosing a transaction type.

Transaction Use Case

Note: Transaction is an abstract generalization. Each specific concrete type of transaction implements certain operations in the appropriate way. The flow of events given here describes the behavior common to all types of transactions. The flows of events for the individual types of transaction (withdrawal, deposit, transfer, inquiry) give the features that are specific to that type of transaction.

A transaction use case is started within a session when the customer chooses, a transaction type from a menu of options. The customer will be asked to furnish appropriate details (e.g. account(s) involved, amount). The transaction will then be sent to the bank, along with information from the customer's card and the **PIN** the customer entered.

If the bank approves the transaction, any steps needed to complete the transaction (e.g. dispensing cash or accepting an envelope) will be performed, and then a receipt will be printed. Then the customer will be asked whether he/she wishes to do another transaction.

If the bank reports that the customer's **PIN** extension will be performed and then an attempt will be made to continue the transaction. If the customer's card is retained due to too many invalid **PIN**s, the transaction will be aborted, and the customer will not be offered the option of doing another.

If a transaction is cancelled by the customer, or fails for any reason other than repeated entries of an invalid **PIN**, a screen will be displayed informing the customer of the reason for the failure of the transaction, and then the customer will be offered the opportunity to do another.

The customer may cancel a transaction by pressing the Cancel key as described for each individual type of transaction below.

All messages to the bank and responses back are recorded in the ATM's log.

Withdrawal Transaction Use Case

A withdrawal transaction asks the customer to choose a type of account to withdraw from (e.g. checking) from a menu of possible accounts, and to choose a dollar amount from a menu of possible amounts. The system verifies that it has sufficient money on hand to satisfy the request before sending the transaction to the bank. (If not, the customer is informed and asked to enter a different amount.) If the transaction is approved by the bank, the appropriate amount of cash is dispensed by the machine before it issues a receipt. (The dispensing of cash is also recorded in the ATM's log.) A withdrawal transaction can be cancelled by the customer pressing the Cancel key any time prior to choosing the dollar amount.

Deposit Transaction Use Case

A deposit transaction asks the customer to choose a type of account to deposit to (e.g. checking) from a menu of possible accounts and to type in a dollar amount on the keyboard. The transaction is initially sent to the bank to verify that the ATM can accept a deposit from this customer to this account. If the transaction is approved, the machine accepts an envelope from the customer containing cash and/or checks before it issues a receipt. Once the envelope has been received, a second message is sent to the bank, to confirm that the bank can credit the customer's account – contingent on manual verification of the deposit envelope contents by an operator later. (The receipt of an envelope is also recorded in the ATM's log.)

A deposit transaction can be cancelled by the customer pressing the Cancel key any time prior to inserting the envelope containing the deposit. The transaction is automatically cancelled if the customer fails to insert the envelope containing the deposit within a reasonable period of time after being asked to do so.

Transfer Transaction Use Case

A transfer transaction asks the customer to choose a type of account to transfer from (e.g. checking) from a menu of possible accounts, to choose a different account to transfer to, and to type in a dollar amount on the keyboard. No further action is required once the transaction is approved by the bank before printing the receipt.

A transfer transaction can be cancelled by the customer pressing the Cancel key any time prior to entering a dollar amount.

Inquiry Transaction Use Case

An inquiry transaction asks the customer to choose a type of account to inquire about from a menu of possible accounts. No further action is required once the transaction is approved by the bank before printing the receipt.

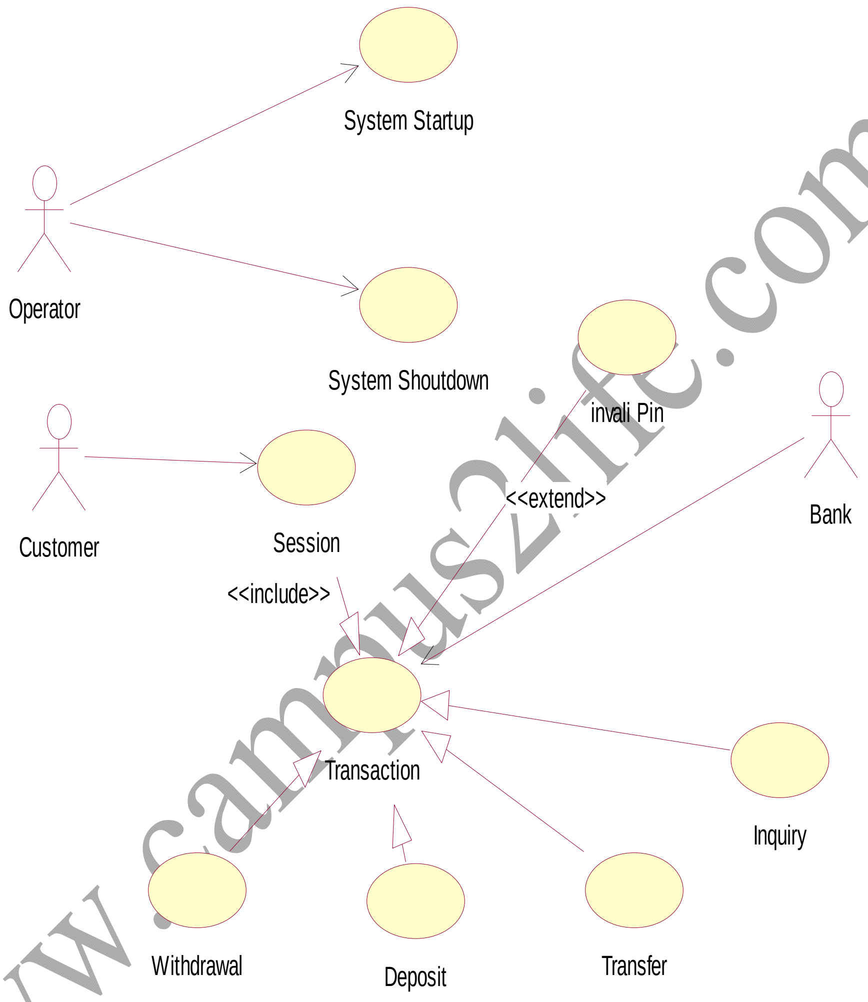
An inquiry transaction can be cancelled by the customer pressing the Cancel key any time prior to choosing the account to inquire about.

Invalid PIN Extension

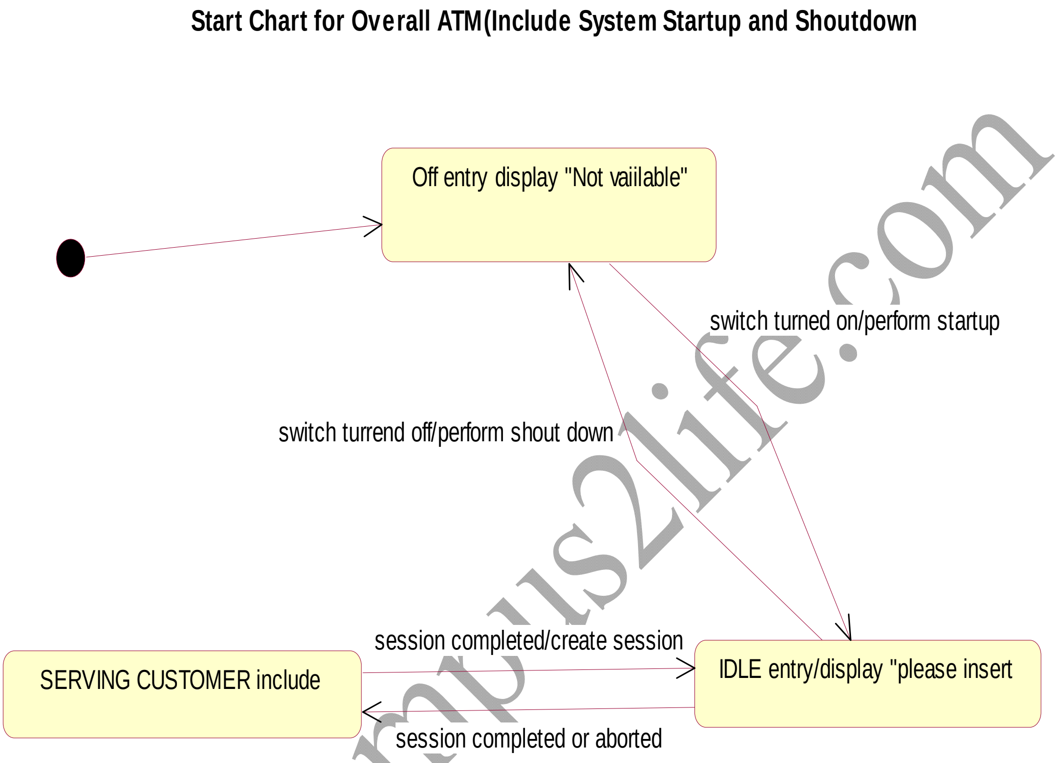
An invalid **PIN** extension is started from within a transaction when the bank reports that the customer's transaction is disapproved due to an invalid **PIN**. The customer is required to re-enter the **PIN** and the original request is sent to the bank again. If the bank now approves the transaction, or disapproves it for some other reason, the original use case is continued; otherwise the process of re-entering the **PIN** is repeated. Once the **PIN** is successfully re-entered, it is used for both the current transaction and all subsequent transactions in the session. If the customer fails three times to enter the correct **PIN**, the card is permanently retained, a screen is displayed informing the customer of this and suggesting he/she contact the bank, and the entire customer session is aborted.

If the customer presses "Cancel" instead of re-entering a **PIN** the original transaction is cancelled.

Use case for ATM



State-Chart Diagram for ATM System:


Start Chart for Overall ATM(Include System Startup and Shoutdown
Off entry display "Not vailable"
switch turned on/perform startup
switch turrend off/perform shout down
session completed/create session
SERVING CUSTOMER include
IDLE entry/display "please insert
session completed or aborted

- Three of the objects we have identified have behavior that is sufficiently complex to warrant developing a State Chart for them. These are the objects that were identified as the major controlled Objects. Responsible for the System Startup and Shutdown use case, responsible for the Session use case, responsible for the Transaction use case, use case for the specific types of transaction, and Invalid PIN extinction.

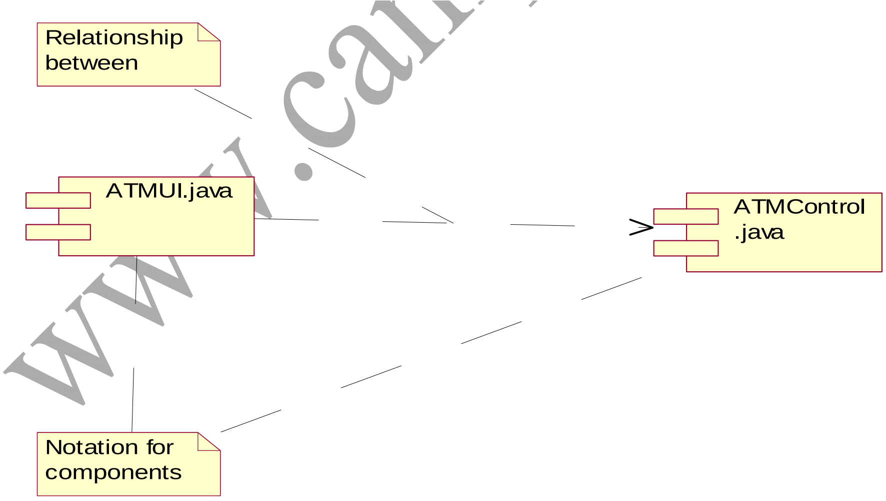
- In its basic form, an activity diagram is a simple and intuitive illustration of what happens in a workflow, what activity can be done in parallel, and whether there are alternative paths throw the workflow.

- In the first stage of identifying objects and classes based on the use cases(use-case analysis), activity diagrams can be useful when exploring responsibilities of analysis classes. You might use the activity diagrams techniques to draw a first sketch of class responsibilities, a sketch that you then throw away.

Component diagram for ATM Systems:

A component diagram describes the organization of the physical component in a system.

UML Component Diagram



Deployment Diagram for ATM systems:

The deployment view shows the physical arrangements of nodes.
Node is a Runtime computational resource such as computer or other device.

UML Deployment Diagrams

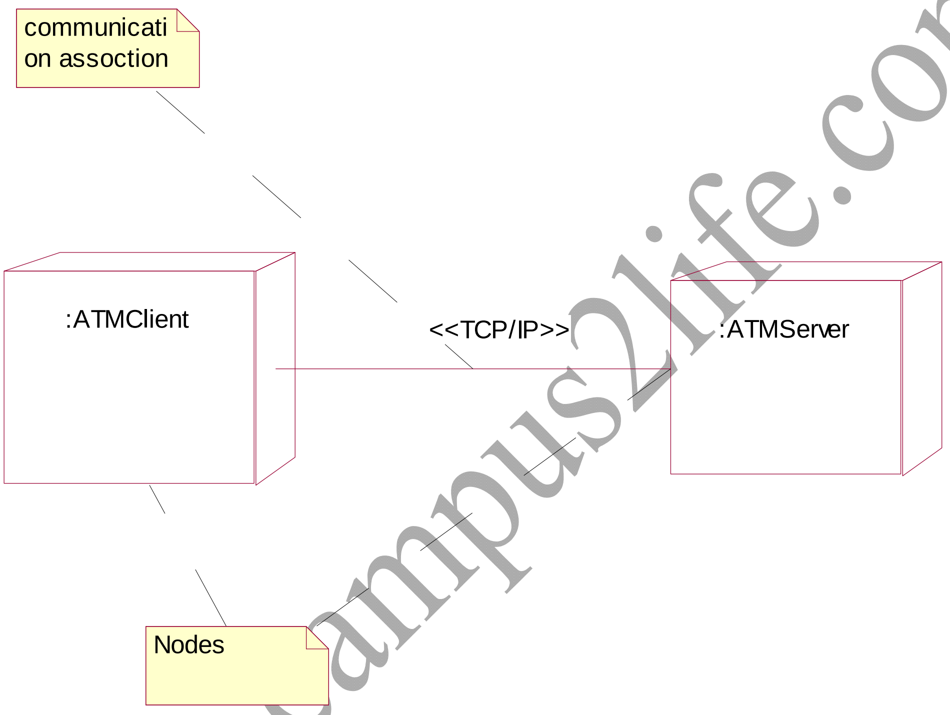


Table of contents: