Testing Documentation (BankAccount Class)

Test Description	**Expected Results**	**Test Code & Results**
Test the getBalance()method	It should return the value that is defined by the user as the current balance and since (private final double CURRENT_BLALANCE = 9.0), that means that it should return 9.0.	> BankAccount b1 = new BankAccount() > b1.getBalance() 9.0 The result of 9.0 matches the expected results.
Test the Interest Rate Get & Set Methods	Initially the interest has it's default value that is set to 2.1 (private final double INTEREST_RATE = 2.1), however the Interest rate set method is set to 3.2 using the setInterest method and the get method should return 3.2. This is because the purpose of the getter method is to return the value that was set by the setter method.	> BankAccount b1 = new BankAccount() > b1.getInterestRate() 2.1 > b1.setInterestRate(3.2) > b1.getInterestRate() 3.2 The result of 3.2 matches the expected results stated above.
Test the Minimum Balance Get & Set Methods	Initially the minimum balance is defined by the user as the (private final int MIN_BLALANCE = 10) that means that it should first return 10 however by using the set method and setting the value to 25, then the get method should return 25. -- Additionally if a double is passed through the set then an error is shown since min balance can only have enough space for an int so an error will be shown while testing.	> BankAccount b1 = new BankAccount() > b1.getMinimumBalance() 10 > b1.setMinimumBalance(25) > b1.getMinimumBalance() 25 The result of 25 matches the expected results. -- > b1.setMinimumBalance(10.80) Static Error: No method in BankAccount with name 'setMinimumBalance' matches this invocation Arguments: (double)

		Candidate signatures: void setMinimumBalance(int) The result of an static error matches the expected results stated.
Test the get and set ATMFee Methods	Initially the ATM fee is set to the user defined constant where (private final double ATM_FEE = 5.0). However, when we set it to 78.9 and utilize the get method we should see an expected result of 78.9 because the value is set to the value inside the parameter.	>BankAccount b1 = new BankAccount() > b1.getATMFee() 5.0 > b.setATMFee(78.9) > b.getATMFee() 78.9 The result of 78.9 is the result that was exceptected from utilizing the get and set ATM fee methods.
Test the get and set OverDraft Fee Methods	Initially the overdraft fee is set to 10.0 which is the value that is user defined and that is changed by the set method when another value is placed in. So, if we set it to 30.4 the overdraft fee should change.	> BankAccount b = new BankAccount() > b.getOverDraftFee() 10.0 > b.setOverDrafFee(30.4) > b.getOverDraftFee() 30.4 The result of 30.4 matches what was expected.
Test the get and set BouncedCheck Fee Methods	Initially the bounced check fee is set to 30.0 by the user and is BOUNCEDCHECK_FEE = 30.0;However, when the set function is used to change that value to another value such as 20.4 it should change.	> BankAccount b = new BankAccount() > b.getBouncedCheckFee() 30.0 > b.setBouncedCheckFee(20.4) > b.getBouncedCheckFee()

		20.4 The result of 20.4 matches what was expected.
Test the get and set Withdraw Fee Methods	Since the withdrawal fee is set to the user defined value of 20.0(WITHDRAW_FEE = 20.0). The set withdrawal fee can be used to place a double value such as 30.2 and change that value.	> b.getWithdrawFee() 20.0 > b.setWithdrawFee(27.4) > b.getWithdrawFee() 27.4
Tests the getWithdrawLimit Method	Since the withdrawal limit is set to the user defined value of 4.0(WITHDRAW_LIMIT = 4). The set withdraw limit can be used to change that value to another double such as 5.4.	> b.getWithdrawLimit() 4.0
Test the setWithdrawLimit	A method to set the withdrawal limit of the bank account where the limit should only be set to 0 only if the account allows an unlimited number of withdrawals.However in all other cases, the limit is the maximum number of free withdrawals allowed per month.	> BankAccount b = new BankAccount() > b.getWithdrawLimit() 4 **Set to 0 for unlimited number of withdrawals without any fee** > b.setWithdrawLimit(0) > b.getWithdrawLimit() 0 > b.getBalance() 0.0 > b.deposit(1000.0) > b.getBalance() 1000.0 > b.withdraw(10) true > b.getBalance()

		990.0 > b.withdraw(10) true > b.getBalance() 980.0 > b.withdraw(10) true > b.getBalance() 970.0 > b.withdraw(10) true > b.getBalance() 960.0
Tests the deposit method	The deposit method works by adding to the current balance that is in the bank account.	> b1.getBalance() 0.0 > b1.deposit(100.0) > b1.getBalance() 100.0 > b1.deposit(200.0) > b1.getBalance() 300.0 > b1.deposit(0.0) > b1.getBalance() 300.0 **When a negative number is put in** > b1.deposit(-100.0) > b1.getBalance() 200.0

Test the withdraw method	The purpose of the withdrawal method is to subtract the amount that is placed in the current balance. So if I put 5.0 in the get balance method and the balance is 9 then the output should be 4 after withdrawal is used. Also if the limit is set to 4 then it should allow only 4 withdrawals and if it exceeds that a fee that is set by the user is applied and in this case the fee is $20.0 so that value is subtracted. Since the balance after withdrawing over 4 times is 1 it subtracted 20$ and the value now is -19.0	> b.getBalance() 9.0 > b.withdraw(4.0) true > b.getBalance() 5.0 > b.withdraw(10.0) false > b.getBalance() 5.0 (the value is not changed) > b.getWithdrawLimit() 4.0 > b.withdraw(1.0) true > b.getBalance() 4.0 > b.withdraw(1.0) true > b.getBalance() 3.0 > b.withdraw(1.0) true > b.getBalance() 2.0 > b.withdraw(1.0) true > b.getBalance() -19.0

		> b.getWithdrawFee() 20.0 The value that is expected of -19 is met because the withdrawal fee is 20.0 and that is only applied when it is met.
Test the withdrawDraft method	In the withdrawal draft method, if the current balance is greater or equal to this amount, then the balance is reduced by the amount and the method returns true. So if I enter 50 it is less than the current balance of 100 then it should return true. Additionally, the total number of withdrawals for the month is incremented, and the withdrawal fee is applied only if the total number of withdrawals exceeds the limit.That is why if withdrawals over the limit of 4 times then only will the fee be applied. Otherwise, the balance is reduced by the bounced check fee amount and the method returns false.	> b.getBalance() 0.0 > b.deposit(100.0) > b.getBalance() 100.0 > b.withdrawDraft(50.0) true > b.getBalance() 50.0 > b.withdrawDraft(100.0) false > b.getBalance() 20.0 > b.getBouncedCheckFee() 30.0 > b.withdrawDraft(5.0) true > b.getBalance() 15.0 > b.withdrawDraft(5.0) true > b.getBalance() 10.0

		> b.withdrawDraft(5.0) true > b.getBalance() 5.0 > b.withdrawDraft(1.0) true > b.getBalance() -16.0 > b.getWithdrawFee() 20.0 > b.getWithdrawLimit() 4.0 The expected results match the actual results when tested.
Test the withdrawATM method	If the current balance is greater or equal to the sum of the amount that is inputted and the ATM fee, then the balance is reduced by the value imputed additionally the balance is further reduced by the ATM fee and the method returns true. Also, only if this case is met then the total number of withdrawals for the month is incremented, and the withdrawal fee is applied only if the number of withdrawals which in this case is 4 is exceeded. Otherwise, the method returns false and nothing is removed from the account's balance.	> b.getBalance() 100.0 > b.getATMFee() 5.0 > b.getWithdrawFee() 20.0 > b.getWithdrawLimit() 4.0 > b.withdrawATM(5.0) true > b.getBalance() 90.0 > b.withdrawATM(5.0) true > b.getBalance() 80.0

		> b.withdrawATM(5.0) true > b.getBalance() 70.0 > b.withdrawATM(5.0) true > b.getBalance() 60.0 > b.withdrawATM(5.0) true > b.getBalance() 30.0 > b.withdrawATM(155.0) false The expected results match the actual results when tested. With the conditions of the current balance being greater than or equal to the summing of the amount returned false.
Test the incrementDay & incrementMonth method	The increment day method works in conjunction with the increment month method. By summing the current balance and any interest earned so far this month and multiplying this sum by the interest rate divided by 365. Moreover, it also multiples this sum by the interest rate divided by 365. Inorder to test the increment month we must also in conjunction test the increment day since the increment month adds the interest earned too far to the account balance and monthly resets to 0 while also setting the overdraft flag to false only if that case is met.	> BankAccount b = new BankAccount() > b.deposit(100.0) > b.getBalance() 100.0 > b.getInterestRate() 2.1 > b.getMinimumBalance() 10 > b.getWithdrawFee() 20.0 > b.incrementDay() > b.incrementDay()

		> b.incrementDay() > b.incrementDay() > b.incrementDay() > b.getBalance() 100.0 > b.incrementMonth() > b.getBalance() 100.02877043366766 > b.withdraw(95.0) true > b.getBalance() 5.02877043366766 > b.incrementDay() > b.incrementDay() > b.incrementDay() > b.getBalance() -4.97122956633234 > b.getMinimumBalance() 10 > b.getWithdrawFee() 20.0 > b.getOverDraftFee() 10.0 > b.incrementMonth() > b.getBalance() -4.97122956633234 > b.deposit(50.0) > b.getBalance() 45.02877043366766 > b.incrementDay() > b.incrementDay() > b.incrementDay() > b.getBalance()

		45.02877043366766 > b.withdraw(40.0) true > b.getBalance() 5.02877043366766 > b.withdraw(2.0) true > b.incrementDay() > b.getBalance() -6.97122956633234 > b.incrementMonth() > b.getBalance() -6.963457030019126 The expected results match the actual results when tested.

Test bankAccount default constructor for default value set for fields	Following Default Values are set by user as the constants below
	CURRENT_BLALANCE = 0.0;
	MIN_BLALANCE = 10;
	INTEREST_RATE = 3.2;
	ATM_FEE = 5.0;
	OVERDRAFT_FEE = 10.0;
	BOUNCEDCHECK_FEE = 30.0;
	WITHDRAW_FEE = 20.0;
	WITHDRAW_LIMIT = 4;
	MONTH_WITHDRAW_COUNT = 0;
	INTEREST_EARNED = 0.0;
	
	It is expected that they return the values that they are equal to.
		> BankAccount b = new BankAccount()
		> b.getBalance()
		0.0
		> b.getInterestRate()
		3.2
		> b.getMinimumBalance()
		10.0
		> b.getATMFee()
		5.0
		> b.getOverDraftFee()
		10.0
		> b.getBouncedCheckFee()
		30.0
		> b.getWithdrawFee()
		20.0

		> b.getWithdrawLimit() 4.0 The expected results match the actual results when tested.

Test BankAccount constructor with parameters	Constructor is set with following parameters that are expected to be the output:
	interestRate = 2.1
	minBalance = 100
	overdraftFee = 25.0
	atmFee = 10.0
	bouncedCheckFee = 30.0
		> BankAccount b1 = new BankAccount(2.1, 100, 25.0, 10.0, 30.0)
		> b1.get Interest Rate()
		2.1
		> b1.getMinimumBalance()
		100.0
		> b1.getOverD raftFee()
		25.0
		> b1.getATMFee()
		10.0
		> b1.getBouncedCheckFee()
		30.0
		The expected results match the actual results when tested.

Testing Documentation (CreditCard Account Class)

Test Description	**Expected Results**	**Test Code & Results**
Test the generic Credit Card Constructor	The values that are defined by the user as the final preset values in the code should be	> CreditCardAccount c = new CreditCardAccount()

	displayed here since the constructor takes no input.	> c.getCreditLimit() 1000 > c.getInterestRate() 10.0 > c.getMinMonthlyPayment() 50 > c.getLatePaymentPenalty() 30 > c.getBalance() 0.0 > c.getMonthlyPayment() 100.0 The expected results match the actual results when tested.
Test the Credit Card constructor that takes four inputs	The constructor when set to the 4 input values of credit limit. interest rate, minimum monthly payment. and late payment penalties should return those values.	> CreditCardAccount c = new CreditCardAccount(2000, 15, 50, 30) > c.getCreditLimit() 2000 > c.getInterestRate() 15.0 > c.getMinMonthlyPayment() 50 > c.getLatePaymentPenalty() 30 > c.getBalance() 0.0 > c.getMonthlyPayment() 0.0 The expected results match the actual results when tested.

Test getCreditLimit and setCreditLimit	The getter at first returns the default value set by the user as 1000 however the setter method allows the value to be set to the input method and return that.	> c.getCreditLimit() 1000 > c.setCreditLimit(20) > c.getCreditLimit() 20 The expected results match the actual results when tested.
Test getInterestRate and setInterestRate	The getter at first returns the interest rate set by the user, however the setter method allows the value that is taken by the input of the interest rate to be set as that value.	> c.getInterestRate() 10.0 > c.setInterestRate(35) > c.getInterestRate() 35.0 The expected results match the actual results when tested.
Test getMinMonthlyPayment and setMinMonthlyPayment	The getter at first returns the minimum monthly payment set by the user, however the setter method allows the value that is taken by the input of the minimum monthly payment to be set as that value.	> c.getMinMonthlyPayment() 50 > c.setMinMonthlyPayment(62) > c.getMinMonthlyPayment() 62 The expected results match the actual results when tested.
getLatePaymentPenalty and setLatePaymentPenalty	The getter at first returns theLate Payment Penalty Set by the user, however the setter	> c.getLatePaymentPenalty() 30

	method allows the value that is taken by the input of the Late Payment Penalty to be set as that value.	> c.setLatePaymentPenalty(23) > c.getLatePaymentPenalty() 23 The expected results match the actual results when tested.
Test getBalance method	Since the user defined final constant value is set to 0.0 (private final double CURRENT_BLALANCE = 0.0) It is expected that the get Balance method will return 0.0.	> CreditCardAccount c = new CreditCardAccount() > c.getBalance() 0.0 The expected results match the actual results when tested.
Test getMonthlyPayment method	The getter at first returns the monthly payment set by the user, however the setter method allows the value that is taken by the input of the monthly payment to be set as that value.	> c.getMonthlyPayment() 100.0 The expected results match the actual results when tested.
Test charge method	The charge method is expected to return true and add the amount that was inputted if the sum of the current balance and the amount that was input is less than or equal to the credit limit.However if that case is not met it should return false and do nothing.	> CreditCardAccount c = new CreditCardAccount(1000, 10.0, 100, 30) > c.getBalance() 0.0 > c.charge(100.0) true > c.getBalance() 100.0 > c.getCreditLimit() 1000 > c.charge(500.0) true

		> c.getBalance() 600.0 > c.charge(700.0) false > c.getBalance() 600.0 > c.charge(600.0) false > c.charge(400.0) true > c.getBalance() 1000.0 The expected results match the actual results when tested.
Test payment method	This method works by subtracting the input value from the current balance, and adding the input value to a variable that stores the total amount paid so far this month.	> c.getBalance() 1000.0 > c.payment(500.0) > c.getBalance() 500.0 > c.payment(200.0) > c.getBalance() 300.0 The expected results match the actual results when tested.
Test incrementDay & incrementMonth methods	The increment day method works in conjunction with the increment month method. First the credit card is giving its 4 man parameters. Then it is expected that the increment month method should first check if the paid-in-full flag is true or false. Since only	> c.getBalance() 300.0 > c.incrementDay() > c.incrementDay() > c.getBalance() 300.0

	if it is false it should sum the current balance and the interest charged so far and then multiply this sum by the interest rate divided by 365. Additionally it should also add this value to the interest charged so far.However, if it is false it should have nothing. We can check this through the increment month method since it is only added if the previous month's balance was not paid in full, the credit card charges interest on the current balance after each day. This we must check with conjunction to the previous months whether or not the balance was paid in full.	> c.getInterestRate() 10.0 > c.incrementMonth() > c.getBalance() 300.0 > c.incrementDay() > c.incrementDay() > c.incrementDay() > c.incrementDay() > c.getBalance() 300.0 > c.incrementMonth() > c.getBalance() 300.0 > c.getMinMonthlyPayment() 100 > c.getMonthlyPayment() 0.0 > c.getLatePaymentPenalty() 30 The expected results match the actual results when tested.
Test the increment Month method	The increment month method is expected to first add the interest charged so far to the current balance in the account. Also, if the total payment this month is greater than or equal to the monthly payment, set the paid-in-full flag to true, otherwise set it to false.Evermoth the interest charged so far must be reset to 0. Also if the minimum	3. > CreditCardAccount c = new CreditCardAccount(2000, 18, 100, 30) > c.payment(50.0) > c.getBalance() -50.0 > c.charge(1000.0) true > c.getBalance()

	monthly payment is smaller than the monthly payment and the total payment this month is less than the minimum monthly payment the method should add the late payment penalty to the current balance as well as set total payment the value to 0 when it is called but at the same time setting the monthly payment equal to the current balance.	950.0 > c.incrementMonth() > c.getBalance() 950.0 > c.getLatePaymentPenalty() 30 > c.getMonthlyPayment() 0.0 > c.getMonthlyPayment() 0.0 > c.getBalance() 950.0 > c.incrementMonth() > c.getBalance() 950.0 The expected results match the actual results when tested.

Testing Documentation (Account Class)

Test Description	**Expected Results**	**Test Code & Results**
Test the The Account class constructor and get methods	The account constructor creates a new account with the 4 string values and a date input as well. Thus, when they are called they should return the value produced.	> Date d = new Date(2, 10) > Account a = new Account("Rob", "Baigert", "123 Syracuse Ct", "23123", d) > a.getFirstName() "Rob"

		> a.getLastName() "Baigert" > a.getStreetAddress() "123 Syracuse Ct" > a.getZipCode() "23123" > a.getDate() Month: 10 and Day: 2 The expected results match the actual results when tested.
Test the getFirstName and setFirstName	The getter at first returns the first name stored, however the setter method allows the value that is taken by the input of the first name to be set as that value and changed.	> a.getFirstName() "Rob" > a.setFirstName("Robert") > a.getFirstName() "Robert" The expected results match the actual results when tested.
Test the getLastName and setLastName	The getter at first returns the last name stored, however the setter method allows the value that is taken by the input of the last name to be set as that value and changed.	> a.getLastName() "Baigert" > a.setLastName("Strong") > a.getLastName() "Strong" The expected results match the actual results when tested.
Test getStreetAddress and setStreetAddress	The getter at first returns the street address stored by the constructor, however the setter method allows the value that is taken by the	> a.getStreetAddress() "123 Syracuse Ct" > a.setStreetAddress("One Ave")

	input of the street address to be set as that value and changed.	> a.getStreetAddress() "One Ave"
Test getZipCode and setZipCode	The getter at first returns the street address stored by the constructor, however the setter method allows the value that is taken by the input of the street address to be set as that value and changed.	> a.getZipCode() "23123" > a.setZipCode("06067") > a.getZipCode() "06067"
Test getSavingsAccount and setSavingsAccount	The getter at first returns the SavingsAccount that is set to be null. The method will assign and retrieve the value of this field. The method will not create an account, but will assume that the parameter is a correctly created account. The set will change that value to another allocation.	> a.getSavingsAccount() null > BankAccount s = new BankAccount() > a.setSavingsAccount(s) > a.getSavingsAccount() BankAccount@2abc2fdd > a.getSavingsAccount().getBalance() 0.0
Test getCheckingAccount and setCheckingAccount	The getter at first returns the Checking Account that is set to be null. The method will assign and retrieve the value of this field. The method will not create an account, but will assume that the parameter is a correctly created account. The set will change that value to another allocation when utilizing the set method.	> a.getCheckingAccount() null > BankAccount c = new BankAccount() > a.setCheckingAccount(c) > a.getCheckingAccount() BankAccount@60ea35c
Test getMoneyMarketAccount and setMoneyMarketAccount	The getter at first returns the Money Market account that is set to be null. The method will assign and retrieve the value of this field. The method will not create an account, but will	> a.getMoneyMarketAccount() null > BankAccount m = new BankAccount() > a.setMoneyMarketAccount(m)

	assume that the parameter is a correctly created account. The set will change that value to another allocation when utilizing the set method that is also a Bank Account object.	> a.getMoneyMarketAccount() BankAccount@29ba3b60
Test getCreditCardAccount and setCreditCardAccount	The getter at first returns the Credit Card account that is set to be null. The method will assign and retrieve the value of this field. The method will not create an account, but will assume that the parameter is a correctly created account. The set will change that value to another allocation when utilizing the set method that is also a Credit Card object.	> a.getCreditCardAccount() null > CreditCardAccount c = new CreditCardAccount() > a.setCreditCardAccount(c) > a.getCreditCardAccount() CreditCardAccount@5f08d06a
Test setDate and getDate	This method gets the date that is already pre-set and then through the setter method it can be set to another value.	> a.getDate() Month: 10 and Day: 2 > Date d1 = new Date(1, 11) > a.setDate(d1) > a.getDate() Month: 11 and Day: 1
Test incrementDay	incrementDay method calls the increment day method of the associated date object and various types of Bank Accounts and Credit Card Account	> a.getDate() Month: 10 and Day: 2 > Date d1 = new Date(1, 11) > a.setDate(d1) > a.getDate() Month: 11 and Day: 1 > a.incrementDay() > a.getDate() Month: 11 and Day: 2 > a.incrementDay() > a.getDate() Month: 11 and Day: 3

		> Date d1 = new Date(31, 11) > a.setDate(d1) > a.getDate() Month: 11 and Day: 31 > a.incrementDay() > a.getDate() Month: 12 and Day: 1
Test equals	Equals method returns true when two accounts have same first name, last name, street address and zip code else returns false	> Account a1 = new Account("Rob", "Baigert", "123 Syracuse Ct", "23123", d) > Account a2 = new Account("Rob", "Baigert", "123 Syracuse Ct", "23123", d1) > Account a3 = new Account("Robert", "Baigert", "123 Syracuse Ct", "23123", d) > a1.equals(a2) true > a1.equals(a3) false
Test toString	toString method returns the text representation of the date in the below format *First_Name Last_Name Stree_Address Zip_Code associated Account Types and corresponding balance*	> Date d = new Date(18, 10) > Account a1 = new Account("Rob", "Baigert", "123 Syracuse Ct", "23123", d) > a1.toString() "Rob Baigert 123 Syracuse Ct. " > BankAccount s = new BankAccount() > a1.setSavingsAccount(s) > a1.toString() "Rob Baigert 123 Syracuse Ct. Savings: 0.0" > a1.getSavingsAccount().deposit(100.0) > a1.toString() "Rob Baigert 123 Syracuse Ct. Savings:

		100.0" > BankAccount c = new BankAccount() > a1.setCheckingAccount(c) > a1.toString() "Rob Baigert 123 Syracuse Ct. Savings: 100.0, Checking: 0.0" > BankAccount m = new BankAccount() > a1.setMoneyMarketAccount(m) > a1.toString() "Rob Baigert 123 Syracuse Ct. Savings: 100.0, Checking: 0.0, Money Market: 0.0" > CreditCardAccount cc = new CreditCardAccount() > a1.setCreditCardAccount(cc) > a1.toString() "Rob Baigert 123 Syracuse Ct. Savings: 100.0, Checking: 0.0, Money Market: 0.0, Credit Card: 0.0"

Testing Documentation (Date Class)

Test Description	**Expected Results**	**Test Code & Results**
Test the Date class constructor, getDay and getMonth methods	Constructor takes two values and the get methods shows the day and month set by the constructor	> Date d = new Date(15, 8) > d.getDay() 15 > d.getMonth() 8

Test the incrementDay method	incrementDay method increments the day by one day. If the day is more than 31, it increments the month and sets the day to 1. If the month is more than 12 it sets the month to 1.	> Date d = new Date(30, 8) > d.getDay() 30 > d.getMonth() 8 > d.incrementDay() > d.getDay() 31 > d.getMonth() 8 > d.incrementDay() > d.getDay() 1 > d.getMonth() 9 > Date d = new Date(31, 12) > d.getDay() 31 > d.getMonth() 12 > d.incrementDay() > d.getDay() 1 > d.getMonth() 1
Test the equals method	Equals method returns true when the day and month are the same for two dates else returns	> Date d1 = new Date(20, 8) > Date d2 = new Date(20, 8)

	false.	> Date d3 = new Date(15, 8) > d1.equals(d2) true > d1.equals(d3) false
Test the toString method	toString method returns the text representation of the date in the below format Month: *month_number* Day: *day_of_the_month*	> Date d1 = new Date(20, 8) > Date d2 = new Date(20, 8) > Date d3 = new Date(15, 8) > d1.toString() "Month: 8 and Day: 20" > d2.toString() "Month: 8 and Day: 20" > d3.toString() "Month: 8 and Day: 15"