OOP

IS612 Introduction To Coding

Spring 2022

Beautiful is better than ugly. Explicit is better than implicit. Simple is better than complex. Complex is better than complicated. Flat is better than nested. Sparse is better than dense. Readability counts. Special cases aren't special enough to

Although practicality beats purity. Errors should never pass silently. Unless explicitly silenced. In the face of ambiguity, refuse the temptation to guess. There should be one – and preferably only one — obvious way to do it. Although that way may not be obvious at first unless you're Dutch. Now is better than never. Although never is often better than right now. If the implementation is hard to explain, it's a bad

may be a good idea.

op s,191 - eapt ne honking great Namespaces are may be a good idea is easy to explain, it dea. If the implementation

now. If the implementation is hard to explain, it's a bad better than never. Although never is often better than right way may not be obvious at first unless you're Dutch. Now is and preferably only one — obvious way to do it. Although that ambiguity, refuse the temptation to guess. There should be one pass silently. Unless explicitly silenced. In the face of Although practicality beats purity. Errors should never break the rules.

op ugnoua jeroads Readability counts, Special cases aren't nested. Sparse is better than dense. than complicated. Flat is better than is better than complex. Complex is bette Explicit is better than implicit. Simple Beautiful is better than ugly.

CEX 1: OOP 1

SIMPLE CLASSES AND OBJECTS

General: Bank Account Model

A bank account is a financial account maintained by a bank for a customer.

BankAccount

accountNumber: String

lastName: String firstName: String taxIDNumber: String checkingBalance: float minCheckingBalance: float maxCheckingWithdrawal: float savingsBalance: float minSavingBalance: float maxSavingWithdrawal: float savingsInterestRate: float static AccountCount = 0: int

init(lastName: String, firstName: String, taxIDNumber: String, checkingBalance: float, savingsBalance: float, savingsInterestRate: float)

displayAccountInfo()...void depositToChecking(double d)...boolean depositToSavings(double d)...boolean withdrawFromChecking(double d)...boolean withdrawFromSavings(double d)...boolean

Constructor Specification

- The Constructor has the signature specified and ALL data fields must be initialized by the constructor.
- Data fields not defined in the constructor parameter list are specified in the following manner:
- accountNumber = 'A-00' + AccountCount (starts at 1)
 minCheckingBalance = 25% of checkingBalance
 maxCheckingWithdrawal = 40% of checkingBalance
 minSavingBalance = 55% of comingBalance

- minSavingBalance = 55% of savingsBalance maxSavingWithdrawal = 20% of savingsBalance
- The constructor displays 'Account [Account Number] HAS BEEN **CREATED** to the Shell when the object is created and increments the AccountCount by 1
- A BankAccount object <u>cannot be fully initialized</u> unless the checkingBalance is greater than \$1000.00 and the savingsBalance is greater than \$500.00...if the object cannot be <u>fully initialized</u> the following message is displayed to the Shell: 'Account [Account Number] IS PENDING ADDITIONAL FUNDS' and all balances and allowances are set to 0.

General: Bank Account Model

A bank account is a financial account maintained by a bank for a customer and has the following behavior.

BankAccount (Method Specifications)

displayAccountInfo()...void

See insert for required data and format

depositToChecking(double d)...boolean

Adds d to the checkingBalance and displays '\$[d] deposited to Checking' ...returns true

depositToSavings(double d)...boolean

Adds d to the savingsBalance and displays 'S[d] deposited to Savings' ...returns true

withdrawFromChecking(double d)...boolean

Subtract d from the checkingBalance after checking to ensure the amount of d is available and that the <u>remaining</u> checkingBalance is greater than the minCheckingBalance and displays '\$[d] withdrawn from Checking' returns true... if either condition is not met then display message 'Withdraw cannot be made' and returns false

withdrawFromSavings(double d)...boolean

Subtract d from the savingsBalance after checking to ensure the amount of d is available and that the <u>remaining</u> savingsBalance is greater than the minSavingBalance and displays '\$[d] withdrawn from Savings' return true... if either condition is not met then display message 'Withdraw cannot be made' and returns false

Account Number: A-002 Name on Account: Richard Thomas Tax ID Number: 654888 BALANCES

> Checking: \$6590.02 Savings: \$5635.0 ____ALLOWANCES___

Minimum Checking Balance: \$1340.0 Maximum Checking Withdrawal: \$2144.0

Minimum Savings Balance: \$2970.0000000000005

Maximum Savings Withdrawal: \$1080.0

Task: Object Creation

Create the following BankAccount objects.

Account Reference (Identifier)	Last Name	First Name	Tax IDNumber	Checking Balance	Savings Balance	Savings Interest Rate
B1	Johnson	Bobby	123456	2390.00	3400.00	2%
B2	Thomas	Richard	654888	5360.00	5400.00	2.5%
В3	Turner	James	622898	360.00	5400.00	2.1%

Task: Validate Model

- Perform the following operations to validate the BankAccount models operation
- Test 1: Deposit \$100.32 to Johnson Checking Account
- ► Test 2: Deposit \$2,000.00 to Johnson Savings Account
- ► Test 3: Withdraw \$90.00 from Johnson Checking Account
- Test 4: Display Johnson Account Information
- Test 5: Deposit \$1,230.02 to Thomas Checking Account
- ► Test 6: Deposit \$1,000.00 to Thomas Savings Account
- Test 7: Withdraw \$765.00 from Thomas Savings Account
- Test 8: Display Thomas Account Information
- Test 9: Display Account Information for Turner Account