Test Plan

Class	**Money** - AG		
Method(s)	Testing For	in	out
set/getDollars()	Set / Get as expected	4	4
set/getCents()	Set / Get as expected	12	12
getFraction()	Get as expected	0.21	0.21
set/getisPositive()	Set / Get as expected	false	false
add()	adding two Money values give proper sum	4.25 + 3.75	8.00
add()	adding a positive and a negative Money value does subtraction	4.25 + -3.75	0.50
add()	add negative and negative	-4.25 + -3.75	-8.00
sub()	subtract two values	4.25 - 3.75	0.50
sub()	subtract two negative value	-4.25 - -3.75	-0.50
sub()	subtract a negative from a positive	4.25 - -3.75	8.00
sub()	subtract a positive from a negative	-4.25 - 3.75	-8.00
mult()	multiply by double gives correct value	3.33 x 1.116	toString() = 3.71
mult()	multiply by 5 small double gives correct value to 1/10	3.33 x 1.116^5	5.76 getFraction() 0.45
div()	divide by double gives correct value	3.33 / 1.116	2.98
div()	divide by 5 small double gives correct value to 1/10	3.33 / 1.116^5	1.92 getFraction() 0.36

toString()	always put 2 digits for cents	1.10	1.10
toString()	put negative if negative value	-1.00	-1.00
toString()	if value less than 1/10 away from next cent round	2.9991	3.00
isGreater()	a negative is not greater than a positive	-1, 1	false
isGreater()	a positive is greater than a negative	1, -1	true
isGreater()	a larger value is greater than a smaller value	2, 1	true
isGreater()	a larger negative is no greater than a smaller negative	-2, -1	false
absMoney()	a negative is turned into a positive	-2	2
absMoney()	a positive is left positive	2	2
round()	will round value if less than 1/10 cent away	2.991	3.00
Class	**AddAccount** - AG		
constructor: Customer, sessionID, Account	uses Customer to set customerID attribute	Customer, 44, null	addAccount.customer ID = 44
Class	**TellerTransfer** - AG		
TellerTransfer(Customer, sessionID, amount, toID, frID)	as expected	new TellerTransfer	instanceof Message = true
TellerTransfer(Customer, sessionID, amount, toID, frID)	changes lastTransaction in Account	new TellerTransfer .customer .findAccount() .getLastTransaction() !=	true

		original account last transaction	
TellerTransfer(Customer, sessionID, amount, toID, frID)	toAccount balance is increased	new TellerTransfer .customer .findAccount .getBalance() > original toAccount balance	true
TellerTransfer(Customer, sessionID, amount, toID, frID)	frAccount balance is decreased	new TellerTransfer .customer .findAccount .getBalance() < original frAccount balance	true
Class	**Account** - AG		
isPostiveStatus	as expected	true	true
isPostiveStatus	as expected	false	false
get/Set LastTransaction	as expected	lastTransaction	same
get/Set balance	as expected	0.01	0.01
get/Set opened	as expected	today	today
has/Set attachedCard	as expected	true	true
get/Set cardID	as expected	113	113
get/Set fees	as expected	set list of 3 fees	same
getID()	as expected	512	512
getType()	as expected	SAVINGS	SAVINGS
addFee()	new fee added	new fee	new fee in details()
addFee()	new fee added	new fee	ArrayList .size() > than before
findFee()	find fee by date	date	fee on same date
findFee()	does not find non	date	null

	existing fee		
removeFee()	remove fee based on date	fee	fee no longer found
removeFee()	remove fee based on date	fee	true
removeFee()	does not remove non existing fee	fee	return false
toString()	id, type, balance	511, SAVINGS, 0.01	511, SAVINGS, 0.01
Class	**Customer** - AG		
get/set name	as expected	Aaron Garten	same
get/set passcode	as expected	“sesame”	same
get numSavings	as expected	1	same
get numChecking	as expected	0	same
get/set opened	as expected	yesterday	same
get/set closed	as expected	today	same
get/set PIN	as expected	4321	4321
getID()	as expected	422	422
get/set Accounts	as expected	set 3 list of Accounts	same
get/set closed	as expected	today	same
addAccount()	adds an account	new Account	ArrayList<Account>.size() larger
findAccount()	finds account with id 244	244	account with id 244
findAccount()	does not find non existing account	336	null
removeAccount()	removes account	244	return true
removeAccount()	does not remove non existing account	243	return false
Class	**UniqueIDs** - AG		
addID(int)	adds unused int	46	findID(46) true

addID(int)	does not add taken int	46	false
findNewID()	finds untaken id		findID(received) false
removeID()	removes present id	46	true
removeID()	does not remove any id, does not exist	46	false
findID(int)	finds id	46	true
findID(int)	does not find id, does not exist	46	false
Class	**Teller** - HL		
deposit()	Successfully sending deposit message amount to server	Account # + amount	Successfully deposited with server reply
withdraw()	Successfully sending withdraw amount message to server	Account # + amount	Message from server will be sent back to verify withdraw
dismiss()	Disconnecting from customer's account	Send dismiss message to end session	Receive success message if session was ended
balance()	Get customer account's balance	Account #	String - Account #, Bank type, and balance
close_account()	Close a customer account	Account #	Successfully closed message
add_account	Add a new customer account	Account #	Successfully added message
transfer()	Customer transfer money from one account to another	Amount, account #1, account #2	Successfully transferred message
new_customer()	Create a new customer account	Customer ID, Name	Successfully created new account
add_employee()	Create new employee	Name, employeeID, username, loginpw	Successfully created new employee
remove_employee()	Delete a current employee	Name, employeeID	Successfully deleted current employee
online()	Sends message to	Online message	Successfully turned

	turn on server		on message
shutdown()	Supervisor sends shutdown message to turn off server	Shutdown message	Server will be turned off, connections will be shut off
convertMoney()	Changing string numbers into dollars double and pennies	double amount	Converted money into dollars and pennies
save()	Saving files with file handler	Save message	Files will be saved
access()	Employee will access a customer account	Customer id, pw	Success message
Class	**CustomerFileHandler** - DO		
save(User user)	Verify if a customer is being saved into list then file	User user	Updated Customer information
parse()	Read Customer files and store into ArrayList	Read Customer csv file and split lines Into customer sections.	Each line in csv will be split and stored into ArrayList List = [noofcustomers, name etc]
Class	**EmployeeFileHandler** - HL		
parse()	Read employee files and store into ArrayList	Read employee csv file and split lines Name, Username, Password	Each line in csv will be split and stored into ArrayList List = [Name, Username, Password]
save(Employee employee)	Save new employee into list then file	Employee employee	Updated cvs Name, username, password
Class	**ATM - DO**		
login()	Verify if an ATMuser will be able to login with a valid Card ID and pin	Card ID: 567890 Pin: 1234	Successful Login
login()	Verify if an ATM user will NOT be able to	Card ID: 845798 Pin: 1435	Unsuccessful Login

	login with a invalid Card ID and pin		
login()	Verify if ATM user will NOT be able to login with an empty login field.	Card ID: “ “ Pin: “ “	Unsuccessful Login
deposit()	Verify if an ATMUser will be able to deposit more than $0 and less than $5000	Deposit amount: $200	Successful
deposit()	Verify if an ATM User will be able to deposit more than $5000	Deposit amount: $7000	Unsuccessful
deposit()	Verify if an ATM user will be able to deposit less than $0	Deposit amount: $-300	Unsuccessful
deposit()	Verify messages for successful and unsuccessful deposits	Successful Unsuccessful	“Deposit successful” “Deposit not successful”
withdrawal()	Verify if an ATM User will be able to withdraw from an account with sufficient funds	Account balance: $500 Withdrawal amount: $200	Successful. Account balance: $300
withdrawal()	Verify if an ATM User will be able to withdraw from an account with money but not enough to cover entire withdrawal.	Account balance: $500 Withdrawal amount: $600	Successful Account balance: -$100
withdrawal()	Verify if an ATM user will be able to withdraw from an overdrafted account.	Account balance: -$100 Withdrawal amount: $200	Unsuccessful
withdrawal()	Verify messages for successful and unsuccessful withdrawals	Successful Unsuccessful	“Withdrawal successful” “Withdrawal not successful”
transferFunds()	Verify if an ATM User will be able to transfer	Source account balance: $500	Successful. Source account

	from an account with sufficient funds	Target account balance: $500 Transfer amount: $200	balance: $300 Target account balance: $700
transferFunds()	Verify if an ATM User will be able to transfer from an account with money but not enough to cover entire withdrawal.	Source account balance: $100 Target account balance: $500 Transfer amount: $200	Successful. Source account balance: -$100 Target account balance: $700
transferFunds()	Verify if an ATM user will be able to transfer from an overdrafted account.	Source account balance: -$100 Target account balance: $500 Transfer amount: $200	Unsuccessful
transferFunds	Verify if target account ATM User wants to send money to exists.	Target CheckingID: 567890	Account exists
transferFunds	Verify if target account ATM User wants to send money to does not exist.	Target CheckingID: 5432efre	Account does not exist.
viewBalance()	Verify if account balance displayed is correct	Account balance: $400	Account balance shown: $400
dispenseCash()			
logout()	Verify if ATMUser will be allowed to logout after he/she has logged in.	login() then logout()	Successful
logout()	Verify if an ATM user will be allowed to logout when he/she has NOT logged in	logout()	Unsuccessful
Class	**ATMUser - Dolu**		
set/getATMUserID()	Set / Get as expected	setATMUserID(345)	getATMUserID() = 345
set/getCheckingID()	Set / Get as expected	setCheckingID(345)	getCheckingID() = 345

set/getSavingsID()	Set / Get as expected	setSavingsID(345)	getSavingsID() = 345
set/getATMpin()	Set / Get as expected	setATMpin(1100)	getATMpin() = 1100
is/setCheckingPositive()	Set / Get as expected	setCheckingPositive(true)	isCheckingPositive() = true
is/setSavingsPositive()	Set / Get as expected	setSavingsPositive(true)	isSavingsPositive() = true
Class	**Employee - Dolu**		
set/getName()	Set / Get as expected	setName(“Dolu”)	getName() = “Dolu”
set/getEmployeeID()	Set / Get as expected	setEmployeeID(1234)	getName() = 1234
set/getLoginpwd()	Set / Get as expected	setLoginPwd(“doluodumosu”)	getLoginPwd() = “doluodumosu”
set/getType()	Set / Get as expected	setType(EMPLOYEE)	getType() = EMPLOYEE
set/getLoginUsername()	Set / Get as expected	setLoginUsername(“dolu123”)	getLoginUsername() = “dolu123”
Employee(String name, String loginusername, String loginpwd)	Verify if constructors work	Employee(“Dolu”, “dolu123”, “doluodumosu”)	Name = “Dolu” Loginusername = “dolu123” Loginpwd = “doluodumosu”
Employee(String name, int employeeID, String loginpwd)	Set / Get as expected	Employee(“Dolu”, 123, “dolu123”, “doluodumosu”, EmployeeType.EMPLOYEE)	Name = “Dolu” employeeID = 123 Loginusername = “dolu123” Loginpwd = “doluodumosu” EmployeeType = EMPLOYEE