Hands-on No.	**: 2**
Topic	**: OOPs- Class, Object, Constructor, Encapsulation**
Date	**: 30.07.2025**

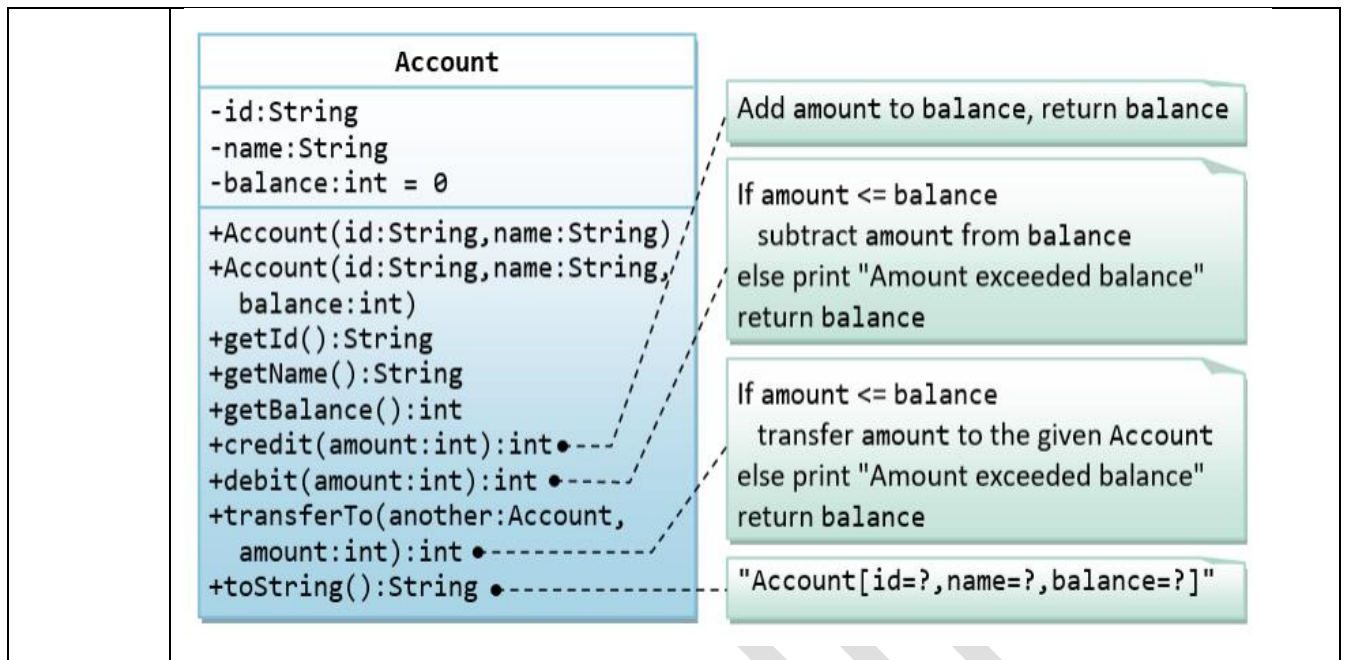**Solve the following problems**

Question No.	Question Detail
1	Design the Rectangle class as given below and test the methods of the class by creating objects of type Rectangle in the driver class or Tester Class. **Rectangle** -length:float = 1.0f -width:float = 1.0f +Rectangle() +Rectangle(length:float,width:float) +getLength():float +setLength(length:float):void +getWidth():float +setWidth(width:float):void +getArea():double +getPerimeter():double +toString():String •----------------- "Rectangle[length=?,width=?]"
2	Design and implement a class named **Employee** to represent an employee working in an organization. Each employee has a unique ID, a first name, a last name, and a monthly salary. Your task is to: 1. **Model the Employee class** using appropriate data members and methods as per the class diagram. 2. **Implement business logic** such as calculating the full name, annual salary, and raising the salary by a given percentage. 3. **Provide a string representation** of the employee object for easy display

It is going to be hard but, hard does not mean impossible.

	Employee -id:int -firstName:String -lastName:String -salary:int +Employee(id:int,firstName:String, lastName:String,salary:int) +getId():int +getFirstName():String +getLastName():String +getName():String ● - - - - - - → *"firstName lastname"* +getSalary():int +setSalary(salary:int):void +getAnnualSalary():int ● - - - - - → salary * 12 +raiseSalary(int percent):int ● - - - → Increase the salary by the percent and return the new salary +toString():String ● - - → "Employee[id=?,name=*firstName Lastname*,salary=?]"
3	Design and implement a class named **Account** to simulate a basic bank account management system. Each account has an account ID, an account holder's name, and a balance. The class should allow performing operations such as credit, debit, and transfer between accounts. Your task is to: 1. **Model the Account class** using appropriate attributes and methods as per the class diagram. 2. **Implement methods** for adding money, withdrawing money, and transferring money to another account with proper balance validation. 3. **Provide a string representation** of the account for easy display.

It is going to be hard but, hard does not mean impossible.

Account

```
-id:String
-name:String
-balance:int = 0

+Account(id:String,name:String)
+Account(id:String,name:String,
   balance:int)
+getId():String
+getName():String
+getBalance():int
+credit(amount:int):int
+debit(amount:int):int
+transferTo(another:Account,
   amount:int):int
+toString():String
```

Add amount to balance, return balance

If amount <= balance
   subtract amount from balance
else print "Amount exceeded balance"
return balance

If amount <= balance
   transfer amount to the given Account
else print "Amount exceeded balance"
return balance

"Account[id=?,name=?,balance=?]"