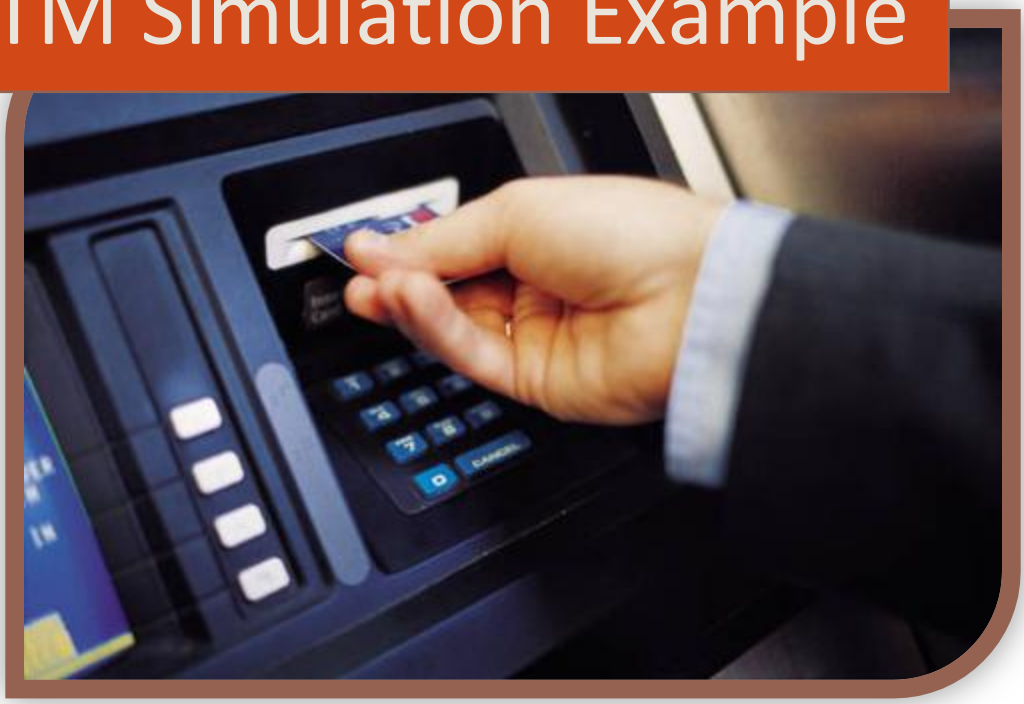
ATM Simulation Example



TABBLE OF CONTENTS

I. Requirement Specification

1. Requirements Statement for Example ATM System

The software to be designed will control a simulated automated teller machine (ATM) having a magnetic stripe reader for reading an ATM card, a customer console (keyboard and display) for interaction with the customer, a slot for depositing envelopes, a dispenser for cash (in multiples of $20), a printer for printing customer receipts, and a key-operated switch to allow an operator to start or stop the machine. The ATM will communicate with the bank's computer over an appropriate communication link. (The software on the latter is not part of the requirements for this problem.)

The ATM will service one customer at a time. A customer will be required to insert an ATM card and enter a personal identification number (PIN) - both of which will be sent to the bank for validation as part of each transaction. The customer will then be able to perform one or more transactions. The card will be retained in the machine until the customer indicates that he/she desires no further transactions, at which point it will be returned - except as noted below.

The ATM must be able to provide the following services to the customer:

1. A customer must be able to make a cash withdrawal from any suitable account linked to the card, in multiples of $20.00. Approval must be obtained from the bank before cash is dispensed.
2. A customer must be able to make a deposit to any account linked to the card, consisting of cash and/or checks in an envelope. The customer will enter the amount of the deposit into the ATM, subject to manual verification when the envelope is removed from the machine by an operator. Approval must be obtained from the bank before physically accepting the envelope.
3. A customer must be able to make a transfer of money between any two accounts linked to the card.
4. A customer must be able to make a balance inquiry of any account linked to the card.

A customer must be able to abort a transaction in progress by pressing the Cancel key instead of responding to a request from the machine.

The ATM will communicate each transaction to the bank and obtain verification that it was allowed by the bank. Ordinarily, a transaction will be considered complete by the bank once it

has been approved. In the case of a deposit, a second message will be sent to the bank indicating that the customer has deposited the envelope. (If the customer fails to deposit the envelope within the timeout period, or presses cancel instead, no second message will be sent to the bank and the deposit will not be credited to the customer.)

If the bank determines that the customer's PIN is invalid, the customer will be required to re-enter the PIN before a transaction can proceed. If the customer is unable to successfully enter the PIN after three tries, the card will be permanently retained by the machine, and the customer will have to contact the bank to get it back.

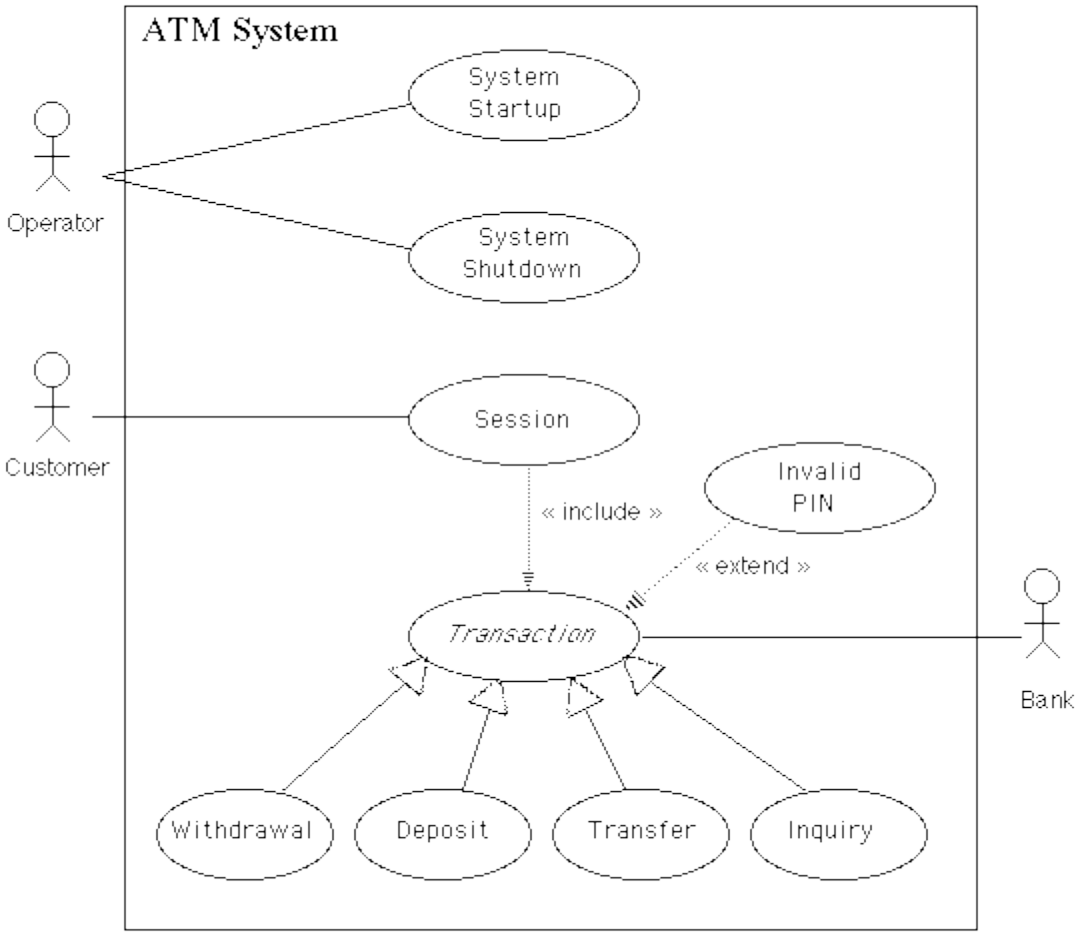
If a transaction fails for any reason other than an invalid PIN, the ATM will display an explanation of the problem, and will then ask the customer whether he/she wants to do another transaction.

The ATM will provide the customer with a printed receipt for each successful transaction, showing the date, time, machine location, type of transaction, account(s), amount, and ending and available balance(s) of the affected account ("to" account for transfers).

The ATM will have a key-operated switch that will allow an operator to start and stop the servicing of customers. After turning the switch to the "on" position, the operator will be required to verify and enter the total cash on hand. The machine can only be turned off when it is not servicing a customer. When the switch is moved to the "off" position, the machine will shut down, so that the operator may remove deposit envelopes and reload the machine with cash, blank receipts, etc.

The ATM will also maintain an internal log of transactions to facilitate resolving ambiguities arising from a hardware failure in the middle of a transaction. Entries will be made in the log when the ATM is started up and shut down, for each message sent to the Bank (along with the response back, if one is expected), for the dispensing of cash, and for the receiving of an envelope. Log entries may contain card numbers and dollar amounts, but for security will *never* contain a PIN.

2. Use Cases for Example ATM System

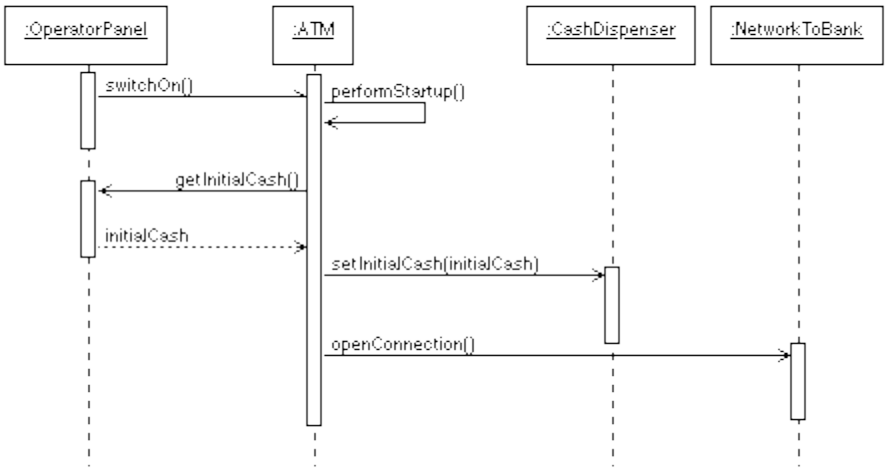


3. Flows of Events for Individual Use Cases

a. 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.

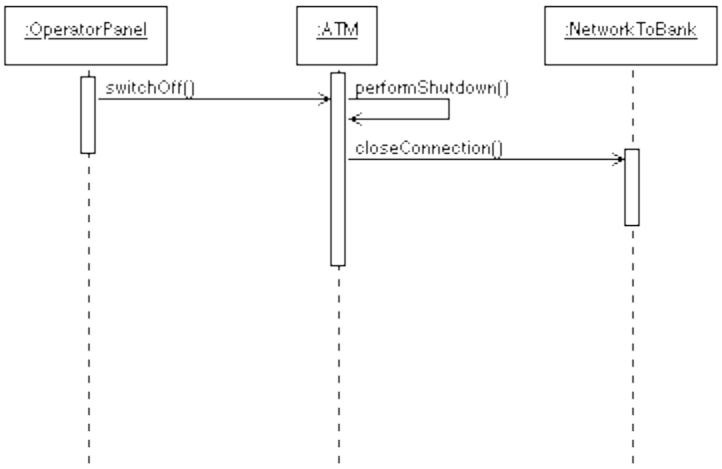
System Startup Sequence Diagram



b. 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.

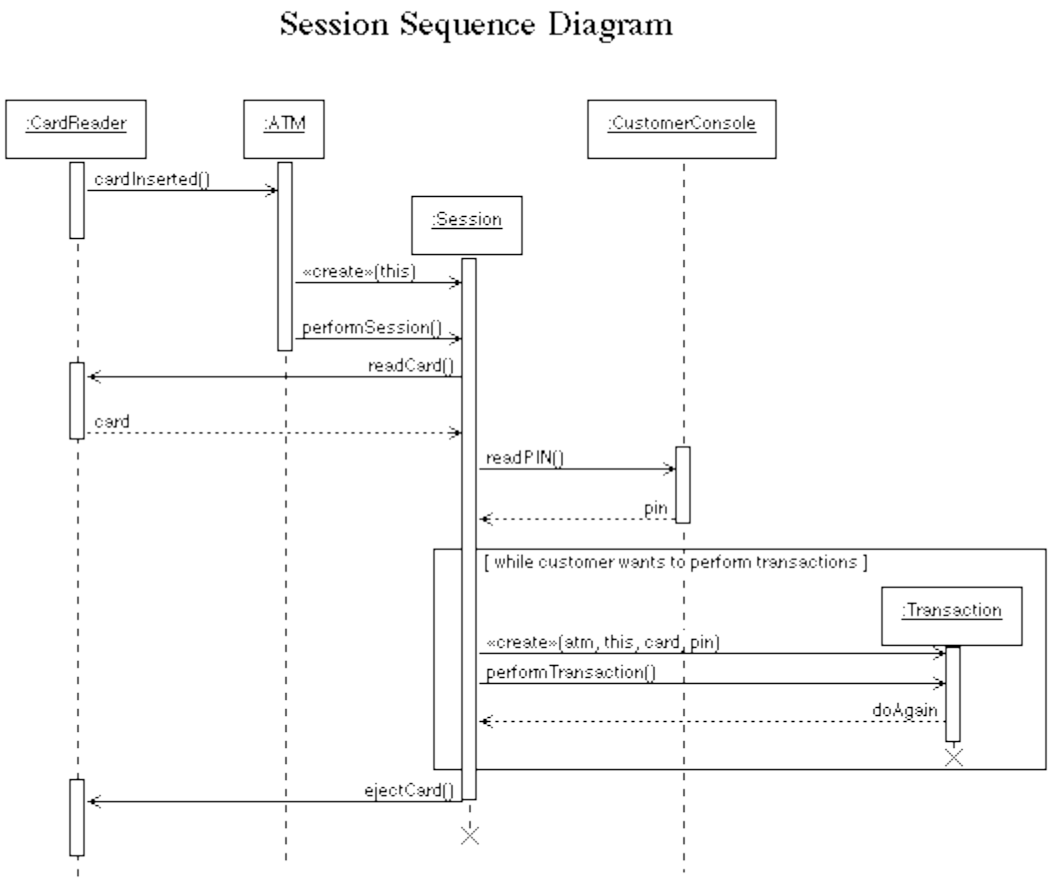
System Shutdown Sequence Diagram



c. 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 ejected, 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.



d. 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 transaction. 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.

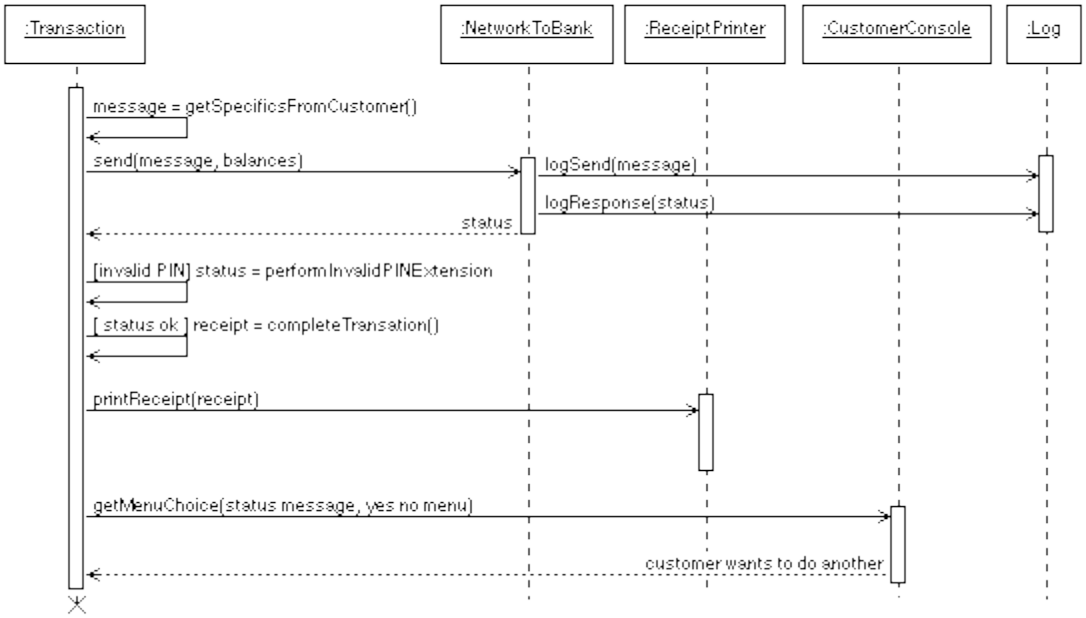
If the bank reports that the customer's PIN is invalid, the Invalid PIN extension will be performed and then an attempt will be made to continue the transaction. If the customer's card is retained due to too many invalid PINs, the transaction will be aborted, and the customer will not be offered the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

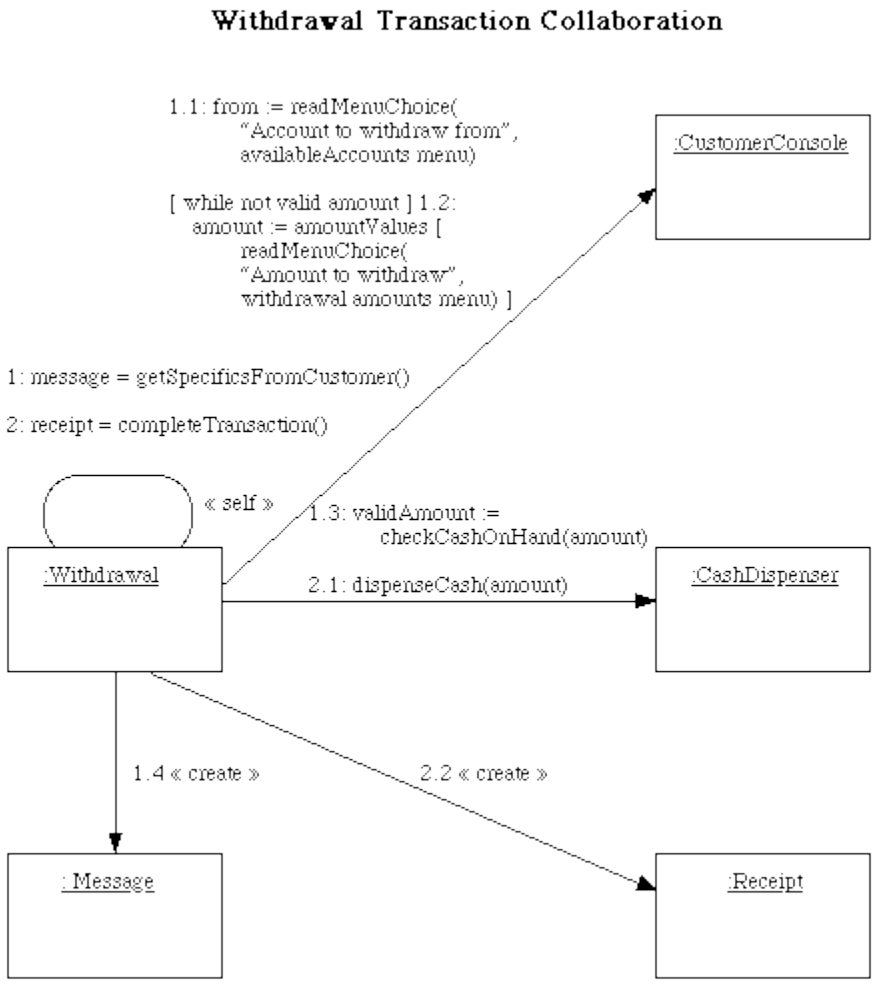
Transaction Sequence Diagram



e. 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.)

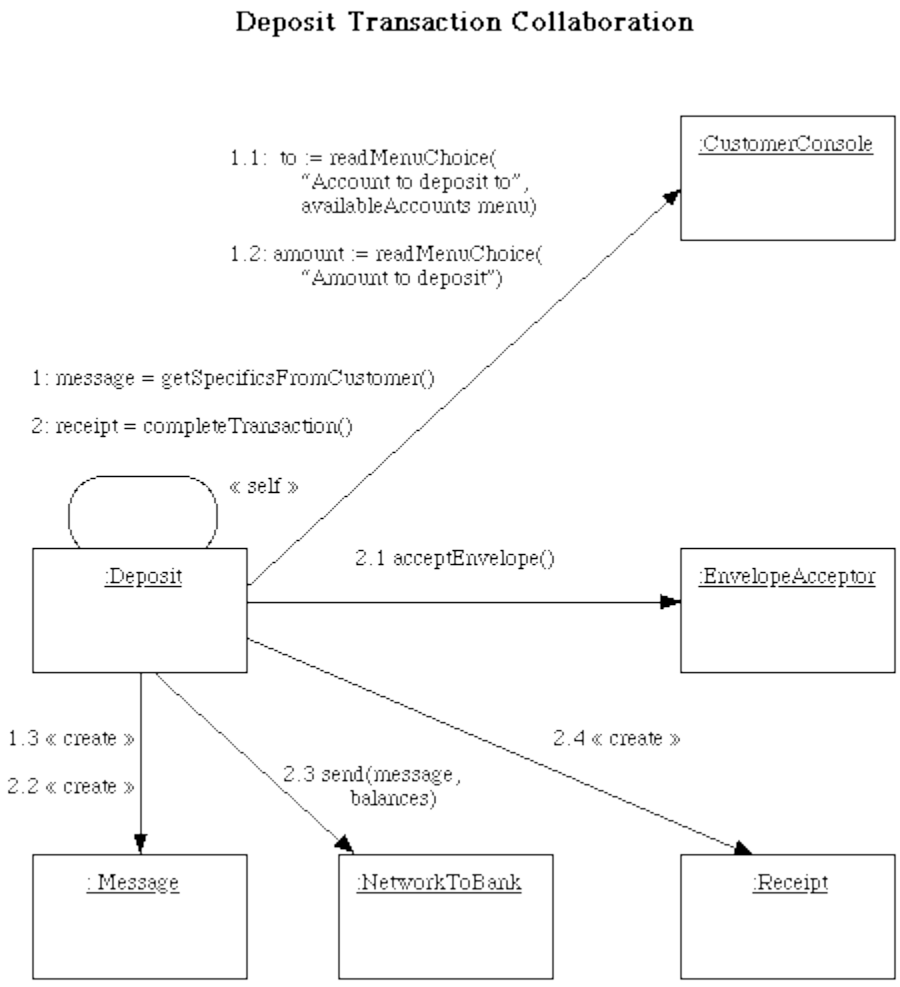
A withdrawal transaction can be cancelled by the customer pressing the Cancel key any time prior to choosing the dollar amount.



f. 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.)

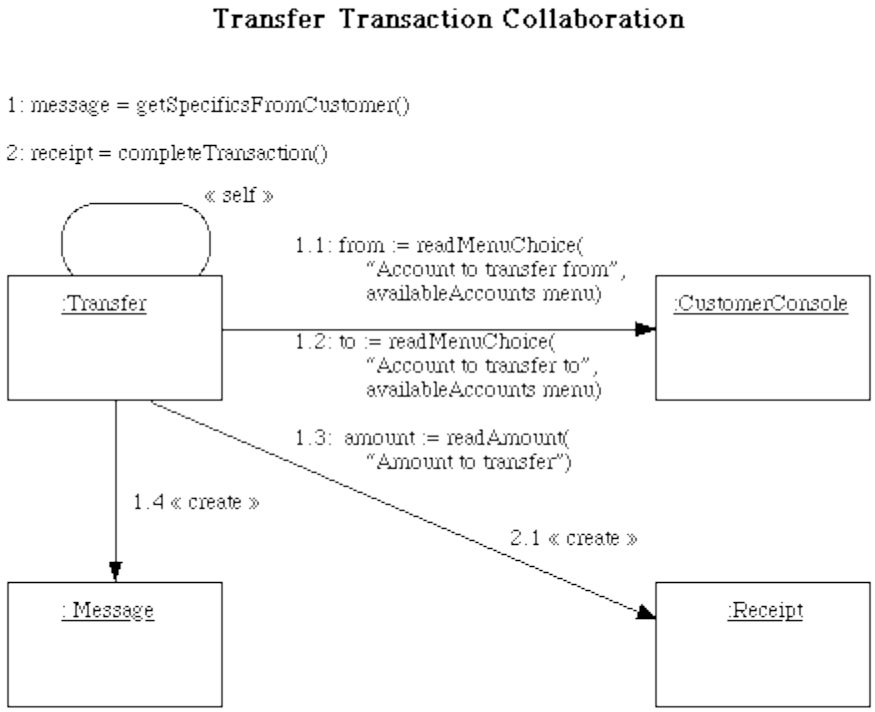
A deposit transaction can be cancelled by the customer pressing the Cancel key any time prior to inserting the envelope containing the deposit. The transaction is automatically cancelled if the customer fails to insert the envelope containing the deposit within a reasonable period of time after being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



g. 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.

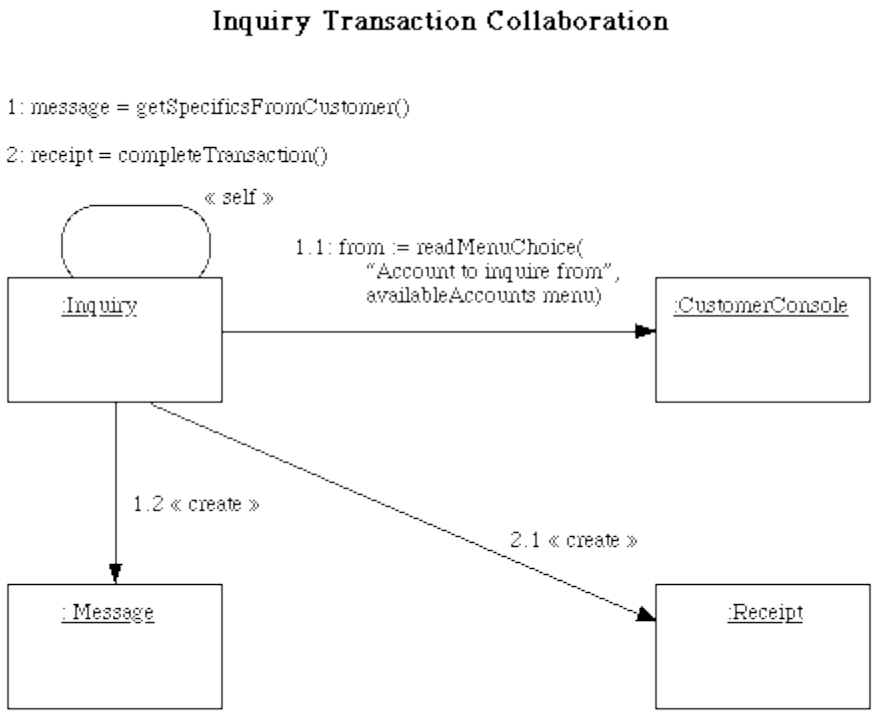
A transfer transaction can be cancelled by the customer pressing the Cancel key any time prior to entering a dollar amount.



h. 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.

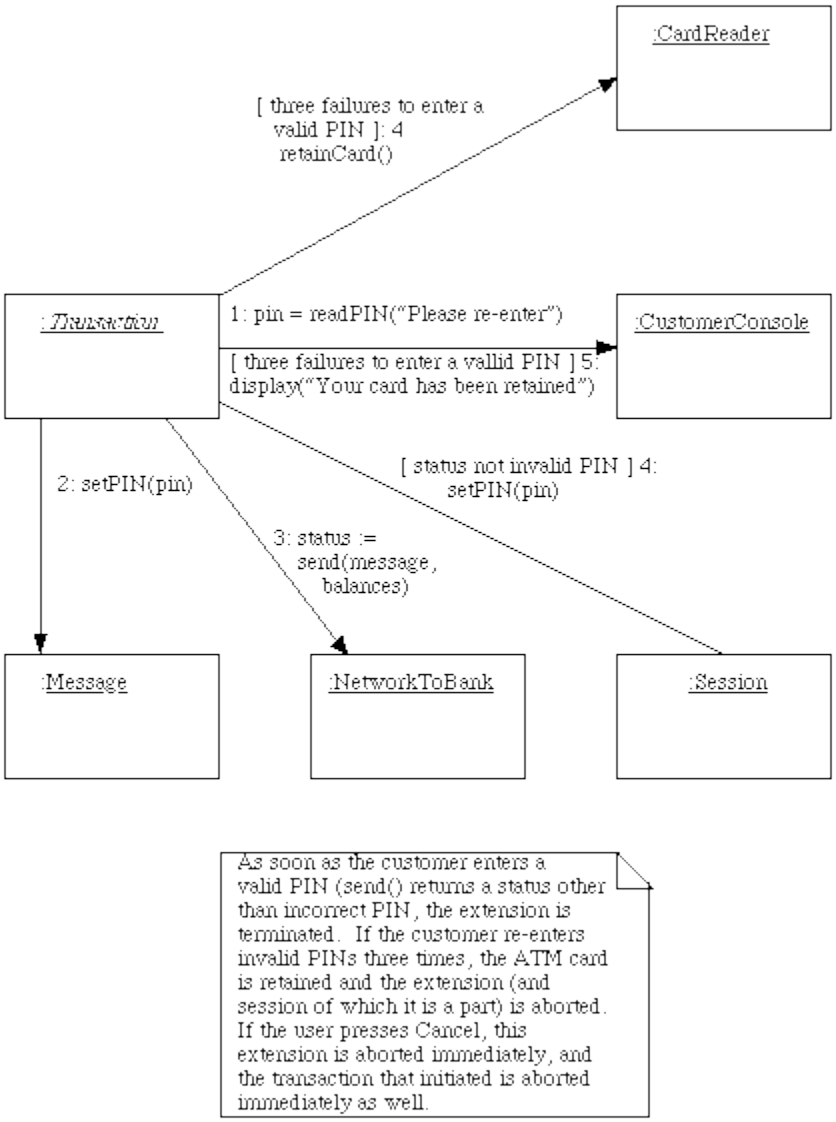


i. Invalid PIN Extension

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.

Invalid PIN Extension Collaboration



4. Sequence based specification

a. Abstraction level

The transactions are considered at the general level, i.e. only the Transaction Use Case is considered and not detailed ones like Inquiry Transaction Use Case or Deposit Transaction Use Case. All other use cases are considered.

The transaction information are always correct, i.e. the only reason for a transaction not working is an incorrect PIN.

b. Requirement Tags

Tags	Requirements
1	System Startup Use Case
2	System Shutdown Use Case
3	Session Use Case
4	Transaction Use Case
5	Invalid PIN Extension

c. Stimuli

Stimulus	Description	Symbol	Trace
Switch turned ON	Operator turns the switch to the "ON" position	SON	1
Enter initial cash	Operator enters the amount of money initially in the cash dispenser	INIT	1
Switch turned OFF	Operator turns the switch to the "OFF" position	SOFF	2
ATM card is inserted properly	A customer inserts properly an ATM card into the card reader slot of the machine	GOODCARD	3
ATM card is inserted improperly	A customer inserts improperly an ATM card into the card reader slot of the machine	BADCARD	3
Cancel key is pressed	A customer pressed the cancel key	CANCEL	3
Good pin is entered		GOODPIN	3
Bad pin is entered		BADPIN	3
Transaction details are entered		TRANSINFO	

d. Responses

Response	Description	Trace
Wait for initial cash	Operator is asked to enter the amount of money initially in the cash dispenser	1
Servicing starts	A connection to the bank will be established. Then the servicing of customers can begin	1
Servicing stops	The connection to the bank will be shut down	2
Session starts	The ATM pulls the card into the machine and reads it and the customer is asked to enter his/her PIN	3
Card is ejected		3
Session is aborted		3
Transaction starts		4
Receipt is printed		4
Customer's card is retained		4
Transaction is cancelled		4

e. Enumeration of sequences

Only sequences of length 0, 1, 2 and 3 are enumerated. You will be asked in an exercise class to continue the enumeration.

Sequence	Response	Equivalence	Trace
λ	Null		D1
D1	The ATM is initially off		

Sequence	Response	Equivalence	Trace
SON	Wait for initial cash		1
INIT	Illegal		D1
SOFF	Illegal		D1
GOODCARD	Illegal		D1
BADCARD	Illegal		D1
CANCEL	Illegal		D1
GOODPIN	Illegal		D1
BADPIN	Illegal		D1
TRANSINFO	Illegal		D1

Sequence	Response	Equivalence	Trace
SON SON	null	SON	D2
SON INIT	Servicing starts		1
SON SOFF	Servicing stops	λ	2
SON GOODCARD	null	SON	D3
SON BADCARD	null	SON	D3
SON CANCEL	null	SON	D3
SON GOODPIN	null	SON	D3
SON BADPIN	null	SON	D3
SON TRANSINFO	null	SON	D3
D2	After the switch is turned on, its physically impossible to turn it on again		
D3	After the switch is turned on, the only valid Stimuli is INIT and SOFF		

Sequence	Response	Equivalence	Trace
SON INIT SON	null	SON INIT	D2
SON INIT INIT	null	SON INIT	D4
SON INIT SOFF	null	λ	D1
SON INIT GOODCARD	Session starts		3
SON INIT BADCARD	Card is ejected	SON INIT	3
SON INIT CANCEL	null	SON INIT	D5
SON INIT GOODPIN	null	SON INIT	D5
SON INIT BADPIN	null	SON INIT	D5
SON INIT TRANSINFO	null	SON INIT	D5
D4	Initial cash entry is only available once during the startup		
D5	As long as no session has started, any press on the keyboard is ignored		

5. Initial Functional Test Cases for Example ATM System

The following initial test cases can be identified early in the design process as a vehicle for checking that the implementation is basically correct. *No attempt has been made at this point to do thorough testing, including all possible errors and boundary cases.* That needs to come later. These cases represent an initial check that the functionality specified by the use cases is present.

Some writers would argue for developing test cases like these *in place of* use cases. Here, they are presented as a vehicle for "fleshing out" the use cases, not as a substitute for them.

Use Case	Function Being Tested	Initial System State	Input	Expected Output
System Startup	System is started when the switch is turned "on"	System is off	Activate the "on" switch	System requests initial cash amount
System Startup	System accepts initial cash amount	System is requesting cash amount	Enter a legitimate amount	System is on
System Startup	Connection to the bank is established	System has just been turned on	Perform a legitimate inquiry transaction	System output should demonstrate that a connection has been established to the Bank
System Shutdown	System is shut down when the switch is turned "off"	System is on and not servicing a customer	Activate the "off" switch	System is off
System Shutdown	Connection to the Bank is terminated when the system is shut down	System has just been shut down		Verify from the bank side that a connection to the ATM no longer exists
Session	System reads a customer's ATM card	System is on and not servicing a customer	Insert a readable card	Card is accepted; System asks for entry of PIN

Use Case	Function Being Tested	Initial System State	Input	Expected Output
System Startup	System is started when the switch is turned "on"	System is off	Activate the "on" switch	System requests initial cash amount
Session	System rejects an unreadable card	System is on and not servicing a customer	Insert an unreadable card	Card is ejected; System displays an error screen; System is ready to start a new session
Session	System accepts customer's PIN	System is asking for entry of PIN	Enter a PIN	System displays a menu of transaction types
Session	System allows customer to perform a transaction	System is displaying menu of transaction types	Perform a transaction	System asks whether customer wants another transaction
Session	System allows multiple transactions in one session	System is asking whether customer wants another transaction	Answer yes	System displays a menu of transaction types
Session	Session ends when customer chooses not to do another transaction	System is asking whether customer wants another transaction	Answer no	System ejects card and is ready to start a new session
Transaction	Individual types of transaction will be tested below			
Transaction	System handles an invalid PIN properly	A readable card has been entered	Enter an incorrect PIN and then attempt a transaction	The Invalid PIN Extension is performed
Withdrawal	System asks customer to choose an account to	Menu of transaction types is being	Choose Withdrawal transaction	System displays a menu of account types

Use Case	Function Being Tested	Initial System State	Input	Expected Output
System Startup	System is started when the switch is turned "on"	System is off	Activate the "on" switch	System requests initial cash amount
	withdraw from	displayed		
Withdrawal	System asks customer to choose a dollar amount to withdraw	Menu of account types is being displayed	Choose checking account	System displays a menu of possible withdrawal amounts
Withdrawal	System performs a legitimate withdrawal transaction properly	System is displaying the menu of withdrawal amounts	Choose an amount that the system currently has and which is not greater than the account balance	System dispenses this amount of cash; System prints a correct receipt showing amount and correct updated balance; System records transaction correctly in the log (showing both message to the bank and approval back)
Withdrawal	System verifies that it has sufficient cash on hand to fulfill the request	System has been started up with less than the maximum withdrawal amount in cash on hand; System is requesting a withdrawal amount	Choose an amount greater than what the system currently has	System displays an appropriate message and asks customer to choose a different amount
Withdrawal	System verifies that customer's balance is sufficient to fulfill the request	System is requesting a withdrawal ammount	Choose an amount that the system currently has but which is greater than the account balance	System displays an appropriate message and offers customer the option of choosing to do another transaction

Use Case	Function Being Tested	Initial System State	Input	Expected Output
System Startup	System is started when the switch is turned "on"	System is off	Activate the "on" switch	System requests initial cash amount
				or not.
Withdrawal	A withdrawal transaction can be cancelled by the customer any time prior to choosing the dollar amount	System is displaying menu of account types	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
Withdrawal	A withdrawal transaction can be cancelled by the customer any time prior to choosing the dollar amount	System is displaying menu of dollar amounts	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
Deposit	System asks customer to choose an account to deposit to	Menu of transaction types is being displayed	Choose Deposit transaction	System displays a menu of account types
Deposit	System asks customer to enter a dollar amount to deposit	Menu of account types is being displayed	Choose checking account	System displays a request for the customer to type a dollar amount
Deposit	System asks customer to insert an envelope	System is displaying a request for the customer to type a dollar amount	Enter a legitimate dollar amount	System requests that customer insert an envelope
Deposit	System performs a legitimate deposit transaction properly	System is requesting that customer insert an envelope	Insert an envelope	System accepts envelope; System prints a correct receipt showing amount and correct updated

Use Case	Function Being Tested	Initial System State	Input	Expected Output
System Startup	System is started when the switch is turned "on"	System is off	Activate the "on" switch	System requests initial cash amount
				balance; System records transaction correctly in the log (showing message to the bank, approval back, and acceptance of the envelope)
Deposit	A deposit transaction can be cancelled by the customer any time prior to inserting an envelope	System is displaying menu of account types	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
Deposit	A deposit transaction can be cancelled by the customer any time prior to inserting an envelope	System is requesting customer to enter a dollar amount	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
Deposit	A deposit transaction can be cancelled by the customer any time prior to inserting an envelope	System is requesting customer to insert an envelope	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
Deposit	A deposit transaction is cancelled automatically if an envelope is not inserted within a	System is requesting customer to insert an envelope	Wait for the request to time out	System displays an appropriate message and offers customer the option of choosing to do another transaction

Use Case	Function Being Tested	Initial System State	Input	Expected Output
System Startup	System is started when the switch is turned "on"	System is off	Activate the "on" switch	System requests initial cash amount
	reasonable time			or not.
Transfer	System asks customer to choose an account to transfer from	Menu of transaction types is being displayed	Choose Transfer transaction	System displays a menu of account types specifying transfer from
Transfer	System asks customer to choose an account to transfer to	Menu of account types to transfer from is being displayed	Choose checking account	System displays a menu of account types specifying transfer to
Transfer	System asks customer to enter a dollar amount to transfer	Menu of account types to transfer to is being displayed	Choose savings account	System displays a request for the customer to type a dollar amount
Transfer	System performs a legitimate transfer transaction properly	System is displaying a request for the customer to type a dollar amount	Enter a legitimate dollar amount	System prints a correct receipt showing amount and correct updated balance; System records transaction correctly in the log (showing both message to the bank and approval back)
Transfer	A transfer transaction can be cancelled by the customer any time prior to entering dollar amount	System is displaying menu of account types specifying transfer from	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
Transfer	A transfer transaction can be cancelled by the	System is displaying menu of account types	Press "Cancel" key	System displays an appropriate message and offers customer

Use Case	Function Being Tested	Initial System State	Input	Expected Output
System Startup	System is started when the switch is turned "on"	System is off	Activate the "on" switch	System requests initial cash amount
	customer any time prior to entering dollar amount	specifying transfer to		the option of choosing to do another transaction or not.
Transfer	A transfer transaction can be cancelled by the customer any time prior to entering dollar amount	System is requesting customer to enter a dollar amount	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
Inquiry	System asks customer to choose an account to inquire about	Menu of transaction types is being displayed	Choose Inquiry transaction	System displays a menu of account types
Inquiry	System performs a legitimate inquiry transaction properly	System is displaying menu of account types	Choose checking account	System prints a correct receipt showing correct balance; System records transaction correctly in the log (showing both message to the bank and approval back)
Inquiry	An inquiry transaction can be cancelled by the customer any time prior to choosing an account	System is displaying menu of account types	Press "Cancel" key	System displays an appropriate message and offers customer the option of choosing to do another transaction or not.
Invalid PIN Extension	Customer is asked to reenter PIN		Enter an incorrect PIN;	Customer is asked to re-enter PIN

Use Case	Function Being Tested	Initial System State	Input	Expected Output
System Startup	System is started when the switch is turned "on"	System is off	Activate the "on" switch	System requests initial cash amount
			Attempt an inquiry transaction on the customer's checking account	
Invalid PIN Extension	Correct re-entry of PIN is accepted	Request to re-enter PIN is being displayed	Enter correct PIN	Original transaction completes successfully
Invalid PIN Extension	A correctly re-entered PIN is used for subsequent transactions	An incorrect PIN has been re-entered and transaction completed normally	Perform another transaction	This transaction completes successfully as well
Invalid PIN Extension	Incorrect re-entry of PIN is not accepted	Request to re-enter PIN is being displayed	Enter incorrect PIN	An appropriate message is displayed and re-entry of the PIN is again requested
Invalid PIN Extension	Correct re-entry of PIN on the second try is accepted	Request to re-enter PIN is being displayed	Enter incorrect PIN the first time, then correct PIN the second time	Original transaction completes successfully
Invalid PIN Extension	Correct re-entry of PIN on the third try is accepted	Request to re-enter PIN is being displayed	Enter incorrect PIN the first time and second times, then correct PIN the third time	Original transaction completes successfully
Invalid PIN Extension	Three incorrect re-entries of PIN result in retaining card and aborting	Request to re-enter PIN is being displayed	Enter incorrect PIN three times	An appropriate message is displayed; Card is retained by machine;

Use Case	Function Being Tested	Initial System State	Input	Expected Output
System Startup	System is started when the switch is turned "on"	System is off	Activate the "on" switch	System requests initial cash amount
	transaction			Session is terminated

II. Design

6. Class descriptions

Class	Responsibilities	Collaboration
ATM	Start up when switch is turned on	OperatorPanel CashDispenser NetworkToBank
	Shut down when switch is turned off	NetworkToBank
	Start a new session when card is inserted by customer	CustomerConsole Session
	Provide access to component parts for sessions and transactions	
CardReader	Tell ATM when card is inserted	ATM
	Read information from card	Card
	Eject card	
	Retain card	
CashDispenser	Keep track of cash on hand, starting with initial amount	
	Report whether enough cash is available	
	Dispense cash	Log
CustomerConsole	Display a message	
	Display a prompt, accept a PIN from keyboard	
	Display a prompt and menu, accept a choice from keyboard	
	Display a prompt, accept a dollar amount from keyboard	
	Respond to cancel key being pressed by customer	
EnvelopeAcceptor	Accept envelope from customer; report if timed out or cancelled	Log
Log	Log messages sent to bank	
	Log responses from bank	
	Log dispensing of cash	
	Log receiving an envelope	
NetworkToBank	Initiate connection to bank at startup	
	Send message to bank and wait for response	Message Log Balances Status
	Terminate connection to bank at shutdown	
OperatorPanel	Inform ATM of changes to state of switch	ATM

	Allow operator to specify amount of initial cash	
ReceiptPrinter	Print receipt	Receipt
Session	Perform session use case	ATM CardReader Card CustomerConsole Transaction
	Update PIN value if customer has to re-enter it	
Transaction (Abstract class)	Allow customer to choose a type of transaction	ATM CustomerConsole Withdrawal Deposit Transfer Inquiry
	Perform Transaction Use Case	ATM CustomerConsole Withdrawal Deposit Transfer Inquiry Message NetworkToBank Receipt ReceiptPrinter
	Perform invalid PIN extension	CustomerConsole Session CardReader
Withdrawal	Perform operations peculiar to withdrawal transaction use case	CustomerConsole CashDispenser Message Receipt
Deposit	Perform operations peculiar to deposit transaction use case	CustomerConsole Message EnvelopeAcceptor Receipt
Transfer	Perform operations peculiar to transfer transaction use case	CustomerConsole Message Receipt
Inquiry	Perform operations peculiar to inquiry transaction use case	CustomerConsole Message Receipt

Balances	Represent account balance information returned by bank	
Card	Represent information encoded on customer's ATM card	
Message	Represent information to be sent over network to bank	
Receipt	Represent information to be printed on a receipt	
Status	Represent transaction status information returned by bank	

7. Structural view: Class Diagram for Example ATM System

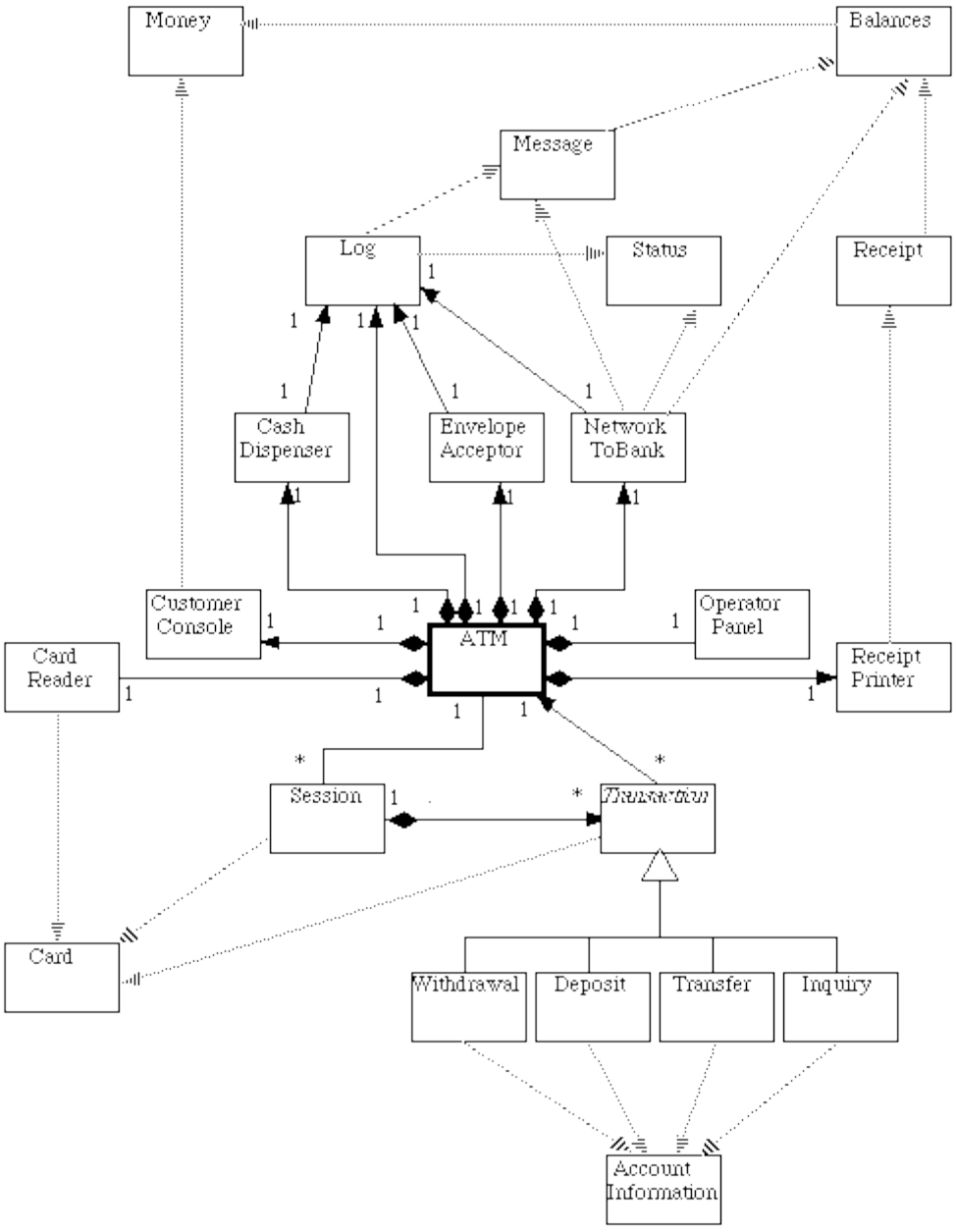
Shown below is the class diagram for the ATM system. The basic structure of the class diagram arises from the responsibilities and relationships discovered when doing the CRC cards and Interaction Diagrams. (If a class uses another class as a collaborator, or sends a message to an object of that class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

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.

That is, OO design is not a "waterfall" process - discoveries made when doing detailed design and coding can impact overall system design.

To prevent the diagram from becoming overly large, only the name of each class is shown - the attribute and behavior "compartments" are shown in the detailed design, but are omitted here.

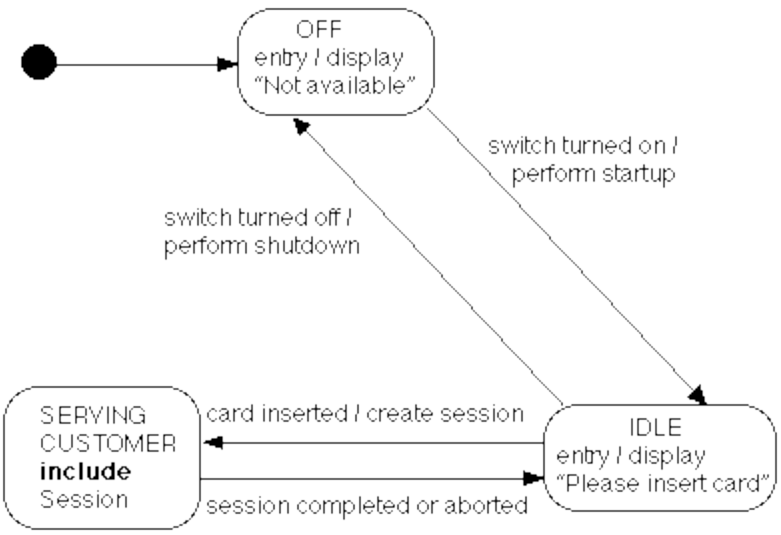


8. State Charts for Example ATM System

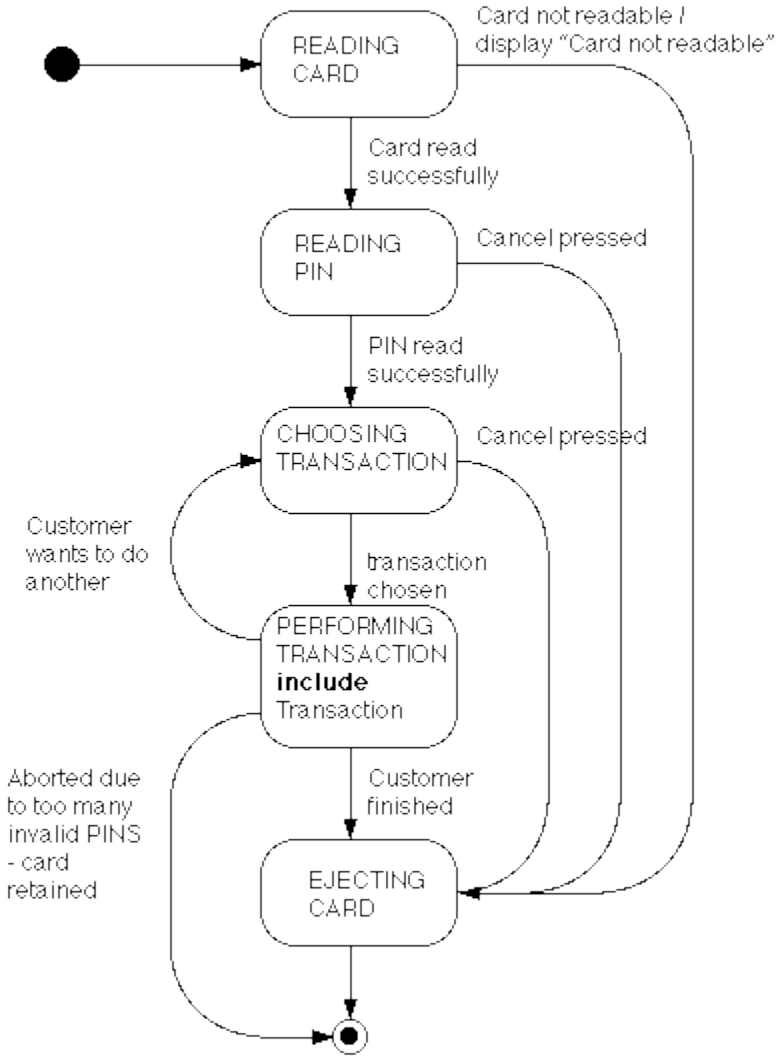
Three of the objects we have identified have behavior that is sufficiently complex to warrant developing a State Chart for them. (These are the objects that were identified as the major controller objects.)

- The object representing the machine itself (responsible for the System Startup and Shutdown use cases)
- Objects representing a customer session (one per session) (responsible for the Session use case)
- Objects representing an individual transaction (one per transaction) (responsible for the Transaction use case, use cases for the specific types of transaction, and Invalid PIN extension).

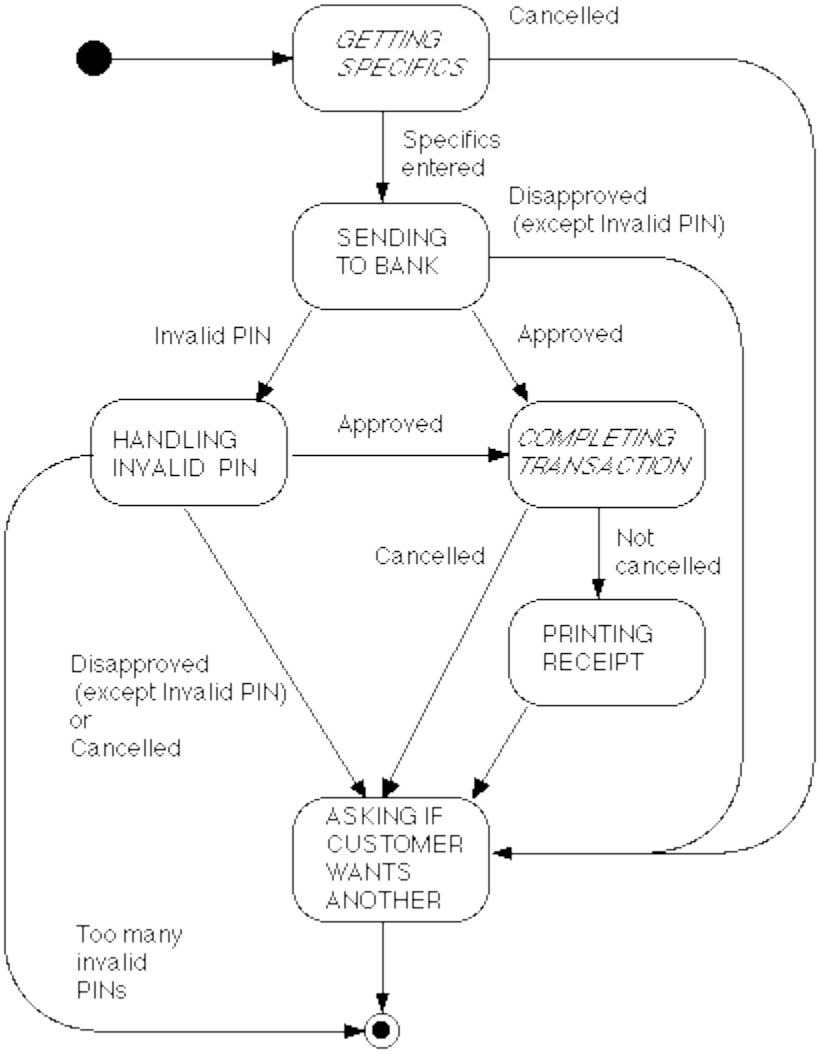
State-Chart for Overall ATM (includes System Startup and System Shutdown Use Cases)



State-Chart for One Session



State-Chart for One Transaction
(italicized operations are unique to each particular type of transaction)



III. Implementation

1. Detailed Design

A major task of detailed design is to spell out, in detail, the attributes and methods needed by each class (the second and third "compartments" of the representation for each class in a class diagram.)

The methods needed by each class are implicit in the responsibilities assigned to the class in the CRC cards, and become explicit in the Interaction Diagrams. A responsibility listed for a class on its CRC card generally maps into a method or methods in the detailed design. Likewise, any time an object belonging to a given class is shown as the recipient of a message in either a Sequence or Collaboration Diagram, the class needs a corresponding method. Many of the needed attributes are also either explicitly or implicitly present in the diagrams; the need for others becomes evident as the code for the class is being written. (Thus detailed design and coding are a "round trip" process - detailed design dictates coding, and coding leads to elaboration of the detailed design.)

In designing this system, a few key design decisions were made:

- The class ATM is made an active class - that is, the ATM object has its own thread. Using the Java thread facllity leads to defining a run() method in this class whose body is executed by the ATM's thread. The fact that class ATM is active is indicated in class diagrams by enclosing it in a heavier outline.

- Certain signals initiate computation - e.g. the signals from the operator console when the state of the switch is changed, or from the card reader when a card is inserted. In the GUI simulation of the ATM, these signals are sent by the "actionPerformed()" method of the appropriate GUI button; in a real ATM they would be sent by the physical components themselves, which might then also need to be active classes. (Note: this forms an exception to the rule that a responsibility on a CRC card translates into a method in the design - in this case the class sends a signal, rather than receiving it, so it does not need a method directly corresponding to the responsibility.)

- The Transaction hierarchy consists of the abstract class Transaction and four concrete subclasses (Withdrawal, Deposit, Transfer and Inquiry). The class Transaction has a "virtual constructor" called makeTransaction() which asks the customer to choose a transaction type and then constructs and returns an object

of the appropriate subclass. The Transaction class is made responsible for carrying out the Transaction use case and the Invalid PIN extension; for the former, it makes use of abstract methods getSpecificsFromCustomer() and completeTransaction() which are implemented concretely by each subclass.

- The class Receipt is abstract. The completeTransaction() method of each kind of transaction creates a concrete instance that contains the information relevant to that kind of transaction.

- The class Status is abstract. The send() method of NetworkToBank constructs a concrete instance that contains the information appropriate to the response received from the bank to a particular message.

ATM
- id: int - place: String - bankName: String - bankAddress: InetAddress - cardReader: CardReader - cashDispenser: CashDispenser - customerConsole: CustomerConsole - envelopeAcceptor: EnvelopeAcceptor - log: Log - networkToBank: NetworkToBank - operatorPanel: OperatorPanel - receiptPrinter: ReceiptPrinter

- state: int
- switchOn: boolean
- cardInserted: boolean
- OFF_STATE: final int
- IDLE_STATE: final int
- SERVING_CUSTOMER_STATE: final int

+ ATM(id: int, place: String, bankName: String, bankAddress: InetAddress)
+ run()
+ switchOn()
+ switchOff
+ cardInserted()
+ getID(): int
+ getPlace(): String
+ getBankName(): String
+ getCardReader(): CardReader
+ getCashDispenser(): CashDispenser
+ getCustomerConsole(): CustomerConsole
+ getEnvelopeAcceptor(): EnvelopeAcceptor
+ getLog(): Log
+ getNetworkToBank(): NetworkToBank
+ getOperatorPanel(): OperatorPanel
+ getReceiptPrinter(): ReceiptPrinter
- performStartup()
- performShutdown()

CardReader
- atm: ATM
+ CardReader(atm: ATM) + readCard(): Card + ejectCard() + retainCard()

CashDispenser
- log: Log - cashOnHand: Money
+ CashDispenser(log: Log) + setInitialCash(initialCash: Money) + checkCashOnHand(amount: Money): boolean + dispenseCash(amount: Money)

CustomerConsole
+ CustomerConsole() + display(message: String) + readPIN(prompt: String): int throws Cancelled + readMenuChoice(prompt: String, menu: String []): int throws Cancelled + readAmount(prompt: String): Money throws Cancelled

EnvelopeAcceptor
- log: Log

+ EnvelopeAcceptor(log: Log) + acceptEnvelope() throws Cancelled

Log
+ Log() + logSend(message: Message) + logResponse(status: Status) + logCashDispensed(amount: Money) + logEnvelopeAccepted()

NetworkToBank
- log: Log - bankAddress: InetAddress
+ NetworkToBank(log: Log, bankAddress: InetAddress) + openConnection() + closeConnection() + sendMessage(message: Message, out balances: Balances): Status

OperatorPanel
- atm: ATM
+ OperatorPanel(atm: ATM) + getInitialCash(): Money

ReceiptPrinter

+ ReceiptPrinter() + printReceipt(receipt: Receipt)

Session
- atm: ATM - pin: int - state: int - READING_CARD_STATE: final int - READING_PIN_STATE: final int - CHOOSING_TRANSACTION_STATE: final int - PERFORMING_TRANSACTION_STATE: final int - EJECTING_CARD_STATE: final int - FINAL_STATE: final int
+ Session(atm: ATM)> + performSession() + setPIN(int pin)

Transaction
atm: ATM # session: Session # card: Card # pin: int # serialNumber: int # message: Message # balances: Balances

- TRANSACTION_TYPES_MENU: final String [] - nextSerialNumber: int - state: int - GETTING_SPECIFICS_STATE: final int - SENDING_TO_BANK_STATE: final int - INVALID_PIN_STATE: final int - COMPLETING_TRANSACTION_STATE: final int - PRINTING_RECEIPT_STATE: final int - ASKING_DO_ANOTHER_STATE: final int
Transaction(atm: ATM, session: Session, card: Card, pin: int) + makeTransaction(atm: ATM, session: Session, card: Card, pin: int): Transaction throws Cancelled + performTransaction(): boolean throws CardRetained + performInvalidPinExtension(): Status throws Cancelled, CardRetained + getSerialNumber(): int # *getSpecificsFromCustomer(): Message throws Cancelled* # *completeTransaction(): Receipt throws Cancelled*

Withdrawal
- from: int - amount: Money
+ Withdrawal(atm: ATM, session: Session, card: Card, pin: int) # getSpecificsFromCustomer(): Message throws Cancelled # completeTransaction(): Receipt

Deposit
- to: int - amount: Money
+ Deposit(atm: ATM, session: Session, card: Card, pin: int) # getSpecificsFromCustomer(): Message throws Cancelled # completeTransaction(): Receipt throws Cancelled

Transfer
- from: int - to: int - amount: Money
+ Transfer(atm: ATM, session: Session, card: Card, pin: int) # getSpecificsFromCustomer(): Message throws Cancelled # completeTransaction(): Receipt

Inquiry
- from: int
+ Inquiry(atm: ATM, session: Session, card: Card, pin: int) # getSpecificsFromCustomer(): Message throws Cancelled # completeTransaction(): Receipt

AccountInformation
+ ACCOUNT_NAMES: final String[] + ACCOUNT_ABBREVIATIONS: final String []

Balances
- total: Money - available: Money
+ Balances() + setBalances(total: Money, available: Money) + getTotal(): Money + getAvailable(): Money

Card
- number: int
+ Card(number: int) + getNumber(): int

Message
+ WITHDRAWAL: final int + INITIATE_DEPOSIT: final int + COMPLETE_DEPOSIT: final int + TRANSFER: final int + INQUIRY: final int - messageCode: int - card: Card - pin: int - serialNumber: int - fromAccount: int

- toAccount: int - amount: int
+ Message(messageCode: int, cardNumber: Card, pin: int, serialNumber: int, fromAccount: int, toAccount: int, amount: Money) + toString(): String + setPIN(pin: int) + getMessageCode(): int + getCard(): Card + getPIN(): int + getSerialNumber(): int + getFromAccount(): int + getToAccount(): int + getAmount(): Money

Money
- cents: long
+ Money(dollars: int) + Money(dollars: int, cents: int) + Money(toCopy: Money) + toString(): String + add(amountToAdd: Money) + subtract(amountToSubtract: Money) + lessEqual(compareTo: Money): boolean

Receipt

- headingPortion: String [] # detailsPortion(): String [] - balancesPortion: String []
Receipt(atm: ATM, card: Card, transaction: Transaction, balances: Balances) + getLines(): Enumeration

Status
+ toString(): String + *isSuccess(): boolean* + *isInvalidPIN(): boolean* + *getMessage(): String*

2. Package Diagram for Example ATM System

The package diagram shows how the various classes are grouped into packages. There are two "top-level" classes - ATMMain and ATMApplet - which allow the system to be run (respectively) as an application or as an Applet. (Only one of the two would be instantiated in any particular use of the system.)

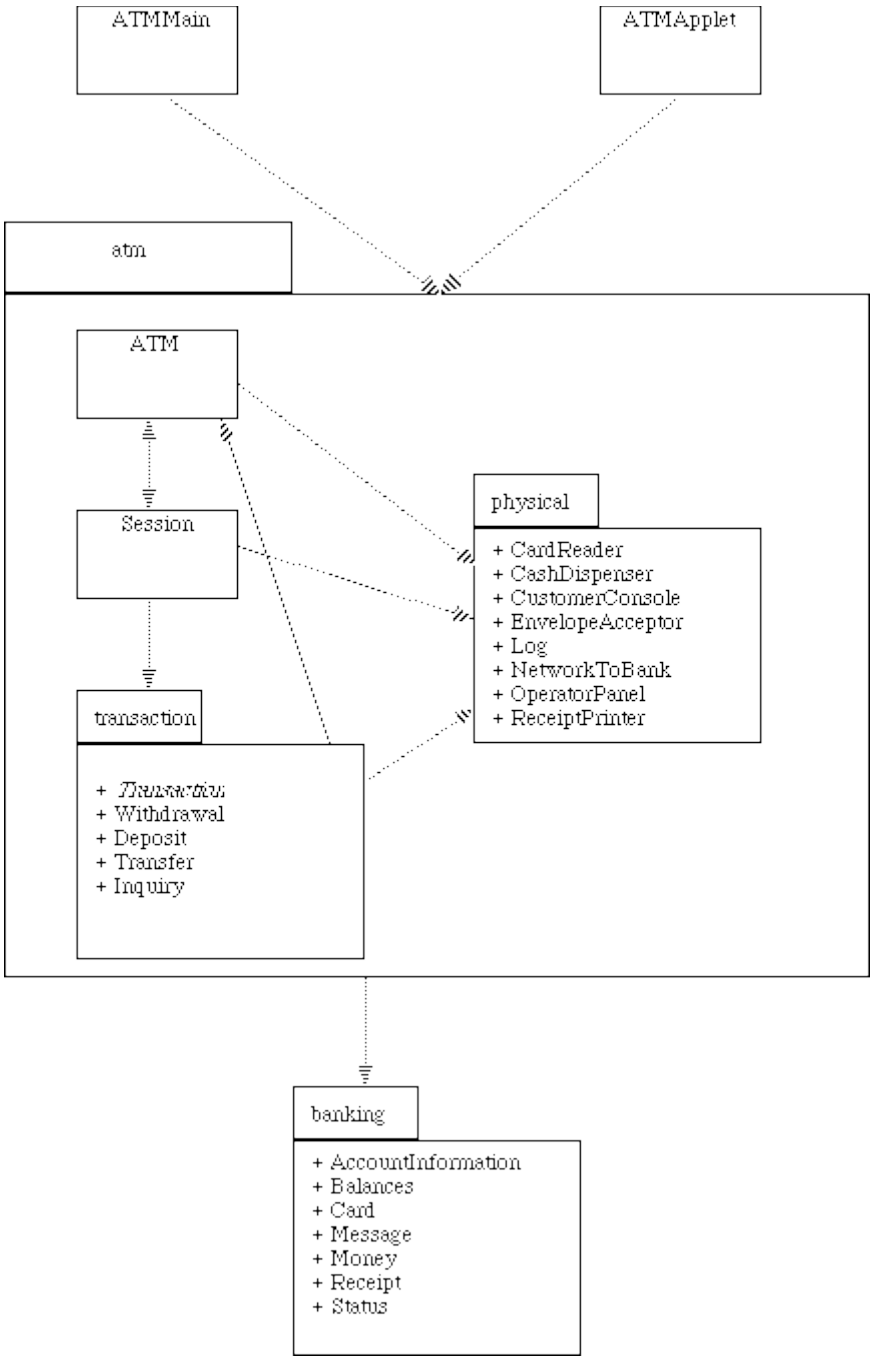
Each of these classes, in turn, depends on the package atm which contains the class ATM that represents the system as a whole, and the class Session that represents one session. ATM depends on Session, and vice versa - since the ATM creates Sessions, and each Session, in turn, uses the ATM to interact with the customer.

The subpackage transaction contains the classes used to represent individual transactions that a customer initiates. The class Session depends on the transaction package because it creates individual transaction objects. These, in turn, again depend on the ATM to interact with the customer.

The subpackage physical contains the classes that represent the various physical components of the ATM. For the purposes of this simulation, these are simulated by a GUI. A real ATM would have quite different classes in this package - classes that actually control its physical components. The class ATM makes use of these components, and Session and the various kinds of transaction gain access to them through ATM to actually perform the needed operations.

Finally, the package banking contains classes that represent the banking enterprise itself and the information communicated back and forth between the ATM and the bank - i.e. classes which might be the same in a totally different implementation of an ATM that interacts with the same bank.

This is, of course, a simulation. However, most of the code that is specific to the simulation resides in the package physical, plus the two top-level classes. Presumably, the other classes and packages might be similar in a real system.


ATMMain
ATMApplet
atm
ATM
Session
transaction
+ Withdrawal
+ Deposit
+ Transfer
+ Inquiry
physical
+ CardReader
+ CashDispenser
+ CustomerConsole
+ EnvelopeAcceptor
+ Log
+ NetworkToBank
+ OperatorPanel
+ ReceiptPrinter
banking
+ AccountInformation
+ Balances
+ Card
+ Message
+ Money
+ Receipt
+ Status