[11-08-2025 18:50] Rithika Jss: package Banking_Transaction_System;

```
//Abstract class
abstract class Account {
 protected double balance;

 // Constructor
 public Account(double initialBalance) {
    this.balance = initialBalance;
 }

 // Abstract methods to be implemented by subclasses
 abstract void deposit(double amount);
 abstract void withdraw(double amount);
 abstract double getBalance();

 // Common method (can be inherited)
 public void showBalance() {
    System.out.println("Current Balance: " + balance);
 }
}

//Subclass 1: SavingsAccount
class SavingsAccount extends Account {

 public SavingsAccount(double initialBalance) {
    super(initialBalance);
 }

 @Override
```

```
    void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited " + amount + " to Savings Account.");
    }

    @Override
    void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn " + amount + " from Savings Account.");
        } else {
            System.out.println("Insufficient balance in Savings Account.");
        }
    }

    @Override
    double getBalance() {
        return balance;
    }
}

//Subclass 2: CheckingAccount
class CheckingAccount extends Account {

    public CheckingAccount(double initialBalance) {
        super(initialBalance);
    }

    @Override
    void deposit(double amount) {
        balance += amount;
```

```
    System.out.println("Deposited " + amount + " to Checking Account.");
}

@Override
void withdraw(double amount) {
    if (amount <= balance) {
        balance -= amount;
        System.out.println("Withdrawn " + amount + " from Checking Account.");
    } else {
        System.out.println("Insufficient balance in Checking Account.");
    }
}

@Override
double getBalance() {
    return balance;
}
}
```

[11-08-2025 18:50] Rithika Jss: package Banking_Transaction_System;

```
public class Bank {
private static int totalAccounts=0;
public Bank() {
            totalAccounts++;
}

public static int getTotalAccount() {
           return totalAccounts;
}
}
```

[11-08-2025 18:51] Rithika Jss: package Banking_Transaction_System;

```
//Final part implementation
class Transaction {
 // final variable: fixed fee for every transaction
 private final double transactionFee;

 // Constructor to set the fee
 public Transaction(double fee) {
    this.transactionFee = fee;
 }

 // final method: cannot be overridden in subclasses
 public final void performTransaction(String type, double amount) {
    System.out.println("Transaction Type: " + type);
    System.out.println("Transaction Amount: ₹" + amount);
    System.out.println("Transaction Fee: ₹" + transactionFee);
    System.out.println("Total Deducted: ₹" + (amount + transactionFee));
    System.out.println("Transaction Completed.\n");
 }

 // Getter to access the final variable if needed
 public double getTransactionFee() {
    return transactionFee;
 }
}
[11-08-2025 18:51] Rithika Jss: package Banking_Transaction_System;

public class Main {
        public static void main(String[] args) {
            Bank acc1 = new Bank();
            Bank acc2 = new Bank();
```

```
        Bank acc3 = new Bank();
        Bank acc4 = new Bank();
        System.out.println("Total Bank Accounts: " + Bank.getTotalAccount());
        SavingsAccount sa = new SavingsAccount(1000);
         sa.deposit(500);
         sa.withdraw(200);
         sa.showBalance();

         CheckingAccount ca = new CheckingAccount(2000);
         ca.deposit(1000);
         ca.withdraw(2500); // Should give insufficient balance
         ca.showBalance();
         // Testing the Transaction class

          // All transactions will have ₹20 as fixed fee
           Transaction t1 = new Transaction(20);
    t1.performTransaction("Withdraw", 500);
          t1.performTransaction("Deposit", 1000);



      }
}
```