BEGIN

// Constants

SET ANNUAL\_INTEREST\_RATE = 4.5

SET OVERDRAFT\_LIMIT = 2000.0

SET TRANSACTION\_FEE = 2.0

// Main Method

FUNCTION main()

CALL Account.setAnnualInterestRate(ANNUAL\_INTEREST\_RATE)

INITIALIZE input AS Scanner

INITIALIZE savingsAccountId, checkingAccountId, checkingBalance, savingsBalance TO 0

INITIALIZE firstName, lastName AS null

SET validInput TO false

// Loop to collect and validate user input

WHILE validInput IS false

TRY

IF savingsAccountId IS 0 THEN

savingsAccountId = CALL validateInt(input, "Enter Savings Account ID:")

ENDIF

IF checkingAccountId IS 0 THEN

checkingAccountId = CALL validateInt(input, "Enter Checking Account ID:")

ENDIF

IF firstName IS null THEN

firstName = CALL validateString(input, "Enter First Name:")

ENDIF

IF lastName IS null THEN

lastName = CALL validateString(input, "Enter Last Name:")

ENDIF

IF checkingBalance IS 0 THEN

checkingBalance = CALL validateDouble(input, "Enter Checking Balance:")

ENDIF

IF savingsBalance IS 0 THEN

savingsBalance = CALL validateDouble(input, "Enter Savings Balance:")

ENDIF

// Create account objects

CREATE checkingAccount AS CheckingAccount(checkingAccountId, firstName, lastName, checkingBalance, OVERDRAFT\_LIMIT)

CREATE savingsAccount AS SavingsAccount(savingsAccountId, firstName, lastName, savingsBalance, TRANSACTION\_FEE)

// Perform operations on accounts

CALL performOperations(input, checkingAccount, savingsAccount)

// Print final account summary

CALL printAccountSummary(checkingAccount, savingsAccount)

SET validInput TO true

CATCH InputMismatchException

PRINT "Invalid input error message"

DISCARD input

CATCH Exception

PRINT "General error message"

ENDWHILE

END FUNCTION

// Function to perform operations

FUNCTION performOperations(input, checkingAccount, savingsAccount)

SET continueOperations TO true

WHILE continueOperations IS true

TRY

DISPLAY operation menu

SET option = CALL validateInt(input, "Choose an option:")

SWITCH (option)

CASE 1:

SET amount = CALL validateDouble(input, "Enter amount to withdraw from Savings:")

CALL savingsAccount.withdraw(amount)

CASE 2:

SET amount = CALL validateDouble(input, "Enter amount to deposit into Savings:")

CALL savingsAccount.deposit(amount)

CASE 3:

SET amount = CALL validateDouble(input, "Enter amount to withdraw from Checking:")

CALL checkingAccount.withdraw(amount)

CASE 4:

SET amount = CALL validateDouble(input, "Enter amount to deposit into Checking:")

CALL checkingAccount.deposit(amount)

CASE 5:

CALL printAccountSummary(checkingAccount, savingsAccount)

CASE 6:

SET continueOperations TO false

PRINT "Exiting application"

DEFAULT:

PRINT "Invalid selection"

ENDSWITCH

CATCH InputMismatchException

PRINT "Invalid input error message"

DISCARD input

CATCH IllegalArgumentException

PRINT "Invalid argument error message"

ENDWHILE

END FUNCTION

// Input validation functions

FUNCTION validateInt(input, prompt)

PRINT prompt

IF input IS NOT valid integer THEN

THROW InputMismatchException("Invalid input. Enter a valid integer.")

ENDIF

SET value TO input

IF value >= 0 THEN

RETURN value

ELSE

THROW InputMismatchException("Invalid input. Enter a positive integer.")

ENDIF

END FUNCTION

FUNCTION validateString(input, prompt)

PRINT prompt

RETURN trimmed input

END FUNCTION

FUNCTION validateDouble(input, prompt)

WHILE true

PRINT prompt

IF input IS NOT valid double THEN

THROW InputMismatchException("Invalid input. Enter a valid double.")

ENDIF

SET amount TO input

IF amount >= 0 THEN

RETURN amount

ELSE

THROW InputMismatchException("Amount cannot be negative.")

ENDIF

ENDWHILE

END FUNCTION

// Function to print account summaries

FUNCTION printAccountSummary(checkingAccount, savingsAccount)

PRINT "Accounts Balance Summary"

PRINT checkingAccount

PRINT savingsAccount

END FUNCTION

END

**Analysis:**

**Problem Statement:**

This assignment is to implement a banking application that manages customer accounts with distinct features for checking and savings accounts. The program must allow users to:

* Create and manage checking and savings accounts for the same customer.
* Perform basic banking operations like deposits and withdrawals.
* Enforce an overdraft limit for checking accounts, allowing customers to withdraw up to a set overdraft limit beyond their balance.
* charge a transaction fee for savings account operations (withdrawals and deposits).
* Display account information including interest rates, balances, and other details when requested.

**Assumptions:**

* The user provides All user inputs required for creating and managing accounts at runtime.
* Each customer will have both a checking and a savings account.
* The bank determines the annual interest rate, transaction fees, and overdraft limits as static constants.
* The program's operation continues until the user decides to exit.

**Inputs:**

1. The user needs to enter an integer for the Savings Account ID.
2. The user needs to enter an integer for the Checking Account ID.
3. The user needs to enter a string for their first name.
4. The user needs to enter a string for their last name.
5. The user needs to enter a real number for the initial Checking account balance.
6. The user needs to enter a real number for the initial Savings account balance.
7. The user needs to enter an integer to select the desired operation (e.g., deposit, withdrawal).
8. The user needs to enter a real number for the amount to deposit or withdraw.

**Validations:**

* **Input Type Validation:** Ensure all user inputs match the expected data types (e.g., integers for IDs, strings for names, real numbers for balances).
* **Positive Value Checks:** Account IDs and transaction amounts must be positive integers or real numbers.
* **Withdrawal Limits:** For checking accounts, allow withdrawals up to the available balance plus the overdraft limit, for savings accounts, ensure withdrawal amounts do not exceed the balance minus the transaction fee.
* **Non-negative Balances:** Ensure balance checks and validations before processing transactions, including zero balances.

**Output:**

1. **User Prompts:** Provide clear prompts for data entry and instructions for selecting operations.
2. **Menu Options:** Offer a menu of operations (withdrawals, deposits, account summaries).
3. **Transaction and Validation Feedback:** Display relevant messages on successful transactions or validation errors (e.g., invalid input, insufficient funds).
4. **Account Summary Display:** Output formatted account details, including:
   * + Account ID
     + Name
     + Balance
     + Annual Interest Rate
     + Monthly Interest Rate
     + Date Created
     + Account Type (Checking/Savings)
     + Overdraft Limit or Transaction Fee

**Sample output:**

Enter Savings Account ID: 1123

Enter Checking Account ID: 1122

Enter First Name: selva

Enter Last Name: kumar

Enter Checking Balance: 30000

Enter Savings Balance: 200000

Select an operation:

1 - Withdraw from Savings

2 - Deposit into Savings

3 - Withdraw from Checking

4 - Deposit into Checking

5 - Print Account Statement

6 - Exit

Choose an option: 1

Enter amount to withdraw from Savings: 50000

Transaction successful. New balance: 149998.0

Select an operation:

1 - Withdraw from Savings

2 - Deposit into Savings

3 - Withdraw from Checking

4 - Deposit into Checking

5 - Print Account Statement

6 - Exit

Choose an option: 3

Enter amount to withdraw from Checking: 15000

Transaction successful. New balance: 15000.0

Select an operation:

1 - Withdraw from Savings

2 - Deposit into Savings

3 - Withdraw from Checking

4 - Deposit into Checking

5 - Print Account Statement

6 - Exit

Choose an option: 6

Exiting application and printing the final account summary.

Accounts Balance Summary:

Account ID: 1122

Name: selva kumar

Balance: $15000.0

Annual Interest Rate: 4.5%

Monthly Interest Rate: 0.375%

Date Created: Sun Nov 17 22:27:48 EST 2024

Account Type: Checking

Overdraft Limit: $2000.0

Account ID: 1123

Name: selva kumar

Balance: $149998.0

Annual Interest Rate: 4.5%

Monthly Interest Rate: 0.375%

Date Created: Sun Nov 17 22:27:48 EST 2024

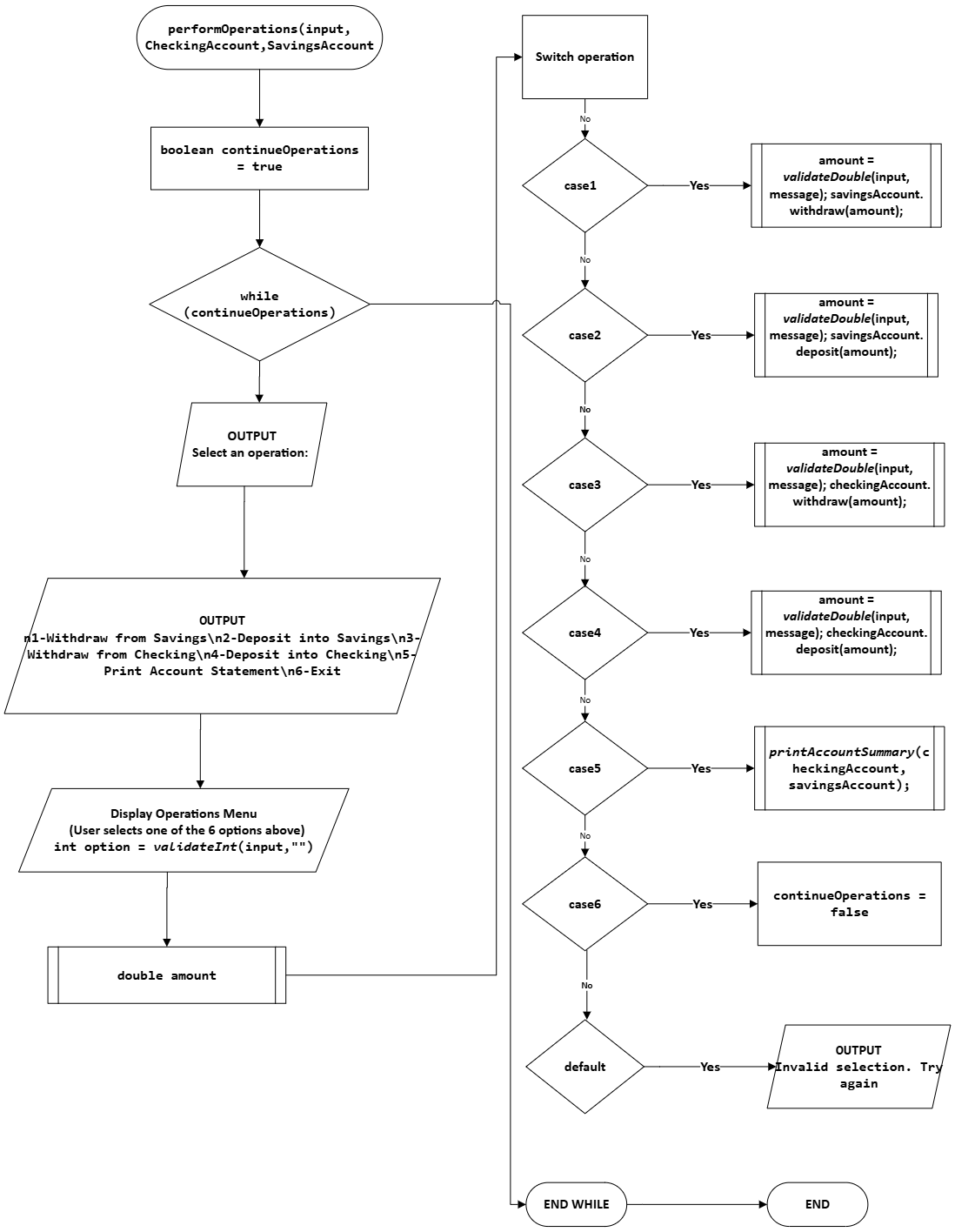
Account Type: Savings

Transaction Fee: $2.0

UML Diagram:

A diagram of a company account

Description automatically generated



A diagram of a function

Description automatically generated with medium confidence A black background with white squares

Description automatically generated A white rectangle with black text

Description automatically generated A white rectangular sign with black text

Description automatically generated

A black background with white squares

Description automatically generated

A black screen with white squares

Description automatically generated