/\*\*

\* Driver class for the ApexSaver simulation, demonstrating AP CSA Unit 2 concepts.

\* Topics: Selection (2.3, 2.4, 2.5), Iteration (2.8), Algorithms (2.1).

\*/

public class AccountManager {

public static void main(String[] args) {

// Initial parameters

double initialBalance = 500.00;

double initialRate = 0.05;

double monthlyDeposit = 100.00;

// 2.5 Compound Boolean Expression Check

// Ensures the simulation starts with valid, active account parameters.

if (initialBalance <= 0.0 || initialRate <= 0.0) {

System.out.println("🛑 ERROR: Initial balance and annual rate must both be positive. Simulation aborted.");

return;

}

// 1.13 Object Creation and Storage (Instantiation)

SavingsAccount myAccount = new SavingsAccount(initialBalance, initialRate);

System.out.println("==================================================");

System.out.println("🏦 ApexSaver Simulation Started!");

System.out.println("Initial Balance: $" + myAccount.getBalance());

System.out.println("Annual Interest Rate: " + (initialRate \* 100) + "%");

System.out.println("==================================================");

// Withdrawal attempts for each month (used in the selection logic)

double[] withdrawalAttempts = {500.00, 50.00, 1500.00};

// 2.8 For Loop: Simulates 3 months of activity

for (int month = 1; month <= 3; month++) {

System.out.println("\n--- Month " + month + " Activity ---");

// Step 1: Deposit at start of month

myAccount.deposit(monthlyDeposit);

// Step 2: Attempt withdrawal using Selection logic

double attempt = withdrawalAttempts[month - 1]; // Get the attempt for this month

// 2.3 If Statement: Check for sufficient funds before withdrawal

if (attempt <= myAccount.getBalance()) {

// Funds are sufficient, proceed with withdrawal

myAccount.withdraw(attempt);

} else {

// Funds are insufficient

System.out.println("❌ FAILED WITHDRAWAL: $" + attempt +

" attempted. Insufficient Funds! Balance remains: $" + myAccount.getBalance());

}

// Step 3: Calculate and apply monthly interest

double interest = myAccount.calculateMonthlyInterest();

// Reuse the deposit method to add the interest earned

myAccount.deposit(interest);

System.out.println("📈 INTEREST: Monthly interest of $" +

((double) Math.round(interest \* 100) / 100) + " applied.");

// Step 4: Output final status for the month

System.out.println("==================================================");

System.out.println("💵 END OF MONTH " + month + " BALANCE: $" + myAccount.getBalance());

System.out.println("==================================================");

}

System.out.println("\nSimulation complete.");

}

}