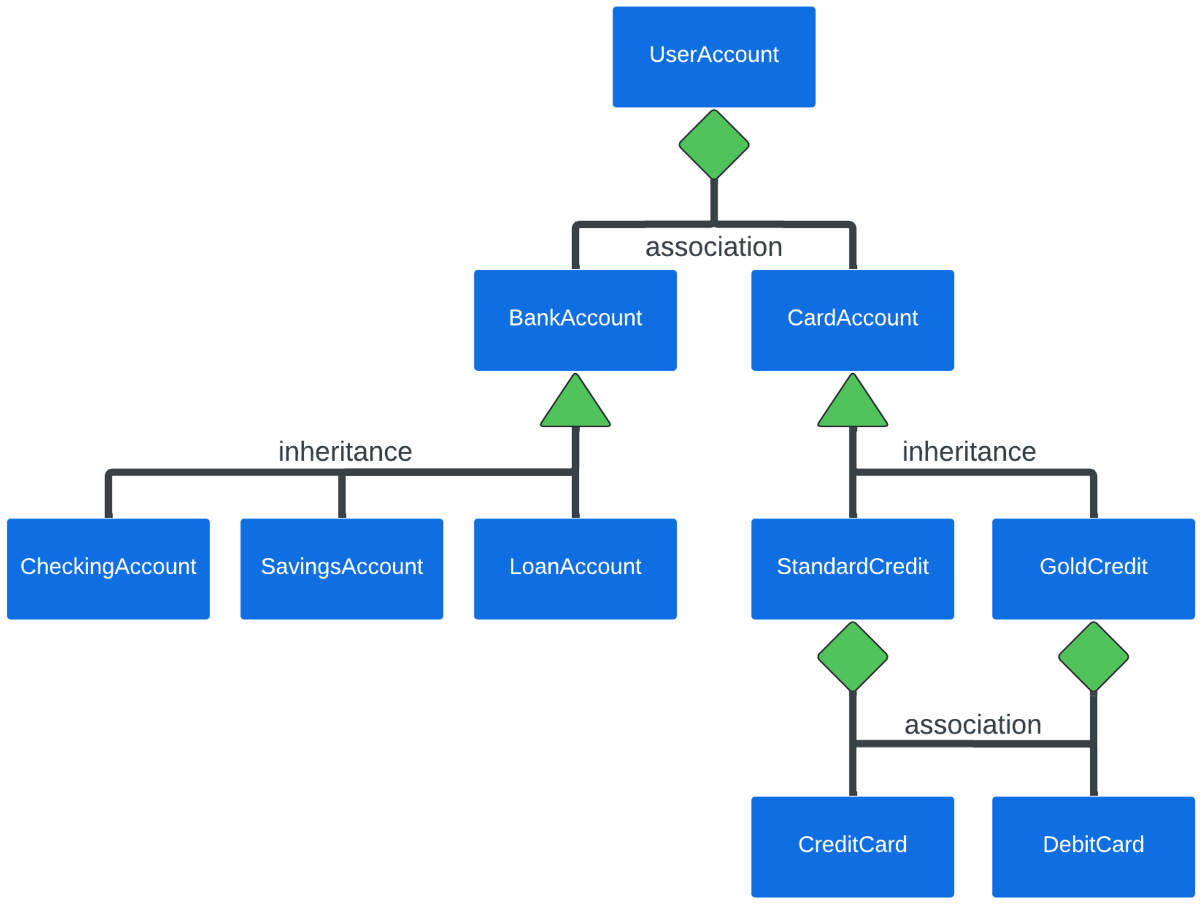
Golden Bank UML Diagram
IT-206-003
Ali Masuth and Sungho Hong

a. All the required classes for the project.
UserAccount, BankAccount, CheckingAccount, SavingsAccount, LoanAccount, CardAccount, StandardCredit, GoldCredit, CreditCard, and DebitCard

b. The complete UML diagram, including the relationship of the classes.

Flowchart:



UML Diagrams:

UserAccount	
// Instance Variables	
(-) name: String	Stores the user’s name.
(-) dateOfBirth: String	Stores the user’s date of birth.
(-) streetAddress: String	Stores the user’s street address.
(-) username: String	Stores the user’s username.
(-) password: String	Stores the user’s password.
(-) <u>numOfUserAccounts</u>: int	Keeps track of the number of user accounts created.
(-) allBankAccounts: BankAccount[]	Stores a collection of all bank accounts created.
(-) allCardAccounts: CardAccount[]	Stores a collection of all card accounts created.
(-) numBankAccounts: int	Keeps track of the total number of bank accounts created.
(-) numCardAccounts: int	Keeps track of the total number of card accounts created.
(+) <u>MAX_ACCOUNTS</u>: int	The maximum number of bank/card accounts allowed to be created (5 accounts).
// Default Constructor	
(+) UserAccount	A regular default constructor
// Defined Specific Constructor	
(+) UserAccount(name: String, dateOfBirth: String, streetAddress: String, username: String, password)	Specific constructor that will initialize the object once all required information has been entered. When the constructor is called, the array of bank and card accounts will also be initialized.

// Accessors	
(+) getName(): String	Retrieves the user’s name.
(+) getDateOfBirth(): String	Retrieves the user’s date of birth.
(+) getStreetAddress(): String	Retrieves the user’s street address.
(+) getUsername(): String	Retrieves the user’s username.
(+) getPassword(): String	Retrieves the user’s password.
(+) getNumOfUserAccounts(): int	Retrieves the current number of user accounts.
(+) getNumBankAccounts(): int	Retrieves the individual’s number of bank accounts.
(+) getNumCardAccounts(): int	Retrieves the individual’s number of card accounts.
(+) getAllBankAccounts(): BankAccount[]	Retrieves all the bank accounts the user had created
(+) getAllCardAccounts(): CardAccount[]	Retrieves all the card accounts the user had created
// Mutators	
(+) setName(name: String): void	Sets the user’s name, with the condition that it cannot be left blank nor can it contain numbers or special characters.
(+) setDateOfBirth(dateOfBirth: String): void	Sets the user’s date of birth. An exception error will be thrown if the date of birth is left blank.
(+) setStreetAddress(streetAddress: String): void	Sets the user’s street address. An exception error will be thrown if the street address is left blank.
(+) setUsername(username: String): void	Sets the user’s unique username. An exception error will be thrown if it is left blank.

(+) setPassword(password: String: void	Sets the user’s password with the condition that it must contain at least 2 digits.
// Special Purpose Methods	
(+) addBankAccount(bankAccount: BankAccount): void	Adds an individual bank account into the array of bank accounts.
(+) addCardAccount(cardAccount: CardAccount): void	Adds an individual card account into the array of card accounts.
(+) deleteBankAccount(searchQuery: String): void	Removes an individual bank account from the array of bank accounts.
(+) deleteCardAccount(searchQuery: String): void	Removes an individual card account from the array of card accounts.
(+) toString(): String	Returns the information of the individual.

BankAccount	**Abstract Superclass and Associated with UserAccount**
// Instance Variables	
(-) accountID: String	The user’s unique bank account ID.
(-) currentBalance: double	The user’s available deposit.
// Default Constructor	
(+) BankAccount()	A regular default constructor
// Defined Specific Constructor	
(+) BankAccount(currentBalance: double)	Specific constructor that will initialize the object once a deposit has been made. A unique bank account ID will also be generated once this constructor is called..

// Accessors (Getter Methods)	
(+) getAccountID(): String	Retrieves the user’s unique account ID.
(+) getCurrentBalancet(): double	Retrieves the user’s current balance.
// Special Purpose Methods	
(+) generateAccountID(): void	Generates and sets a unique bank account ID. Account ID will be a random number between 0-20.
(+) depositIntoAccount(depositAmount: double): void	Deposits the entered value into the bank account. An exception will be thrown if the input is below 0.
(+) withdrawFromAccount(withdrawAmount: double): void	Withdraws the entered value from the bank account. An exception will be thrown if the input is below 0 or if the input exceeds the current balance.
(+) toString(): String	Returns a string containing information regarding the user’s bank account information.

CheckingAccount	**Subclass of BankAccount**
// Default Constructor	
(+) CheckingAccount()	A regular default constructor
// Specific Constructor	
(+) CheckingAccount(currentBalance: double)	Specific constructor that stores the current balance into the BankAccount superclass.
// Special purpose method	
(+) toString(): String	Special purpose method that returns the details of the checking account.

SavingsAccount	**Subclass of BankAccount**
// Instance Variables	
(-) preferredInterestRate: double	Represents the preferred interest rate associated with the account.
(-) minimumBalance: double	Represents the minimum balance required for the account.
(-) numOfWithdrawalsPermitted: int	Represents the number of withdrawals permitted within a specified period, typically a month.
// Default Constructor	
(+) SavingsAccount()	A regular default constructor
// Specific Constructor	
(+) SavingsAccount(currentBalance: double, preferredInterestRate: double, minimumBalance: double, numOfWithdrawalsPermitted: int)	Special constructor that stores the current balance in the BankAccount superclass, with additional variables being validated and stored into this subclass.
// Mutators	
(+) setPreferredInterestRate(preferredInterestRate : double): void	Sets the preferred interest rate for the savings account.
(+) setMinimumBalance(minimumBalance: double): void	Sets the minimum balance required for the account to the specified value.
(+) setNumOfWithdrawalsPermitted(numOfWithdrawalsPermitted: int): void	Sets the number of withdrawals permitted for the savings account.

// Accessors	
(+) getPreferredInterestRate(): double	Returns the preferred interest rate of the savings account.
(+) getMinimumBalance(): double	Returns the minimum balance of the savings account.
(+) getNumOfWithdrawalsPermitted(): int	Returns the number of withdrawals permitted.
// Special Purpose Methods	
(+) decrementNumOfWithdrawalsPermitted(): void	Decrements the number of withdrawals allowed every time a withdrawal is made.
(+) isWithdrawalAllowed(amount: double): boolean	Checks whether a withdrawal of the specified amount is allowed based on the number of account's withdrawals permitted, and the current balance and return Boolean.
(+) toString(): String	Special purpose method that returns the details of the savings account.

LoanAccount	**Subclass of BankAccount**
// Instance Variables	
(-) loanAmount: double	Stores the entered loan amount.
(-) interestRate: double	Stores the desired interest rate.
(-) totalAmount: double	Stores the total amount to pay, which was calculated by the loan amount plus the interest amount.
// Default Constructor	
(+) LoanAccount()	A regular default constructor
// Defined Specific Constructor	

(+) LoanAccount(loanAmount: double, interestRate: double)	Special constructor that automatically sets the deposit amount to 0 since that variable is not needed. This constructor takes in the loan amount and the interest rate. After that, it will call a method that calculates the total amount to pay.
// Mutators	
(+) setLoanAmount(loanAmount: double): void	Mutator that sets the loan amount, requiring that it cannot be less than 0.
(+) setInterestRate(interestRate: double): void	Mutator that sets the interest rate, requiring that it cannot be less than 0.
// Accessors	
(+) getLoanAmount(loanAmount: double): void	Accessor that retrieves the loan amount.
(+) getInterestRate(interestRate: double): void	Accessor that retrieves the interest rate.
// Special Purpose Methods	
(+) calculateTotalAmount(): void	A method where the interest amount is first determined (loanAmount * interestRate) and then the total amount is determined. The total amount is then set by adding the loan amount and the interest amount.
(+) payTotalAmount(paymentAmount: double): void	Method that manages to make the full payment. If the payment amount exceeds the total, an exception will be made.
(+) toString(): String	Returns the details of the loan account.

CardAccount --|> UserAccount **StandardCredit --|> CardAccount** **GoldCredit --|> CardAccount**	

DebitCard	**Associated with CardAccount**
// Instance Variables	
(-) companyName: String	The name of the card's company.
(-) cardNumber: String	Unique number associated with the debit card.
(-) expirationDate: String	Expiration date of the debit card.
(-) cvv: int	CVV (Card Verification Value) of the debit card.
(-) numOfAcct: int	Number of Debit Account number
// Default Constructor	
(+) DebitCard()	
// Defined Specific Constructor	
(+) DebitCard(depositAmount: double, companyName: String)	Requires a company name, as well as deposit amount
// Accessors (Getter Methods)	
(+) getCompanyName(): String	Retrieves the company's name.
(+) getCardNumber(): String	Retrieves the card number.
(+) getExpirationDate(): String	Retrieves the expiration date of the card.
(+) getCVV(): int	Retrieves the CVV of the card.
(+) getNumOfAcct(): int	Retrieves the number of accounts
// Mutators (Setter Methods)	

(+) setCompanyName(companyName: String): void	Sets the company's name with validation.
// Special Purpose Methods	
(+) generateExpirationDate(): void	Generates and sets a random expiration date for the card.
(+) generateCardNumber(): void	Generates and sets a random card number.
(+) generateCVV(): void	Generates and sets a random CCV.
(+) isCardExpired(): boolean	Checks if the card has expired.
(+) isCvvValid(inputCvv: int): boolean	Checks if the provided CVV matches the stored CVV.
(+) isCardValid(): boolean	Check if the card is valid based on the expiration date CVV.
(+) toString(): String	Returns a string containing information regarding the user's debit account information.

CreditCard	**Associated with CardAccount**
// Instance Variables	
(-) companyName: String	The name of the card's company.
(-) cardNumber: String	Unique number associated with the debit card.
(-) expirationDate: String	Expiration date of the debit card.
(-) cvv: int	CVV (Card Verification Value) of the debit card.
(-) numOfAcct: int	Number of Debit Account number
(-) creditLimit: double	This variable stores the maximum amount of credit extended to the cardholder by the issuer.

(-) availableCredit: double	This variable represents the remaining credit that the cardholder can use, considering the current balance and the credit limit.
(-) interestRate: double	This variable holds the annual interest rate applied to the outstanding balance on the credit card.
(-) rewardsPoints: int	This variable tracks the rewards points earned by the cardholder through transactions or promotions.
(-) billingCycle: LocalDate	This variable denotes the date range during which transactions are accumulated for the current billing period.
// Default Constructor	
(+) CreditCard()	
// Defined Specific Constructor	
(+) CreditCard(depositAmount: double, companyName: String, creditLimit: double, interestRate: double)	Requires a company name, creditLimit, interestRate as well as deposit amount
// Accessors (Getter Methods)	
(+) getCompanyName(): String	Retrieves the company's name.
(+) getCardNumber(): String	Retrieves the card number.
(+) getExpirationDate(): String	Retrieves the expiration date of the card.
(+) getCVV(): int	Retrieves the CVV of the card.
(+) getNumOfAcct(): int	Retrieves the number of accounts

(+) getAvailableCredit(): double	This method returns the available credit amount for the cardholder.
(+) getInterestRate(): double	This method returns the annual interest rate applied to the outstanding balance on the credit card.
(+) getRewardsPoints(): int	This method retrieves the total rewards points earned by the cardholder.
(+) getBillingCycle(): LocalDate	This method returns the LocalDate representing the start date of the current billing cycle.
// Special Purpose Methods	
(+) makePayment(amount: double): void	This method allows the cardholder to make a payment towards their outstanding balance.
(+) generateStatement(): void	This method generates a statement summarizing the transactions and balances for the current billing cycle.
(+) generateExpirationDate(): void	Generates and sets a random expiration date for the card.
(+) generateCardNumber(): void	Generates and sets a random card number.
(+) generateCVV(): void	Generates and sets a random CCV.
(+) isCardExpired(): boolean	Checks if the card has expired.
(+) isCvvValid(inputCvv: int): boolean	Checks if the provided CVV matches the stored CVV.
(+) isCardValid(): boolean	Check if the card is valid based on the expiration date CVV.
(+) toString(): String	Returns a string containing information regarding the user’s debit account information.

CardAccount	**Abstract Superclass and Associated with UserAccount**
// Instance Variables	
(-) name: String	The user’s name.
(-) dateOfBirth: String	The user’s date of birth.
(-) streetAddress: String	The user’s street address.
(-) username: String	The user’s unique username.
(-) password: String	The user’s password.
(-) <u>numOfUserAccounts</u>: int	Static variable that keeps track of the number of accounts.
// Defined Specific Constructor	
(+) CardAccount(name: String, dateOfBirth: String, streetAddress: String, username: String, password: String)	Specific constructor that will initialize the object once all required information has been entered.
// Accessors (Getter Methods)	
(+) getName(): String	Retrieves the user’s name.
(+) getDateOfBirth(): String	Retrieves the user’s date of birth.
(+) getStreetAddress(): String	Retrieves the user’s street address.
(+) getUsername(): String	Retrieves the user’s username.
(+) getPassword(): String	Retrieves the user’s password.
(+) <u>getNumOfAccounts()</u>: int	Static method that retrieves the current number of accounts created.
// Mutators (Setter Methods)	

(+) setName(name: String): void	Sets the user's name, with the condition that it cannot be left blank nor can it contain numbers or special characters.
(+) setDateOfBirth(dateOfBirth: String): void	Sets the user's date of birth with the following format: xx-xx-xxxx. The setter will throw an exception error if the above format is not followed.
(+) setStreetAddress(streetAddress: String): void	Sets the user's street address. An exception error will be thrown if the street address is left blank.
(+) setUsername(username: String): void	Sets the user's unique username. An exception error will be thrown if it is left blank.
(+) setPassword(password: String: void	Sets the user's password with the condition that it must contain at least 2 digits.
// Special Purpose Methods	
(+) toString(): String	Returns a string containing information regarding the user's bank account information.

StandardCredit	**Subclass of CardAccount**
// Instance Variables	
(-) balance: double	This variable holds the current balance of the credit card account.
(-) numOfCardAccounts: int	This variable stores the total number of card accounts associated with this credit card account.
(-) MAX_NUM_ACCOUNTS: int	This constant variable defines the maximum number of card accounts that can be associated with this credit card account.

(-) aCreditCards: CreditCard[]	This array stores the credit card objects associated with this credit card account.
(-) aDebitCards: DebitCard[]	This array stores the debit card objects associated with this credit card account.
// Defined Specific Constructor	
(+) StandardCredit(name: String, dateOfBirth: String, streetAddress: String, username: String, password: String, creditCard: CreditCard, debitCard: DebitCard)	This constructor initializes a new credit card account with the provided parameters, including name, date of birth, address, username, password, credit card, and debit card.
// Accessors (Getter Methods)	
(+) getaCreditCards(): CreditCard[]	This method returns an array of credit card objects associated with this credit card account.
(+) getDebitCards(): DebitCard[]	This method returns an array of debit card objects associated with this credit card account.
(+) getnumOfCardAccounts(): int	This method returns the total number of card accounts associated with this credit card account.
// Special Purpose Methods	
(+) processTransaction(): void	This method performs processing for a transaction, such as updating balances and rewards points.
(+) toString(): String	This method returns a string representation of the credit card account, typically including account details and balances.

GoldCredit	**Subclass of CardAccount**
// Instance Variables	
(-) balance: double	This variable holds the current balance of the credit card account.
(-) numOfCardAccounts: int	This variable stores the total number of card accounts associated with this credit card account.
(-) MAX_NUM_ACCOUNTS: int	This constant variable defines the maximum number of card accounts that can be associated with this credit card account.
(-) aCreditCards: CreditCard[]	This array stores the credit card objects associated with this credit card account.
(-) aDebitCards: DebitCard[]	This array stores the debit card objects associated with this credit card account.
// Defined Specific Constructor	
(+) StandardCredit(name: String, dateOfBirth: String, streetAddress: String, username: String, password: String, creditCard: CreditCard, debitCard: DebitCard)	This constructor initializes a new credit card account with the provided parameters, including name, date of birth, address, username, password, credit card, and debit card.
// Accessors (Getter Methods)	
(+) getaCreditCards(): CreditCard[]	This method returns an array of credit card objects associated with this credit card account.
(+) getaDebitCards(): DebitCard[]	This method returns an array of debit card objects associated with this credit card

	account.
(+) getnumOfCardAccounts(): int	This method returns the total number of card accounts associated with this credit card account.
// Special Purpose Methods	
(+) processTransaction(): void	This method performs processing for a transaction, such as updating balances and rewards points.
(+) toString(): String	This method returns a string representation of the credit card account, typically including account details and balances.