Let's design an exercise that involves creating a simple banking application in Java using ArrayList, HashMap, and custom exceptions. The goal is to create a basic system where users can create accounts, deposit money, withdraw money, and check their balance. Custom exceptions will be used to handle errors like insufficient funds and account not found.

**Banking Application Design**

* **Account Class**: Represents a bank account with attributes like account number, balance, and account holder name.
* **BankingSystem Class**: Manages the accounts using an ArrayList and HashMap.
* **Custom Exceptions**: Define custom exceptions for account not found and insufficient funds.

**Step 1: Create the Account Class**

This class will store the account details such as account number, holder's name, and balance.

public class Account {

private String accountNumber;

private String accountHolderName;

private double balance;

public Account(String accountNumber, String accountHolderName, double initialBalance) {

this.accountNumber = accountNumber;

this.accountHolderName = accountHolderName;

this.balance = initialBalance;

}

public String getAccountNumber() {

return accountNumber;

}

public String getAccountHolderName() {

return accountHolderName;

}

public double getBalance() {

return balance;

}

public void deposit(double amount) {

balance += amount;

}

public void withdraw(double amount) throws InsufficientFundsException {

if (balance < amount) {

throw new InsufficientFundsException("Insufficient funds to withdraw " + amount);

}

balance -= amount;

}

}

**Step 2: Create Custom Exceptions**

We will need two custom exceptions: one for insufficient funds and one for account not found.

**InsufficientFundsException.java**

public class InsufficientFundsException extends Exception {

public InsufficientFundsException(String message) {

super(message);

}

}

**AccountNotFoundException.java**

public class AccountNotFoundException extends Exception {

public AccountNotFoundException(String message) {

super(message);

}

}

**Step 3: Banking System Implementation**

We will create a class that uses an ArrayList and HashMap to store and manage the accounts.

**BankingSystem.java**

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class BankingSystem {

// Use HashMap to store accounts by account number

private Map<String, Account> accounts = new HashMap<>();

// Method to create a new account

public void createAccount(String accountNumber, String accountHolderName, double initialBalance) {

Account account = new Account(accountNumber, accountHolderName, initialBalance);

accounts.put(accountNumber, account);

System.out.println("Account created successfully for " + accountHolderName);

}

// Method to deposit money into an account

public void deposit(String accountNumber, double amount) throws AccountNotFoundException {

Account account = accounts.get(accountNumber);

if (account == null) {

throw new AccountNotFoundException("Account with number " + accountNumber + " not found.");

}

account.deposit(amount);

System.out.println("Deposited " + amount + " into account " + accountNumber);

}

// Method to withdraw money from an account

public void withdraw(String accountNumber, double amount) throws AccountNotFoundException, InsufficientFundsException {

Account account = accounts.get(accountNumber);

if (account == null) {

throw new AccountNotFoundException("Account with number " + accountNumber + " not found.");

}

account.withdraw(amount);

System.out.println("Withdrew " + amount + " from account " + accountNumber);

}

// Method to check the balance of an account

public void checkBalance(String accountNumber) throws AccountNotFoundException {

Account account = accounts.get(accountNumber);

if (account == null) {

throw new AccountNotFoundException("Account with number " + accountNumber + " not found.");

}

System.out.println("Balance for account " + accountNumber + " is " + account.getBalance());

}

// Method to display all accounts

public void displayAllAccounts() {

if (accounts.isEmpty()) {

System.out.println("No accounts in the system.");

return;

}

for (Account account : accounts.values()) {

System.out.println(account.getAccountHolderName() + " (" + account.getAccountNumber() + ") - Balance: " + account.getBalance());

}

}

}

**Step 4: Main Class to Run the Application**

Now, we'll create the Main class to run the banking system. It will interact with the user to create accounts, make deposits, withdrawals, and check balances.

**Main.java**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

BankingSystem bankingSystem = new BankingSystem();

Scanner scanner = new Scanner(System.in);

// Sample accounts creation

bankingSystem.createAccount("123456", "Alice", 5000);

bankingSystem.createAccount("789101", "Bob", 3000);

boolean running = true;

while (running) {

System.out.println("\n--- Banking System Menu ---");

System.out.println("1. Deposit");

System.out.println("2. Withdraw");

System.out.println("3. Check Balance");

System.out.println("4. Display All Accounts");

System.out.println("5. Exit");

System.out.print("Choose an option: ");

int option = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (option) {

case 1:

// Deposit money

System.out.print("Enter account number: ");

String depositAccount = scanner.nextLine();

System.out.print("Enter amount to deposit: ");

double depositAmount = scanner.nextDouble();

try {

bankingSystem.deposit(depositAccount, depositAmount);

} catch (AccountNotFoundException e) {

System.out.println(e.getMessage());

}

break;

case 2:

// Withdraw money

System.out.print("Enter account number: ");

String withdrawAccount = scanner.nextLine();

System.out.print("Enter amount to withdraw: ");

double withdrawAmount = scanner.nextDouble();

try {

bankingSystem.withdraw(withdrawAccount, withdrawAmount);

} catch (AccountNotFoundException | InsufficientFundsException e) {

System.out.println(e.getMessage());

}

break;

case 3:

// Check balance

System.out.print("Enter account number: ");

String checkBalanceAccount = scanner.nextLine();

try {

bankingSystem.checkBalance(checkBalanceAccount);

} catch (AccountNotFoundException e) {

System.out.println(e.getMessage());

}

break;

case 4:

// Display all accounts

bankingSystem.displayAllAccounts();

break;

case 5:

// Exit the program

running = false;

System.out.println("Exiting the banking system.");

break;

default:

System.out.println("Invalid option, please try again.");

}

}

scanner.close();

}

}

**Step 5: Running the Application**

When you run the Main.java file, the program will allow the user to:

1. Create accounts (already set up with Alice and Bob as examples).
2. Deposit money into accounts.
3. Withdraw money, with error handling for insufficient funds.
4. Check account balances.
5. Display all accounts in the system.

**Sample Output:**

--- Banking System Menu ---

1. Deposit

2. Withdraw

3. Check Balance

4. Display All Accounts

5. Exit

Choose an option: 1

Enter account number: 123456

Enter amount to deposit: 2000

Deposited 2000.0 into account 123456

--- Banking System Menu ---

1. Deposit

2. Withdraw

3. Check Balance

4. Display All Accounts

5. Exit

Choose an option: 3

Enter account number: 123456

Balance for account 123456 is 7000.0

**Custom Exceptions in Action:**

* If you attempt to withdraw more than the available balance, the program will throw and catch the InsufficientFundsException.
* If you try to access an account that doesn't exist, the program will throw and catch the AccountNotFoundException.