Week 10 - S10 - Advanced OOP - UML Diagram - Assignment Problem (HW)

**Name:** Ramesh Harisabapathi Chettiar

**Date of Submission:** 22/10/25

**QNO1🡪**

**Draw a Class Diagram for a simple Bank Account System with classes such as Account,**

Customer, and Bank. Show attributes and methods for each class, and represent the

relationships between them.

**Hints:**

● Use association between Bank and Customer.

● Represent aggregation between Customer and Account.

● Include visibility symbols (+, -, #) for attributes and methods.

1. UML Class Diagram (Text Representation)

+--------------------------------+

| Bank |

+--------------------------------+

| - bankName: String |

| - branchCode: String |

+--------------------------------+

| + addCustomer(c: Customer): void |

| + viewCustomers(): void |

+--------------------------------+

1

|

| (association)

|

\*

+--------------------------------+

| Customer |

+--------------------------------+

| - customerName: String |

| - customerId: int |

| - accounts: List<Account> |

+--------------------------------+

| + openAccount(a: Account): void |

| + closeAccount(a: Account): void |

| + getDetails(): void |

+--------------------------------+

◇ (aggregation)

|

| 1..\*

|

+--------------------------------+

| Account |

+--------------------------------+

| - accountNumber: String |

| - balance: double |

+--------------------------------+

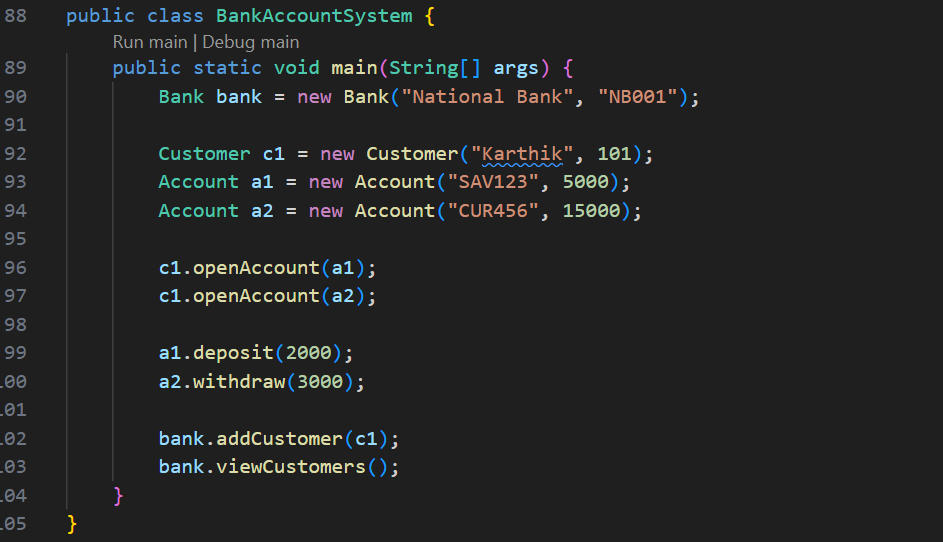
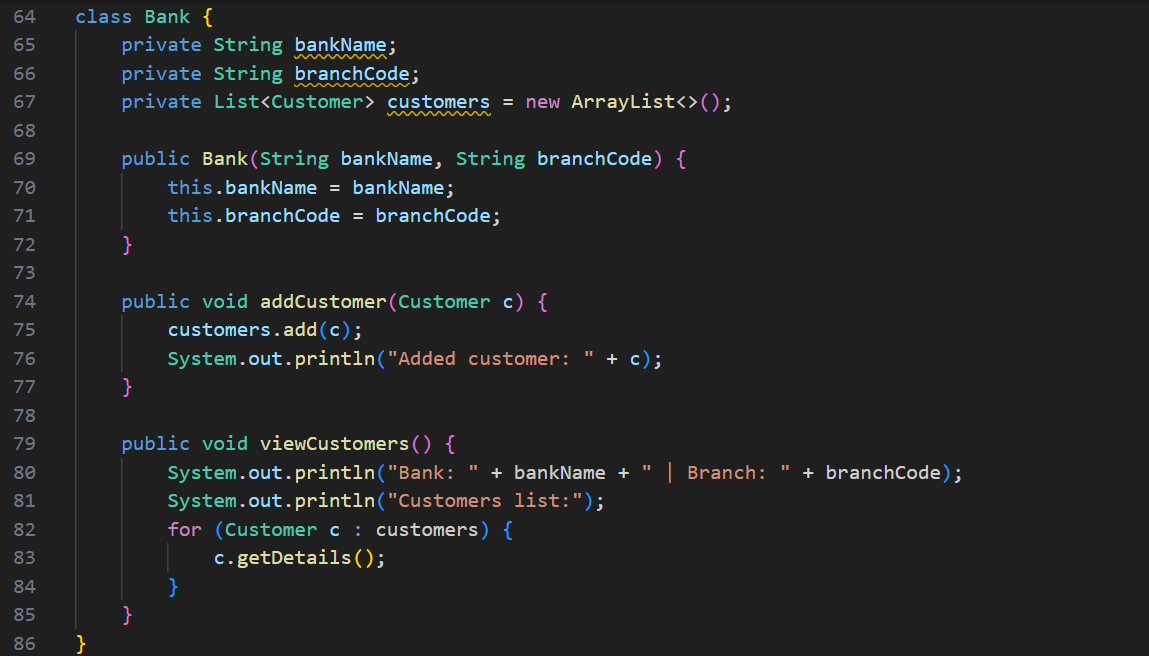
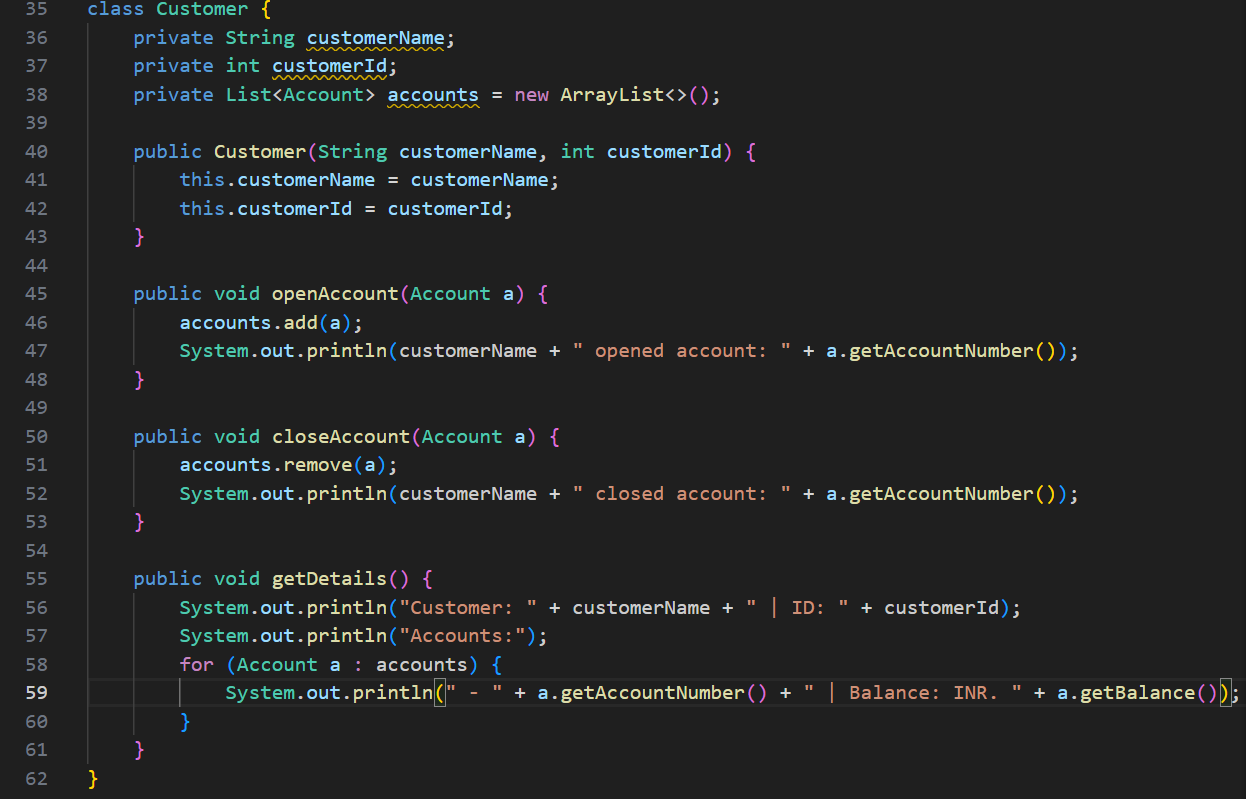
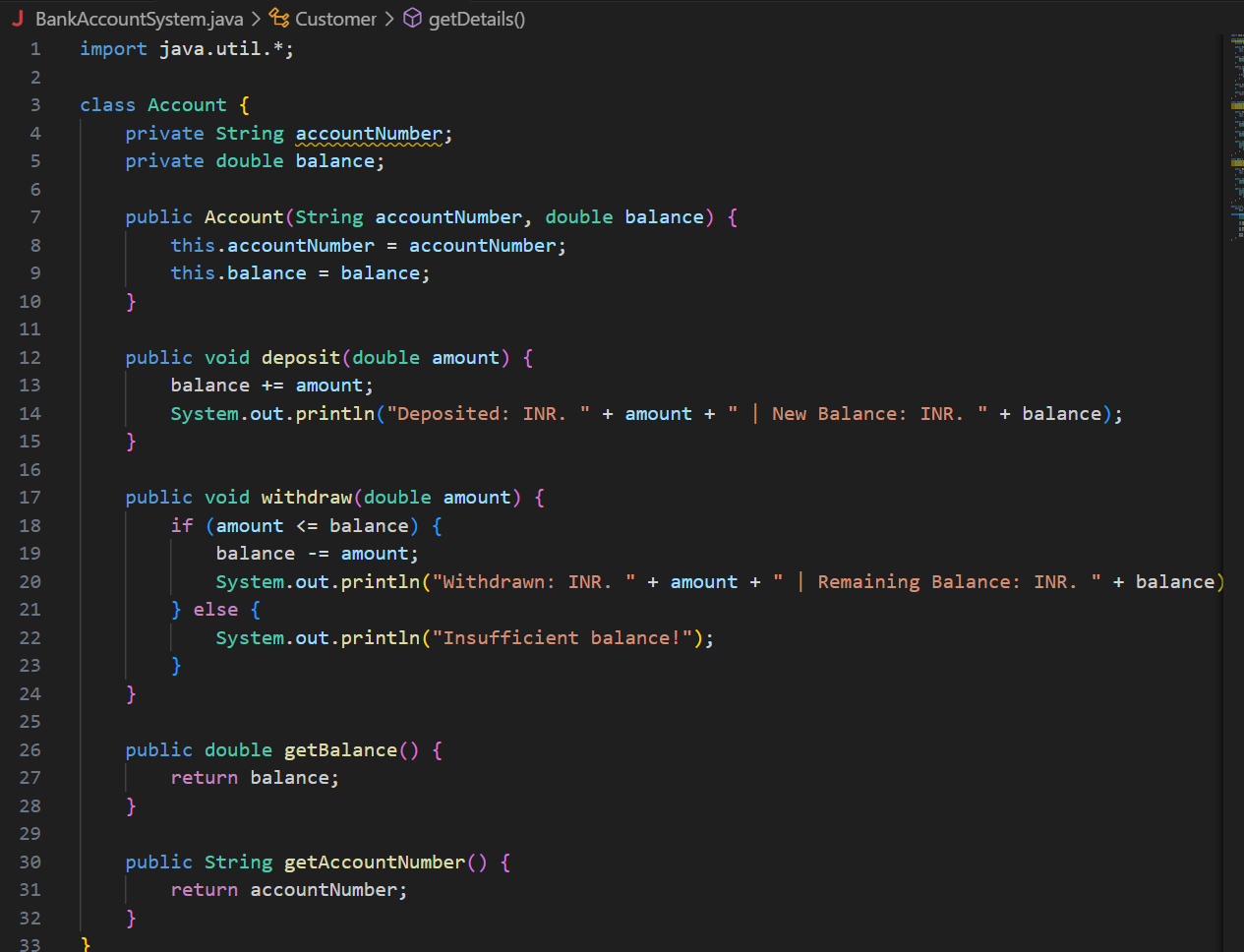
| + deposit(amount: double): void |

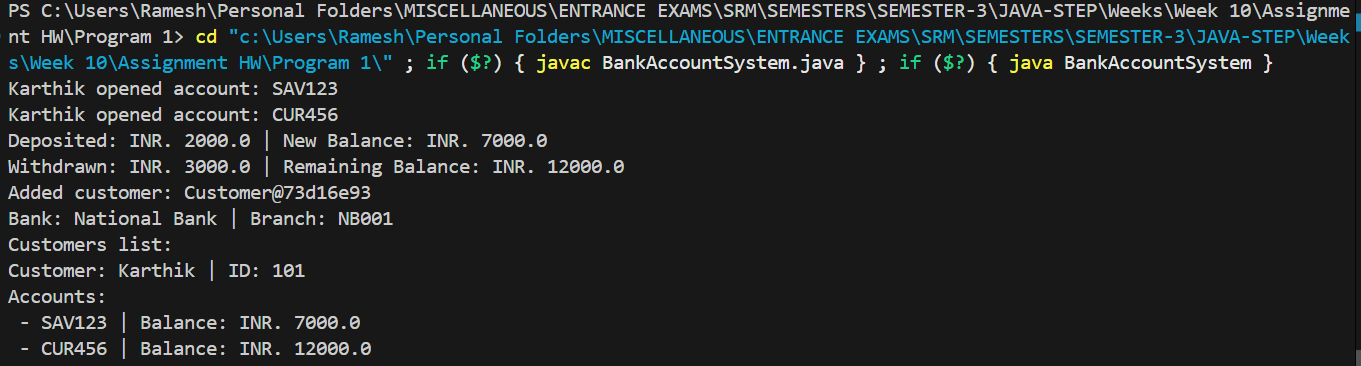
| + withdraw(amount: double): void |

| + getBalance(): double |

+--------------------------------+

BankAccountSystem.java



**OUTPUT🡪**

**QNO2🡪**

**Problem Statement 2:**

Create an Object Diagram showing instances of Customer, Account, and Bank classes.

Display how specific objects are linked during runtime (e.g., one bank has multiple customers,

each with an account).

**Hints:**

● Show object names like cust1:Customer, acc1:Account.

● Include attribute values in objects (e.g., balance = 5000).

● Demonstrate links between instantiated objects.

**Visual Representation (Object Diagram)**

bank1:Bank

---------------

bankName = "Global Bank"

|

| guides

↓

cust1:Customer cust2:Customer

--------------- ---------------

name = "Alice" name = "Bob"

customerId = 101 customerId = 102

| |

↓ ↓

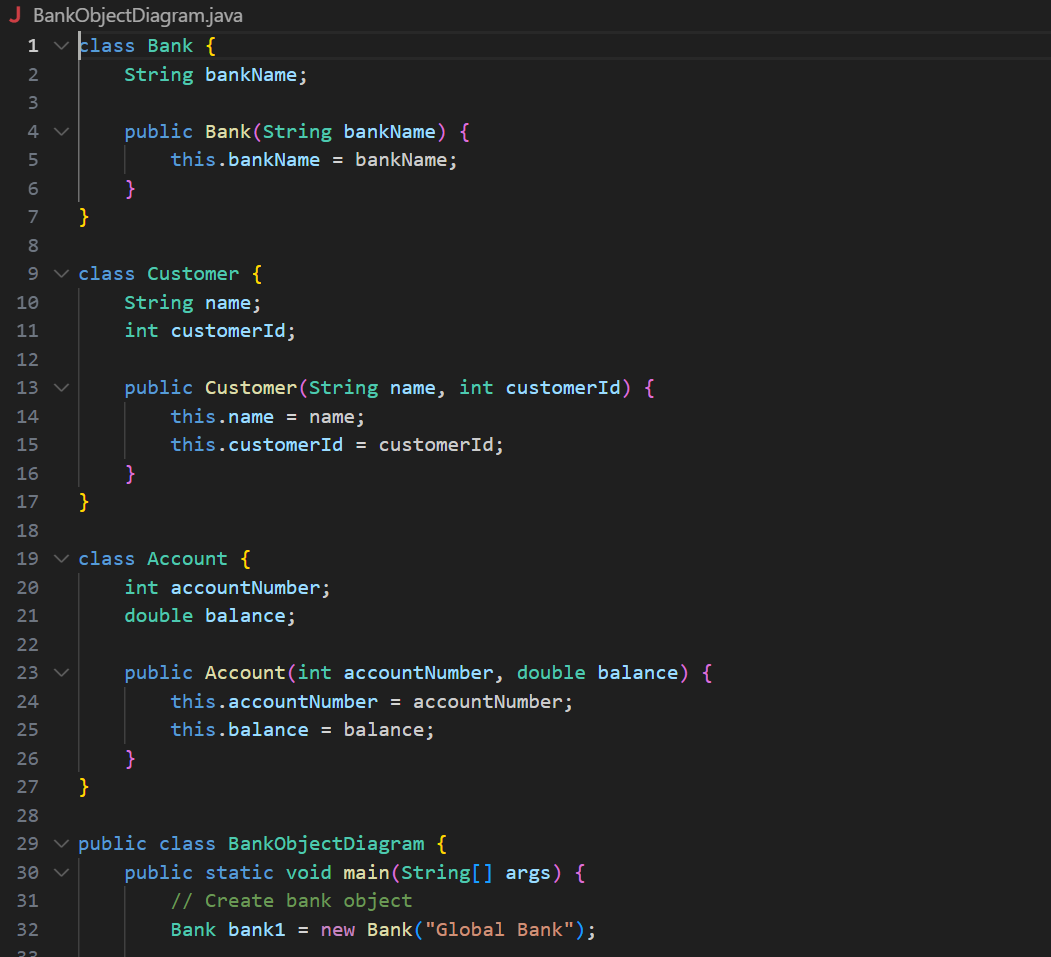
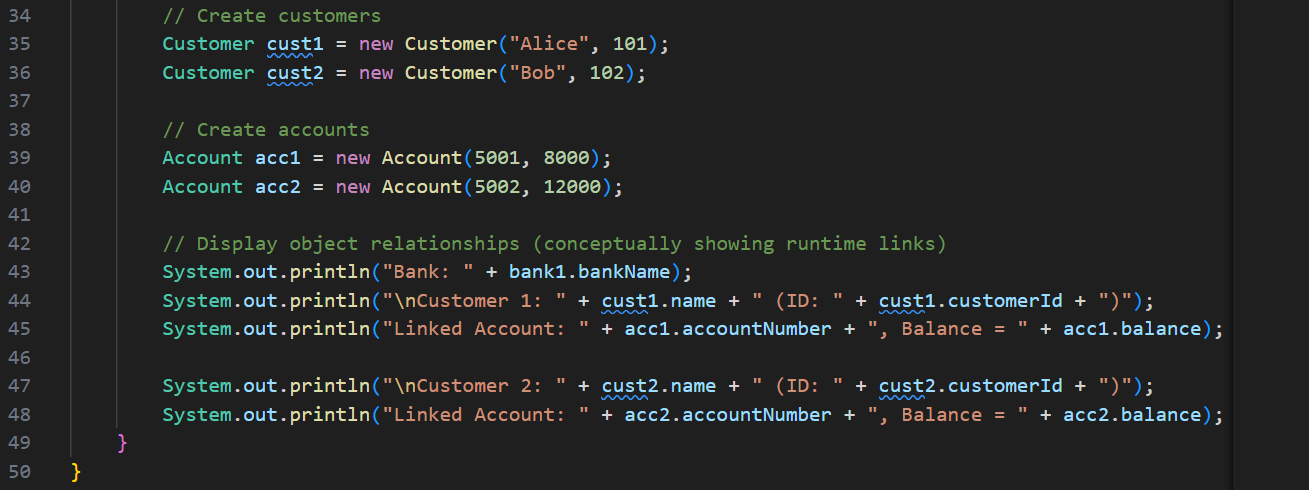
acc1:Account acc2:Account

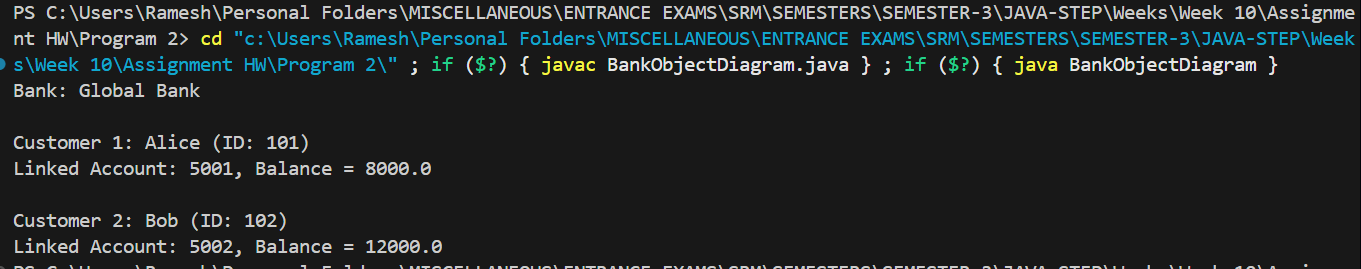
--------------- ---------------

accountNumber = 5001 accountNumber = 5002

balance = 8000 balance = 12000

BankObjectDiagram.java

**OUTPUT🡪** ****

**QNO3🡪**

**Problem Statement 3:**

Draw a Sequence Diagram for an Online Shopping System showing interactions among

Customer, Cart, PaymentService, and OrderService when a customer places an order.

**Hints:**

● Use lifelines for each participant.

● Show method calls (e.g., addItem(), makePayment(), confirmOrder()).

● Include return messages and activation boxes.

Sequence Diagram (Text Representation)

Customer Cart PaymentService OrderService

| | | |

|---addItem()-->| | |

|<--------------| | |

|---makePayment()--------------->| |

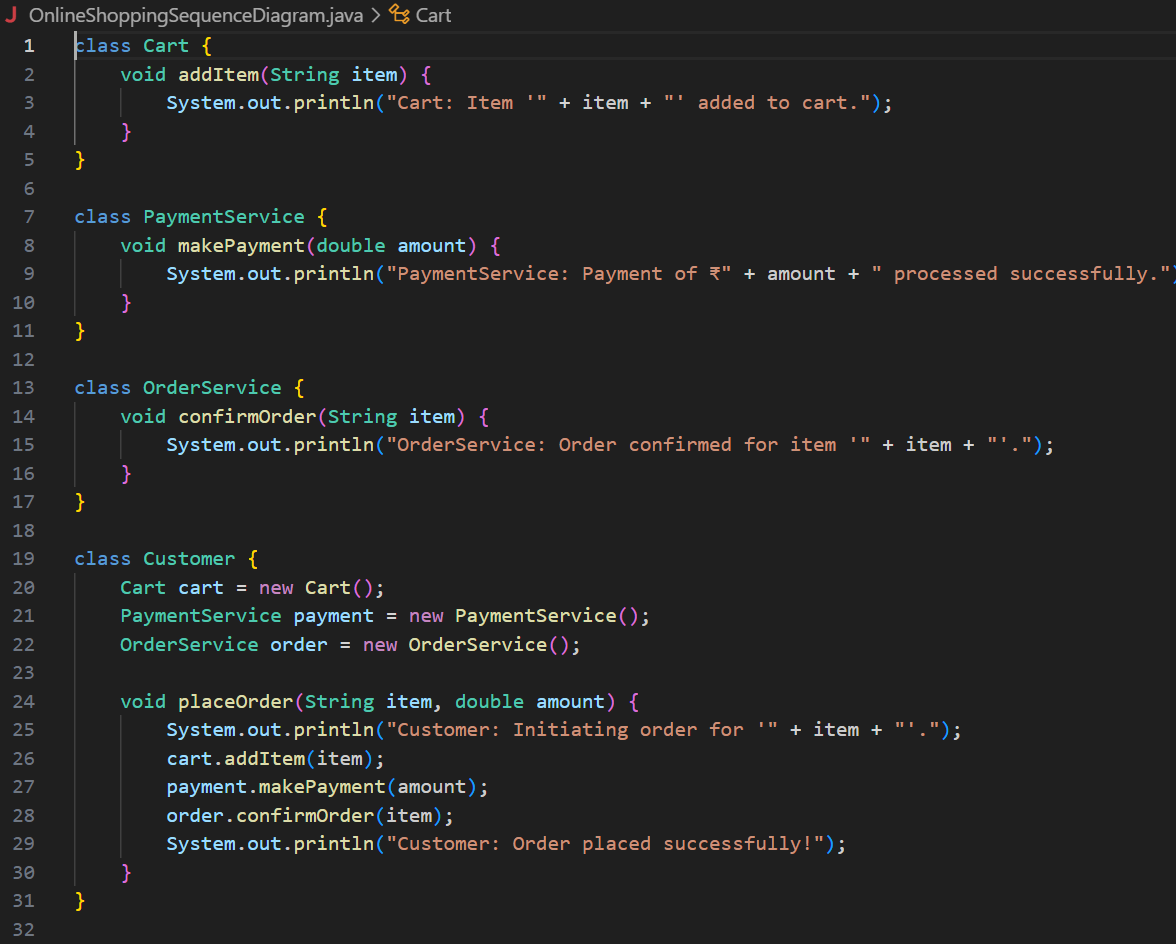
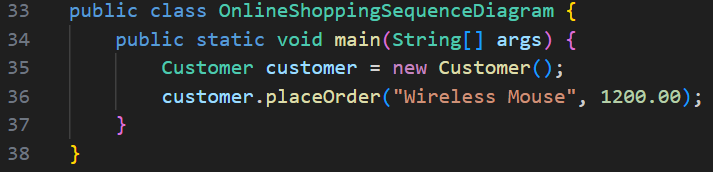
|<-------------------------------| |

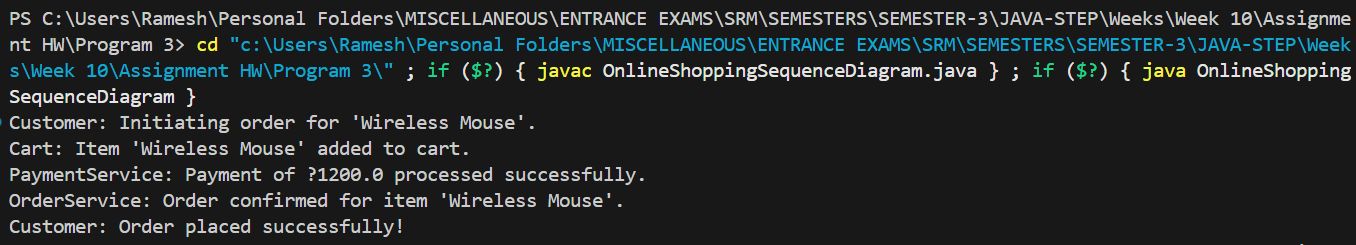
|--------------------------------------confirmOrder()-->|

|<--------------------------------------| |

|---------------(Order Complete)------------------------>|

OnlineShoppingSequenceDiagram.java

**OUTPUT🡪** ****

**QNO4🡪**

**Problem Statement 4:**

Create a Use Case Diagram for a Hospital Management System where an Admin, Doctor,

and Patient interact with features like Manage Appointments, Update Records, and

Generate Bills.

**Hints:**

● Use actor symbols for external users.

● Draw include and extend relationships between use cases.

● Label all relationships clearly.

**Use Case Diagram (Text Representation)**

**+-------------------------------------+**

**| Hospital Management System |**

**+-------------------------------------+**

**Admin ------------------------------------+**

**| |**

**|---> Manage Appointments <---- Patient |**

**| ^ ^ |**

**| | | |**

**| | +--<<extend>> Cancel Appointment**

**| +--<<include>> Book Appointment**

**|**

**+---> Generate Bills <----<<extend>> View Bill (Patient)**

**|**

**+---> Update Records <----- Doctor**

**|**

**Doctor --------------------------------------+**

**| |**

**+---> Manage Appointments --------------+**

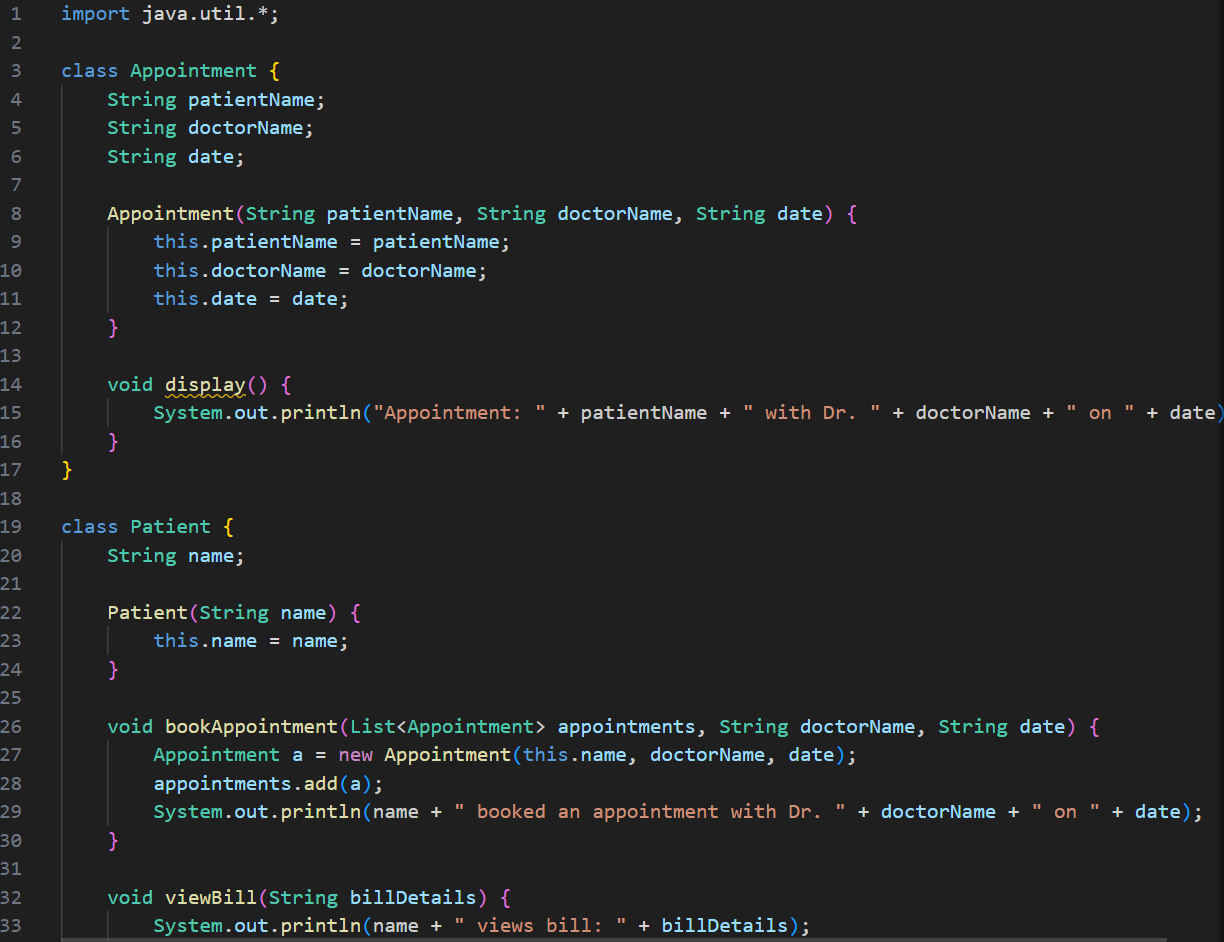
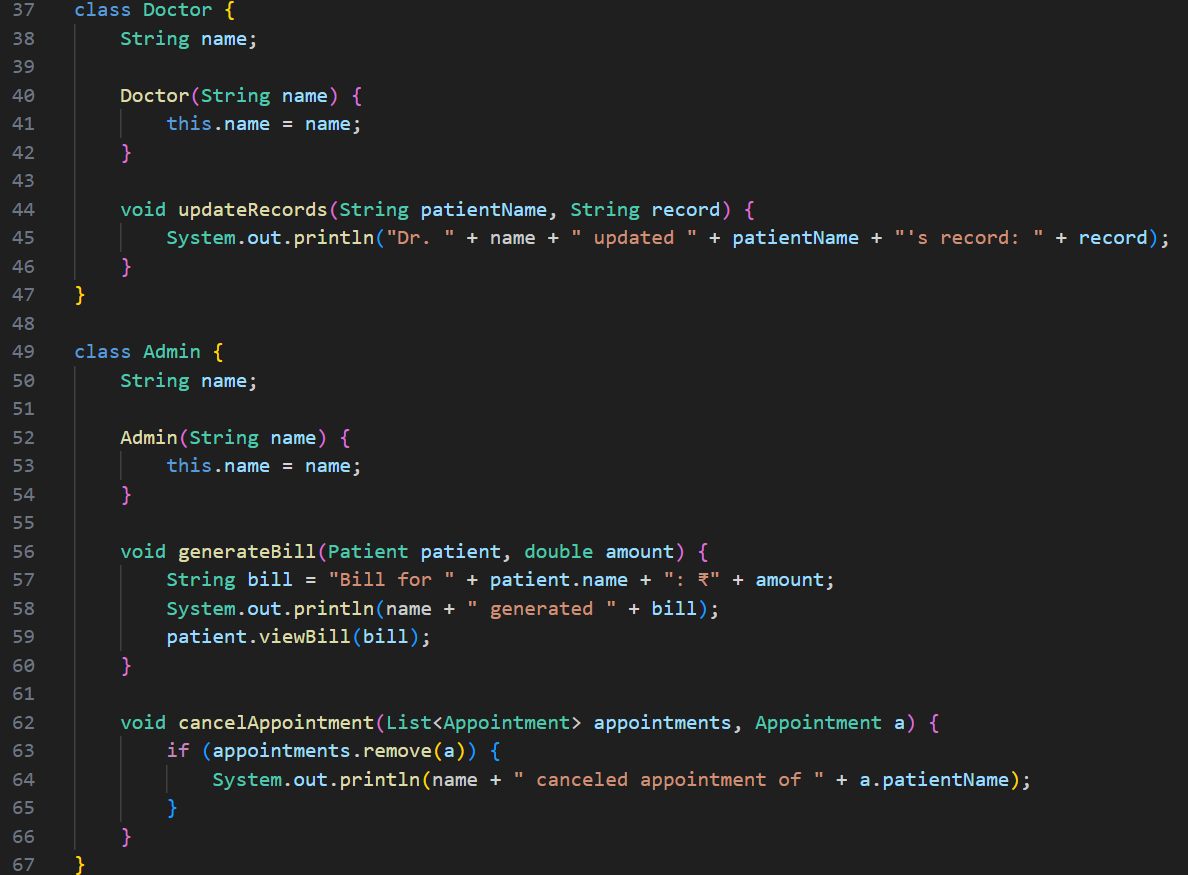
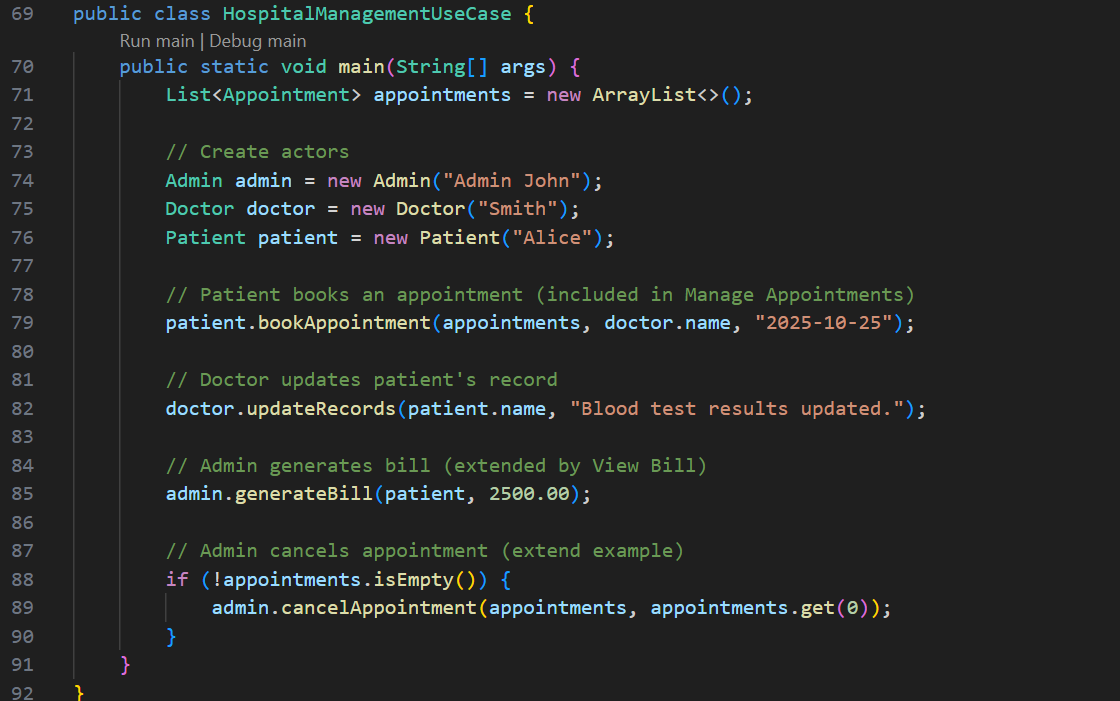
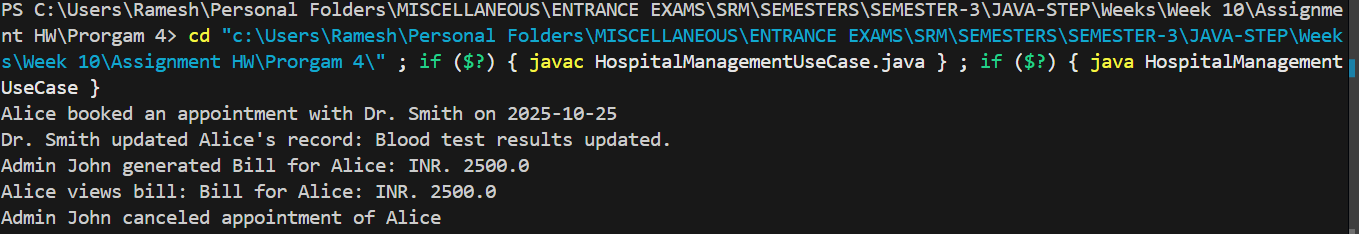
**+---> Update Records**

**|**

**Patient -------------------------------------+**

**+---> Manage Appointments**

**+---> View Bill**

**HospitalManagementUseCase.java** **** **OUTPUT🡪** ****