Java class method name being tested: public class BankTellerController			
Method signature: public void openAccount(ActionEvent event)			
Test case#	**Requirement**	**Test description and input data**	**Expected result/output**
1	This test case tests for missing information for an account type of Checkings	• Open >> Checking • Test data: fn id = “John” • Test data: ln id = null • Test data: dob id = null • Test data: amount id = 2500	“Please fill out all fields”
2	This test case tests for missing information for an account type of College Checkings	• Open >> College Checking • Test data: fn id = “John” • Test data: ln id = “Doe” • Test data: dob = “12/15/2000” • Test data: amount id = 2500 • Test data: no location is selected	“Please fill out all fields”
3	This test case tests for opening an account type of Savings when the loyal value is false	• Open >> Savings • Test data: fn id = “John” • Test data: ln id = “Doe” • Test data: dob = “12/15/2004” • Test data: amount id = 2500 • Test data: loyal not clicked	“Account opened”
4	This test case tests for opening an account type of Savings when the loyal value is true	• Open >> Savings • Test data: fn id = “John” • Test data: ln id = “Doe” • Test data: dob = “12/15/2002” • Test data: amount id = 2500 • Test data: loyal id = loyal	“Account opened”
5	This test case tests for opening an account type of Money Market	• Open >> Money Market • Test data: fn id = “John” • Test data: ln id = “Doe” • Test data: dob =	“Account opened”

		“12/15/2003” ● Test data: amount id = 2500	
6	This test case tests for invalid amount information for an account type of Checkings	● Open >> Checking ● Test data: fn id = “John” ● Test data: ln id = “Doe” ● Test data: dob id = ‘9/3/1989” ● Test data: amount id = 0	“Initial deposit cannot be 0 or negative”
7	This test case tests for negative amount information for an account type of Checkings	● Open >> Checking ● Test data: fn id = “John” ● Test data: ln id = “Doe” ● Test data: dob id = “9/2/1990” ● Test data: amount id = -10	“Initial deposit cannot be 0 or negative”
8	This test case tests for missing amount information for an account type of Checkings	● Open >> Checking ● Test data: fn id = “John” ● Test data: ln id = null ● Test data: dob id = null ● Test data: amount id = null	“Please fill out all fields”
9	This test case tests for opening an information for an account type of College Checkings	● Open >> College Checking ● Test data: fn id = “John” ● Test data: ln id = “Doe” ● Test data: dob = “12/15/2002” ● Test data: amount id = 2500 ● Test data: location id = “newark”	“Account opened”
10	This test case tests for opening an information for an account type of College Checkings	● Open >> College Checking ● Test data: fn id = “John” ● Test data: ln id = “Doe” ● Test data: dob = “12/15/2001” ● Test data: amount id = 2500 ● Test data: location id = “nb”	“Account opened”
11	This test case tests for opening an information for	● Open >> Checking ● Test data: fn id = “John”	“Account opened.

	an account type of College Checkings when a checking exists	• Test data: ln id = “Doe” • Test data: dob = “12/15/2000” • Test data: amount id = 2500 • Open >> College Checking • Test data: fn id = “John” • Test data: ln id = “Doe” • Test data: dob = “12/15/2000” • Test data: amount id = 2500 • Test data: location id = “camden”	John Doe 12/15/2000 same account College Checkings is in the database.”
12	This test case tests for invalid information for an account type of Checkings	• Open >> Checking • Test data: fn id = “John” • Test data: ln id = “Doe” • Test data: dob id = “1/1/2023” • Test data: amount id = 2500	“Date of birth Invalid”
13	This test case tests for opening a valid Checkings account	• Open >> Checking • Test data: fn id = “John” • Test data: ln id = “Doe” • Test data: dob id = “5/19/1971” • Test data: amount id = 2500	“Account opened.”
14	This test case tests for opening an account type of Money Market when it is not given a 2500 amount	• Open >> Money Market • Test data: fn id = “John” • Test data: ln id = “Doe” • Test data: dob = “12/15/2003” • Test data: amount id = 250	“Minimum of $2500 to open a Money Market account.”

Java class method name being tested: public class BankTellerController			
Method signature: public void closeAccount()			
Test case#	**Requirement**	**Test description and input data**	**Expected result/output**
1	This test case tests for	• Close >> Checking	“Account

	closing an account that does not exist because the database is empty	• Test data: fn id = “John” • Test data: ln id = “Doe” • Test data: dob id = “2/3/1992” • Test data: amount id = 2500	database is empty!”
2	This test case tests for closing an account that has missing information	• Close >> Checking • Test data: fn id = “John” • Test data: ln id = null • Test data: dob id = “2/3/1992” • Test data: amount id = 2500	“Please fill out all fields”
3	This test case tests for closing an account that has missing information	• Close >> Checking • Test data: fn id = “John” • Test data: ln id = “Doe” • Test data: dob id = “2/3/1992” • Test data: amount id not clicked	“Please fill out all fields”
4	This test case tests for closing an account that has missing information	• Close >> College Checking • Test data: fn id = “John” • Test data: ln id = null • Test data: dob id = “2/3/1972” • Test data: amount id = 2500 • Test data: location id not selected	“Please fill out all fields”
5	This test case tests for closing an account that does not exist and cannot be found	• Close >> Checking • Test data: fn id = “John” • Test data: ln id = “Doe” • Test data: dob id = “2/3/1982” • Test data: amount id = 2500	“Account not found.”
6	This test case tests for closing an account type of College Checkings that already exists	• Close >> College Checking • Test data: fn id = “John” • Test data: ln id = “Doe” • Test data: dob =	“Account closed.”

		"12/15/2001" ● Test data: amount id = 2500 ● Test data: location id = "nb"	
7	This test case tests for closing an account type of College Checkings that has already been closed	● Close >> College Checking ● Test data: fn id = "John" ● Test data: ln id = "Doe" ● Test data: dob = "12/15/2001" ● Test data: amount id = 2500 ● Test data: location id = "nb"	"Account is closed."

Java class method name being tested: public class BankTellerController			
Method signature: public void depositFromAccount(ActionEvent event)			
Test case#	**Requirement**	**Test description and input data**	**Expected result/output**
1	This test case tests for depositing to an account with an invalid date	● Deposit >> Checking ● Test data: fnWithdrawalDeposit id = "John" ● Test data: lnWithdrawalDeposit id = "Doe" ● Test data: dobWithdrawalDeposit id = "2/3/2023" ● Test data: amount id = 2500	"Date of birth Invalid"
2	This test case tests for depositing to an account that does not exist because the database is empty	● Deposit >> Checking ● Test data: fnWithdrawalDeposit id = "John" ● Test data: lnWithdrawalDeposit id = "Doe" ● Test data: dobWithdrawalDeposit id = "5/19/1975"	"Account database is empty"

		• Test data: amount id = 2500	
3	This test case tests for depositing to an account with a zero amount	• Deposit >> Checking • Test data: fnWithdrawalDeposit id = “John” • Test data: lnWithdrawalDeposit id = “Doe” • Test data: dobWithdrawalDeposit id = “5/19/1971” • Test data: amount id = 0	“Deposit - amount cannot be 0 or negative.”
4	This test case tests for depositing to an account with a negative amount	• Deposit >> Checking • Test data: fnWithdrawalDeposit id = “John” • Test data: lnWithdrawalDeposit id = “Doe” • Test data: dobWithdrawalDeposit id = “5/19/1971” • Test data: amount id = -10	“Deposit - amount cannot be 0 or negative.”
5	This test case tests for depositing to an account when the amount is not given	• Deposit >> Checking • Test data: fnWithdrawalDeposit id = “John” • Test data: lnWithdrawalDeposit id = “Doe” • Test data: dobWithdrawalDeposit id = “5/19/1971” • Test data: amount id not selected	“Please fill out all fields”
6	This test case tests for depositing to an account with missing information	• Deposit >> Checking • Test data: fnWithdrawalDeposit id =	“Please fill out all fields”

		"John" ● Test data: lnWithdrawalDeposit id = null ● Test data: dobWithdrawalDeposit id = "5/19/1971" ● Test data: amount id = 2500	
7	This test case tests for depositing to an account that does not exist	● Deposit >> Checking ● Test data: fnWithdrawalDeposit id = "John" ● Test data: lnWithdrawalDeposit id = "Doe" ● Test data: dobWithdrawalDeposit id = "5/19/1975" ● Test data: amount id = 2500	"John Doe 5/19/1975 Checkings is not in the database"
8	This test case tests for depositing into an account that is closed	● Deposit >> College Checking ● Test data: fnWithdrawalDeposit id= "John" ● Test data: lnWithdrawalDeposit id = "Doe" ● Test data: dobWithdrawalDeposit id = "12/15/2001" ● Test data: amount id = 500 ● Test data: location id = "nb"	"Account reopened."
9	This test case tests for depositing to an account with a valid amount	● Deposit >> Checking ● Test data: fnWithdrawalDeposit id = "John" ● Test data: lnWithdrawalDeposit id = "Doe" ● Test data:	"Deposit - balance updated."

		dobWithdrawalDeposit id = “5/19/1971” ● Test data: amount id = 500	

Java class method name being tested: public class BankTellerController			
Method signature: public void withdrawFromAccount()			
Test case#	**Requirement**	**Test description and input data**	**Expected result/output**
1	This test case tests for withdrawing from an account that does not exist because the database is empty	● Withdraw >> Checking ● Test data: fnWithdrawalDeposit id = “John” ● Test data: lnWithdrawalDeposit id = “Doe” ● Test data: dobWithdrawalDeposit id = “5/19/1975” ● Test data: amount id = 2500	“Account database is empty”
2	This test case tests for withdrawing an account with a zero amount	● Withdraw >> Checking ● Test data: fnWithdrawalDeposit id = “John” ● Test data: lnWithdrawalDeposit id = “Doe” ● Test data: dobWithdrawalDeposit id = “5/19/1971” ● Test data: amount id = 0	“Withdraw - amount cannot be 0 or negative.”
3	This test case tests for withdrawing from an account with a negative amount	● Withdraw >> Checking ● Test data: fnWithdrawalDeposit id = “John” ● Test data: lnWithdrawalDeposit id =	“Withdraw - amount cannot be 0 or negative.”

		"Doe" ● Test data: dobWithdrawalDeposit id = "5/19/1971" ● Test data: amount id = -10	
4	This test case tests for withdrawing from an account when the amount is not given	● Withdraw >> Checking ● Test data: fnWithdrawalDeposit id = "John" ● Test data: lnWithdrawalDeposit id = "Doe" ● Test data: dobWithdrawalDeposit id = "5/19/1971" ● Test data: amount id not selected	"Missing information for withdrawing"
5	This test case tests for withdrawing from an account with missing information	● Withdraw >> Checking ● Test data: fnWithdrawalDeposit id = "John" ● Test data: lnWithdrawalDeposit id = null ● Test data: dobWithdrawalDeposit id = "5/19/1971" ● Test data: amount id = 2500	"Please fill out all fields"
6	This test case tests for withdrawing from an account that does not exist	● Withdraw >> Checking ● Test data: fnWithdrawalDeposit id = "John" ● Test data: lnWithdrawalDeposit id = "Doe" ● Test data: dobWithdrawalDeposit id = "5/19/1975" ● Test data: amount id = 2500	"John Doe 5/19/1975 Checkings is not in the database"

7	This test case tests for withdrawing from an account with a valid amount	• Withdraw >> Checking • Test data: fnWithdrawalDeposit id = “John” • Test data: lnWithdrawalDeposit id = “Doe” • Test data: dobWithdrawalDeposit id = “5/19/1971” • Test data: amount id = 500	“Withdraw - balance updated.”
8	This test case tests for withdrawing from an account when the balance - withdrawal amount < 0. In this case, this specific account from previous test cases has a value balance of $2500. 2500-4000 < 0	• Withdraw >> Checking • Test data: fnWithdrawalDeposit id = “John” • Test data: lnWithdrawalDeposit id = “Doe” • Test data: dobWithdrawalDeposit id = “5/19/1971” • Test data: amount id = 4000	“Withdrawal - insufficient funds.”