**Part-2:Object Oriented Programming: Clase ,Methods, Inheritance**

**Git Hub Link:-**

1.Design a class named Circle containing following attributes and behavior.

• One double data field named radius. The default value is 1.

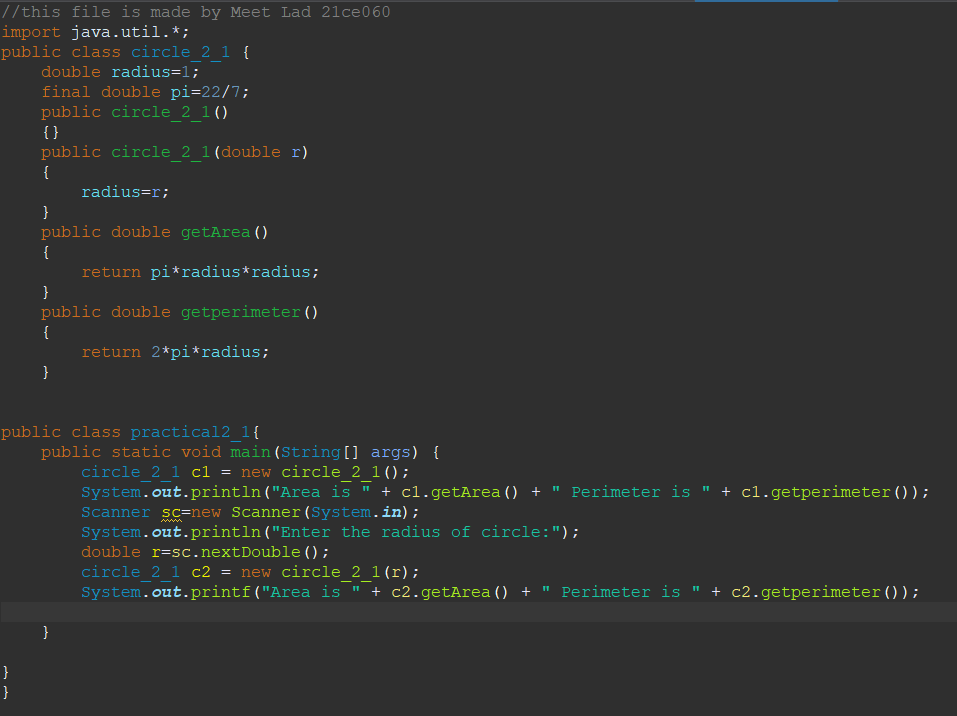
• A no-argument constructor that creates a default circle.

• A Single argument constructor that creates a Circle with the specified radius.

• A method named getArea() that returns area of the Circle.

• A method named getPerimeter() that returns perimeter of it.

Code:



Design a class named Account that contains:

• A private int data field named id for the account (default 0).

• A private double data field named balance for the account (default 500₹).

• A private double data field named annualInterestRate that stores the current interest rate (default 7%). Assume all accounts have the same interest rate.

• A private Date data field named dateCreated that stores the date when the account was created.

• A no-arg constructor that creates a default account.

• A constructor that creates an account with the specified id and initial balance.

• The accessor and mutator methods for id, balance, and annualInterestRate.

• The accessor method for dateCreated.

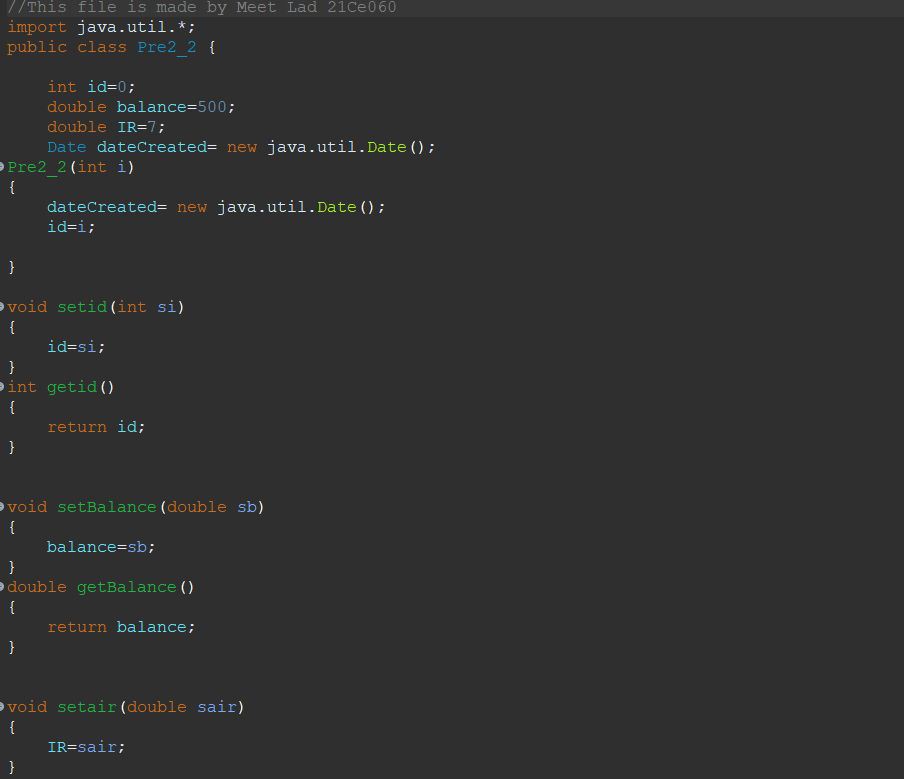
• A method named getMonthlyInterestRate() that returns the monthly interest rate.

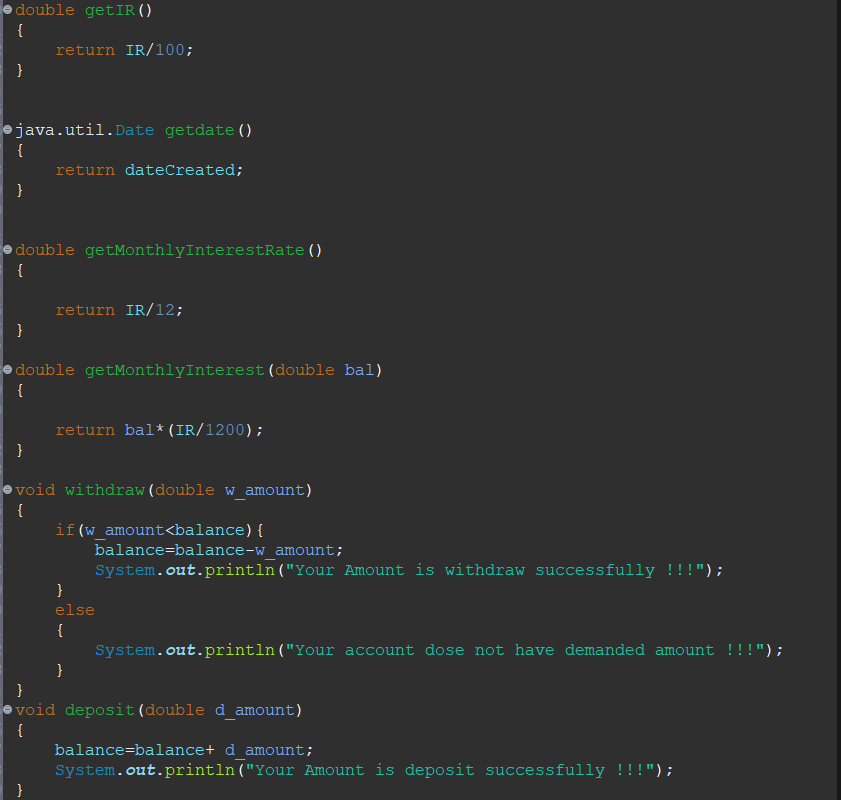
• A method named getMonthlyInterest() that returns the monthly interest.

• A method named withdraw that withdraws a specified amount from the account.

• A method named deposit that deposits a specified amount to the account

Code:







3.Use the Account class created as above to simulate an ATM machine. Create 10 accounts with id AC001…..AC010 with initial balance 300₹. The system prompts the users to enter an id. If the id is entered incorrectly, ask the user to enter a correct id. Once an id is accepted, display menu with multiple choices.

1. Balance inquiry

2. Withdraw money [Maintain minimum balance 300₹]

3. Deposit money

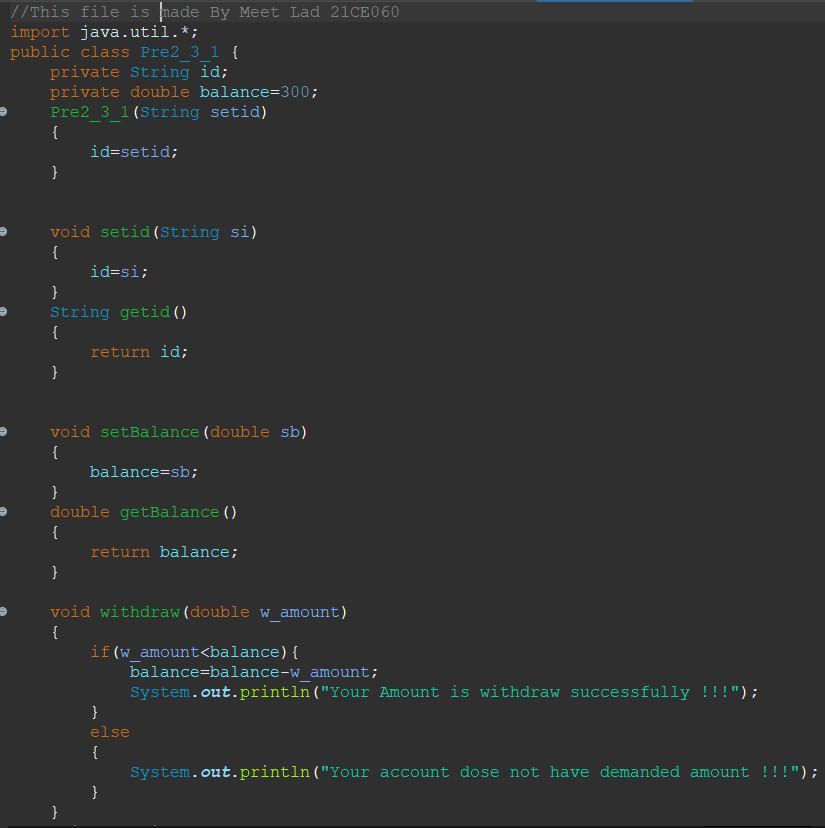
4. Money Transfer

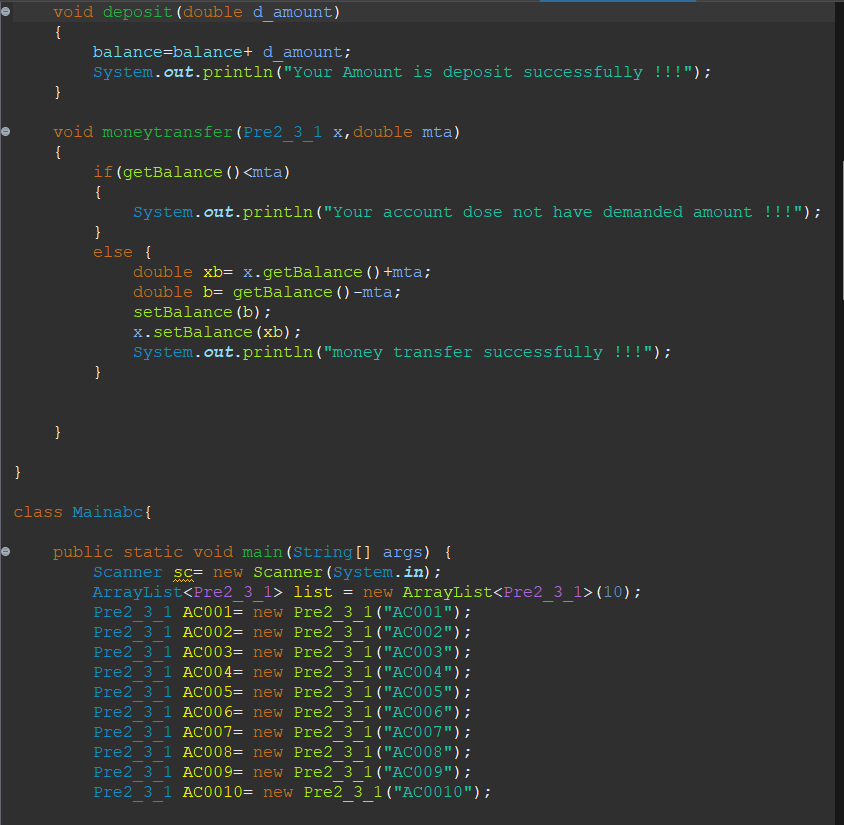
5. Create Account

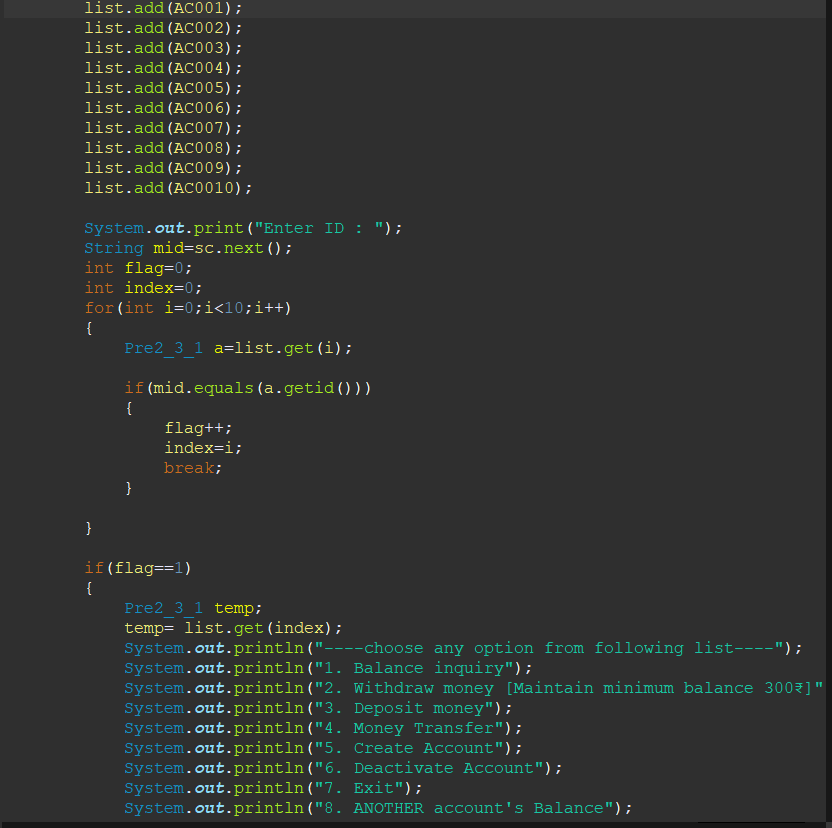
6. Deactivate Account

7. Exit

Hint: Use ArrayList, which is can shrink and expand with compared to Array.

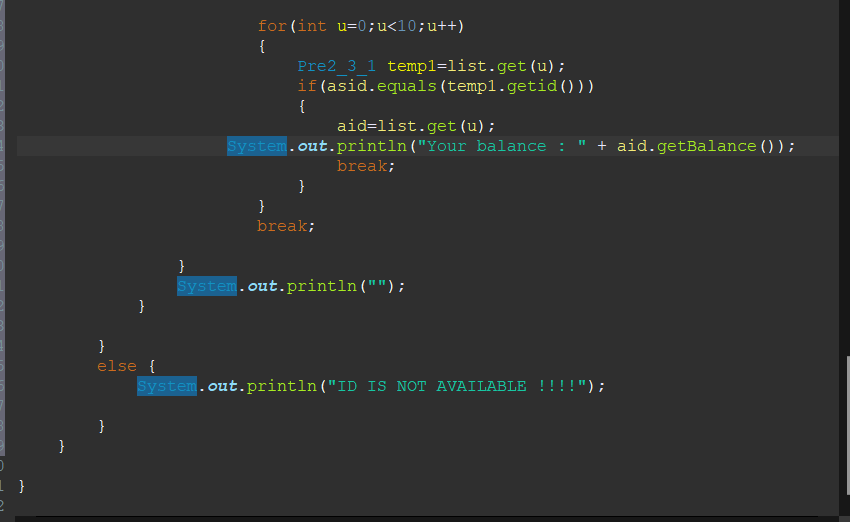










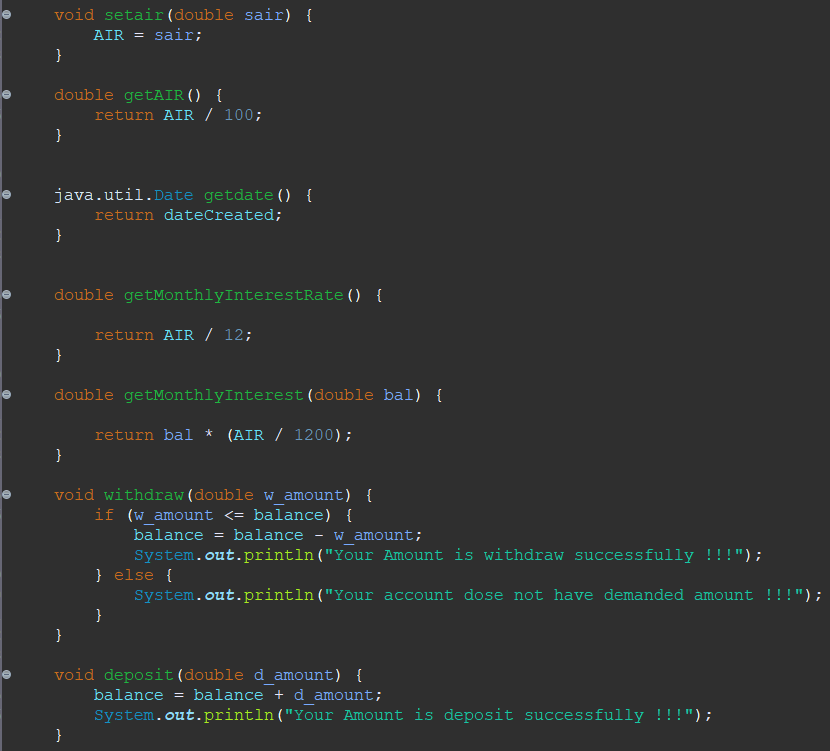


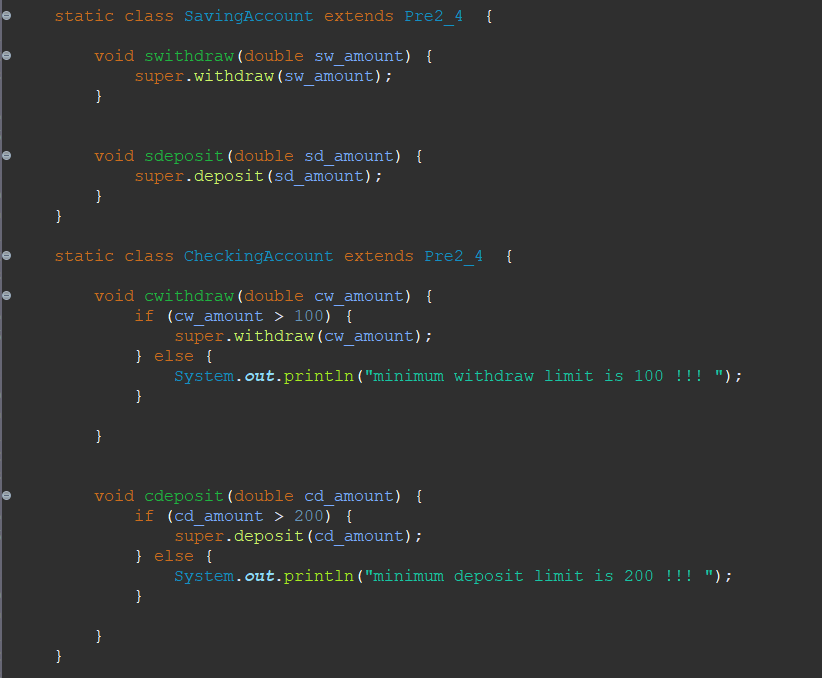
**Hint: Use ArrayList, which is can shrink and expand with compared to Array.**

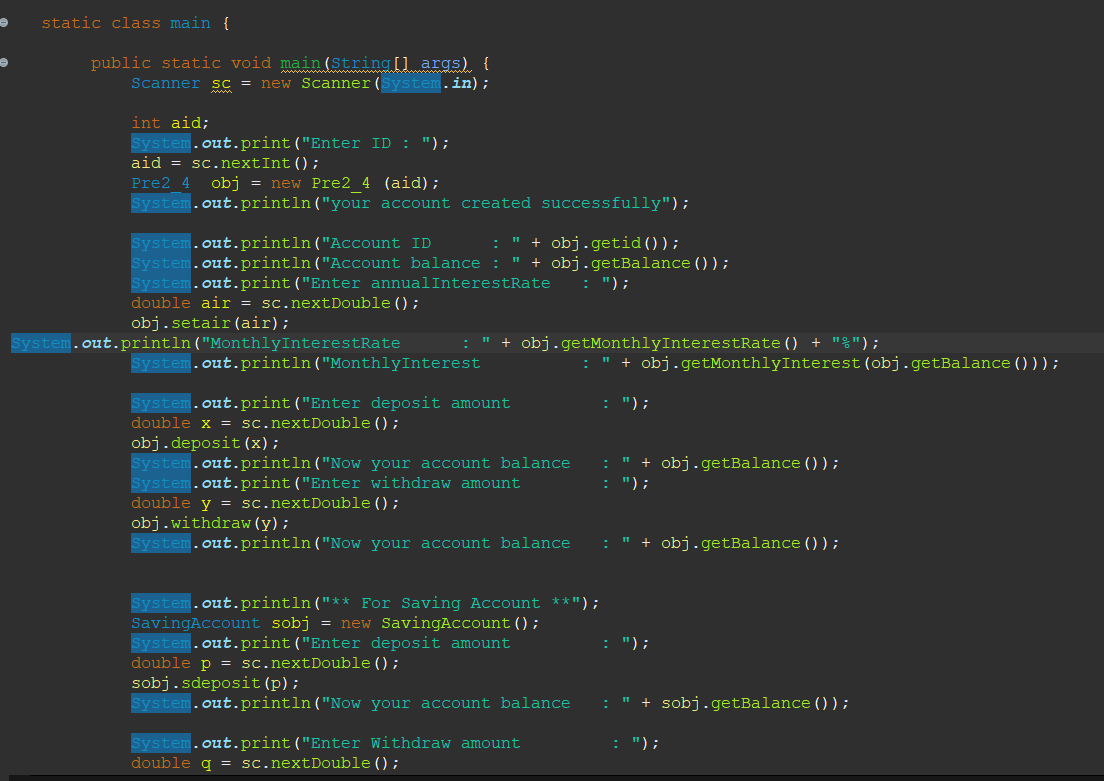
**4. (Subclasses of Account) In Programming Exercise 2, the Account class was defined to model a bank account. An account has the properties account number, balance, annual interest rate, and date created, and methods to deposit and withdraw funds. Create two subclasses for checking and saving accounts. A checking account has an overdraft limit, but a savings account cannot be overdrawn. Draw the UML diagram for the classes and then implement them. Write a test program that creates objects of Account, SavingsAccount, and CheckingAccount and invokes their toString() methods.**

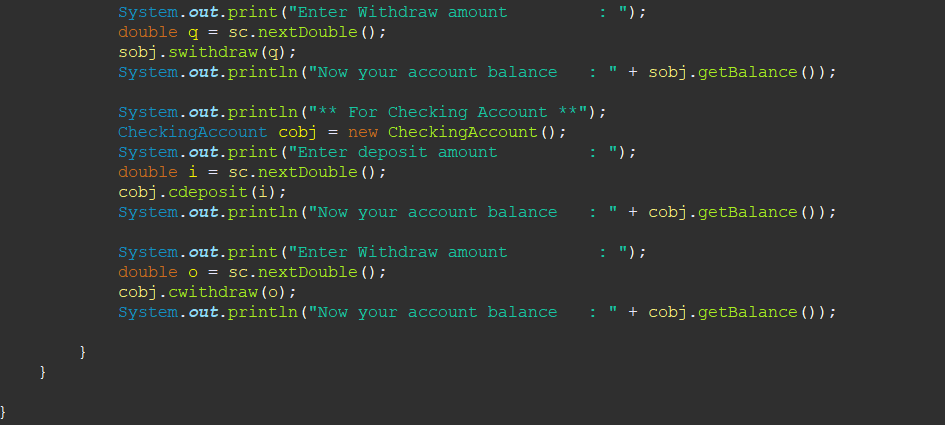
Code:











**5.Develop a Program that illustrate method overloading concept**

**Code:**

