CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

CHANDUBHAI S PATEL INSTITUTE OF TECHNOLOGY

Name:- Patel Vraj

ID:- 21CE105

CSPIT – CE

GitHub Link:- `https://github.com/PatelVraj10/java-practical-file-1`

	Practical-2
Practical 2.1	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.

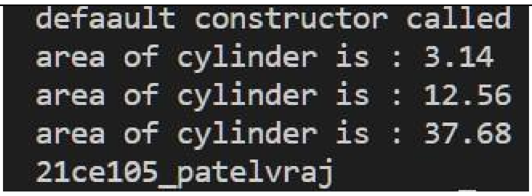
CODE

```java
// this program is prepared by 21ce105_patelvraj
// 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.
// GITHUB LINK : https://github.com/PatelVraj10/java-practical-file-1
public class cylinder
{
    double r=1;
    double h=1;
    double area;

     public cylinder()
    {
        System.out.println("defaault constructor called");
    }
    public cylinder(double a)
    {
        r=a;
    }
    public cylinder(double a,double b)
    {
        r=a;
        h=b;
    }
    public void getarea()
    {
        area=3.14*r*r*h;
        System.out.println("area of cylinder is : "+area);
    }
}
```

MAIN PROGRAM

```
public class Testcylinder
{
    public static void main(String[] args)
    {
        cylinder c1=new cylinder();
        cylinder c2=new cylinder(2);
        cylinder c3=new cylinder(2,3);
        c1.getarea();
        c2.getarea();
        c3.getarea();
        System.out.println("21ce105_patelvraj");
    }

}
```

output



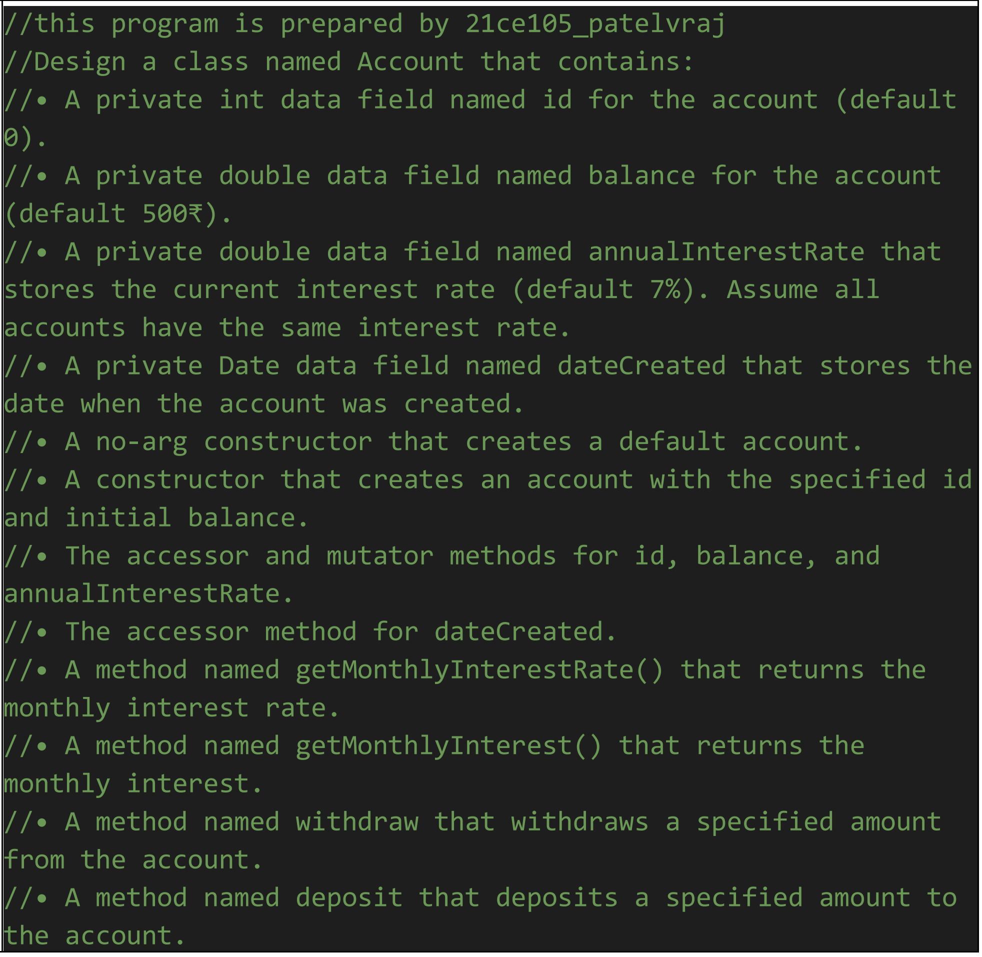
```
defaault constructor called
area of cylinder is : 3.14
area of cylinder is : 12.56
area of cylinder is : 37.68
21ce105_patelvraj
```

Practical 2.2

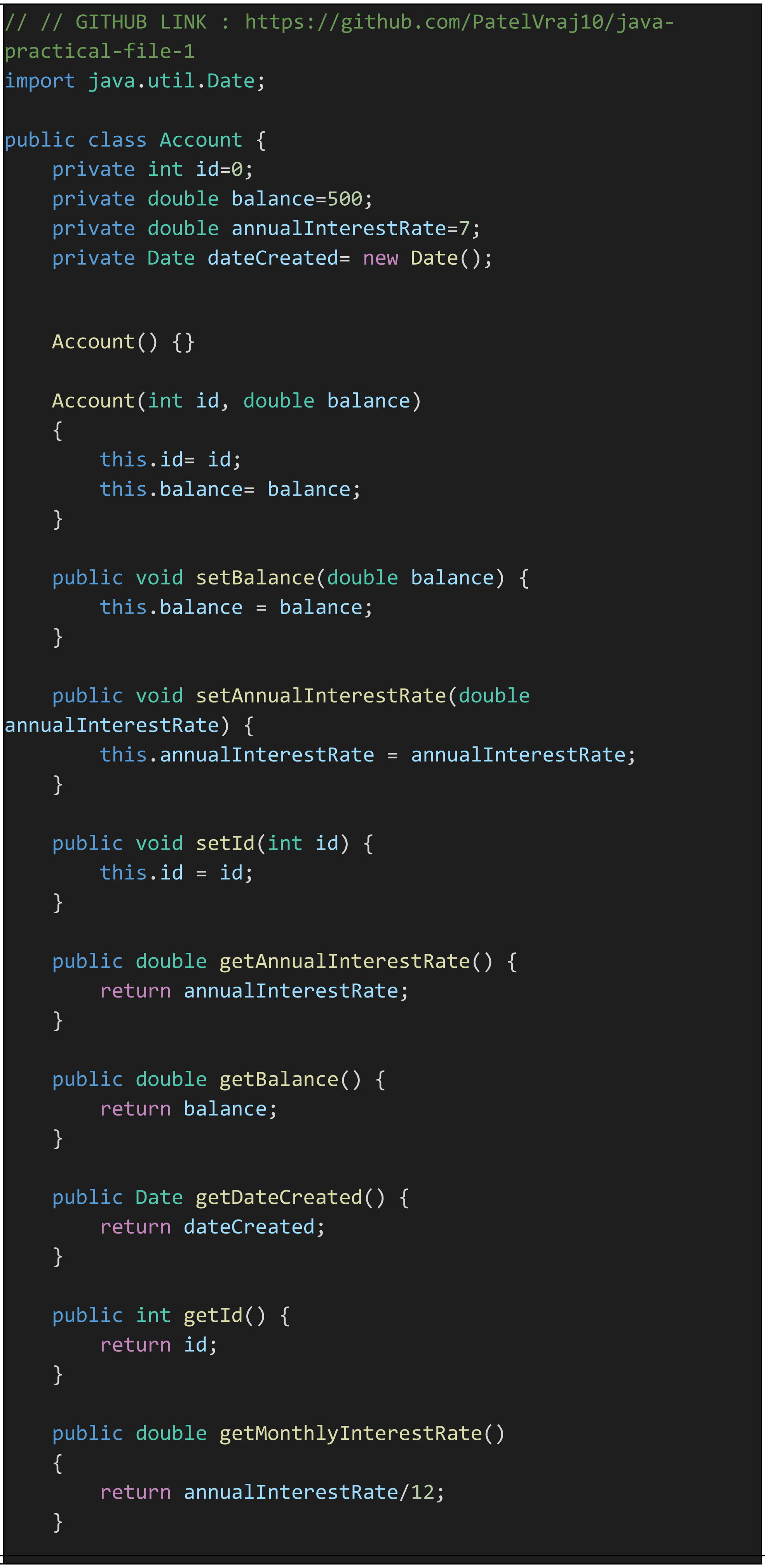
Design a class named Account that contains:

- 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.

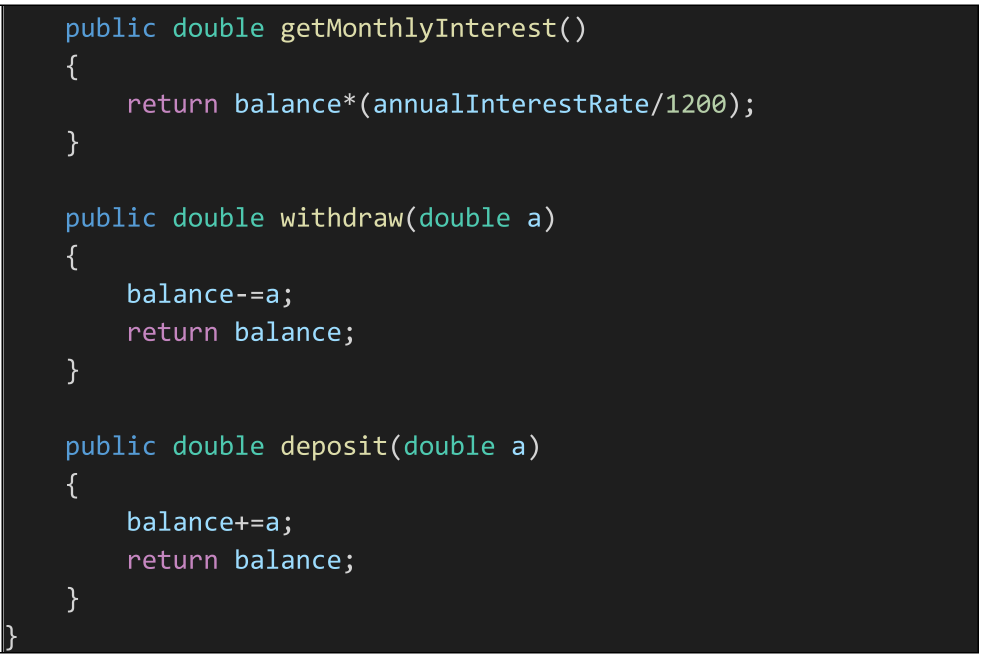
CODE



```
//this program is prepared by 21ce105_patelvraj
//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.
```

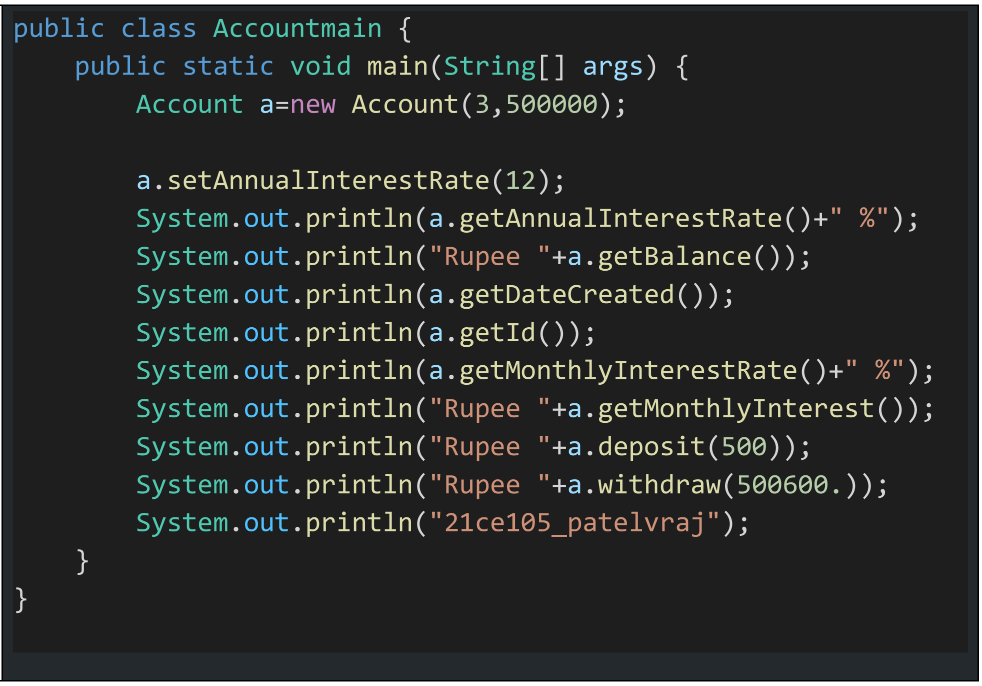


```
// // GITHUB LINK : https://github.com/PatelVraj10/java-
practical-file-1
import java.util.Date;

public class Account {
    private int id=0;
    private double balance=500;
    private double annualInterestRate=7;
    private Date dateCreated= new Date();


    Account() {}

    Account(int id, double balance)
    {
        this.id= id;
        this.balance= balance;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

    public void setAnnualInterestRate(double
annualInterestRate) {
        this.annualInterestRate = annualInterestRate;
    }

    public void setId(int id) {
        this.id = id;
    }

    public double getAnnualInterestRate() {
        return annualInterestRate;
    }

    public double getBalance() {
        return balance;
    }

    public Date getDateCreated() {
        return dateCreated;
    }

    public int getId() {
        return id;
    }

    public double getMonthlyInterestRate()
    {
        return annualInterestRate/12;
    }
```



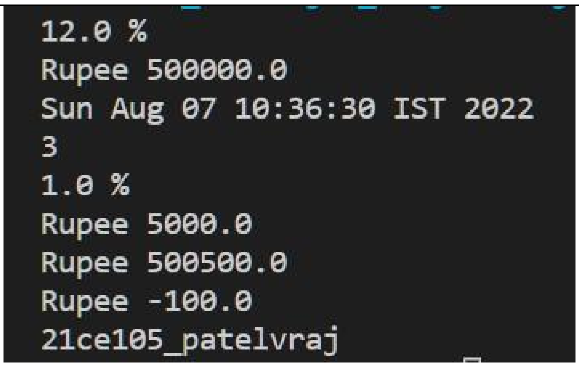
```
    public double getMonthlyInterest()
    {
        return balance*(annualInterestRate/1200);
    }

    public double withdraw(double a)
    {
        balance-=a;
        return balance;
    }

    public double deposit(double a)
    {
        balance+=a;
        return balance;
    }
}
```

Main program



```
public class Accountmain {
    public static void main(String[] args) {
        Account a=new Account(3,500000);

        a.setAnnualInterestRate(12);
        System.out.println(a.getAnnualInterestRate()+" %");
        System.out.println("Rupee "+a.getBalance());
        System.out.println(a.getDateCreated());
        System.out.println(a.getId());
        System.out.println(a.getMonthlyInterestRate()+" %");
        System.out.println("Rupee "+a.getMonthlyInterest());
        System.out.println("Rupee "+a.deposit(500));
        System.out.println("Rupee "+a.withdraw(500600.));
        System.out.println("21ce105_patelvraj");
    }
}
```

Output

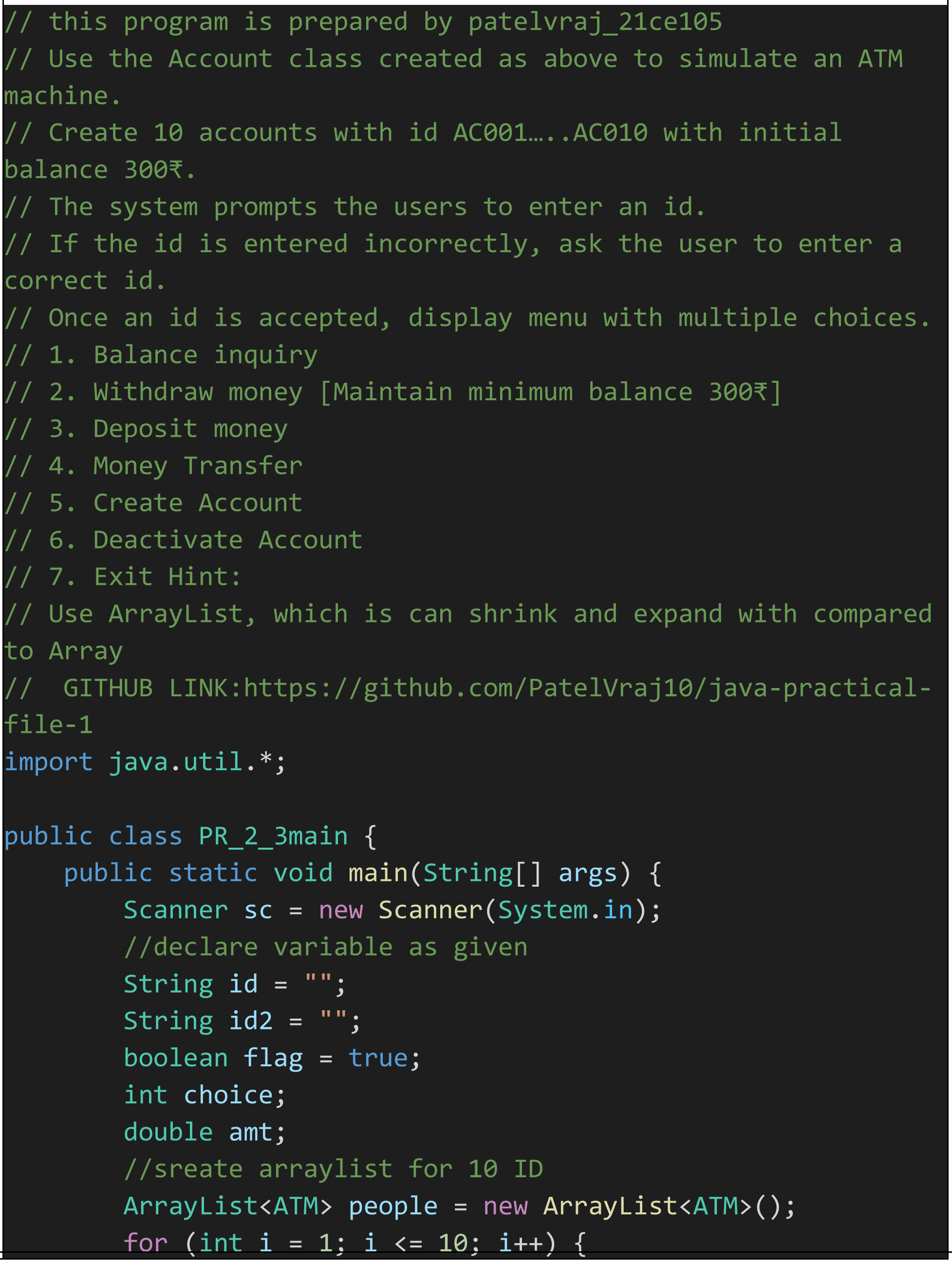


```
12.0 %
Rupee 500000.0
Sun Aug 07 10:36:30 IST 2022
3
1.0 %
Rupee 5000.0
Rupee 500500.0
Rupee -100.0
21ce105_patelvraj
```

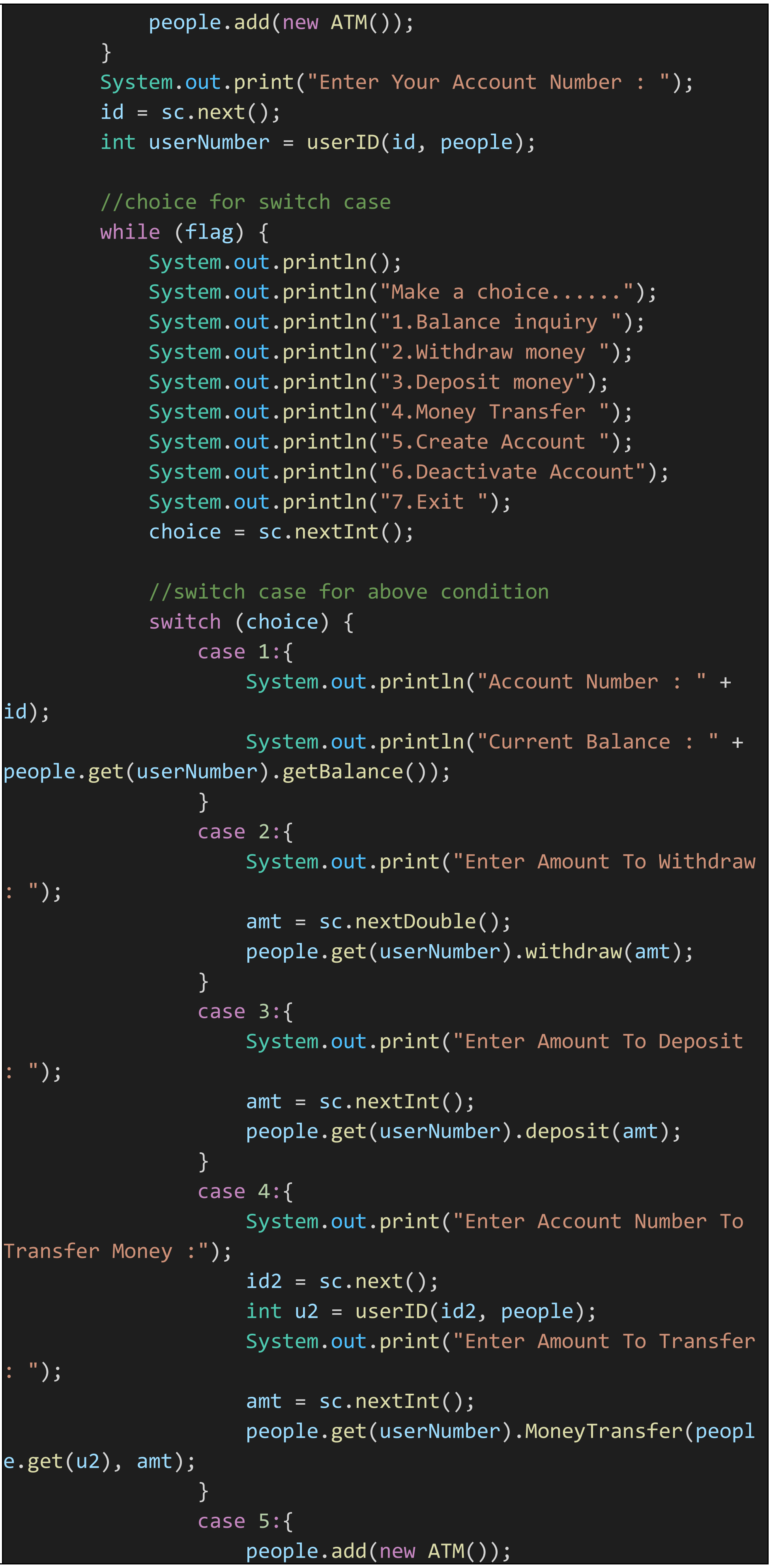
Practical 2.3

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

CODE



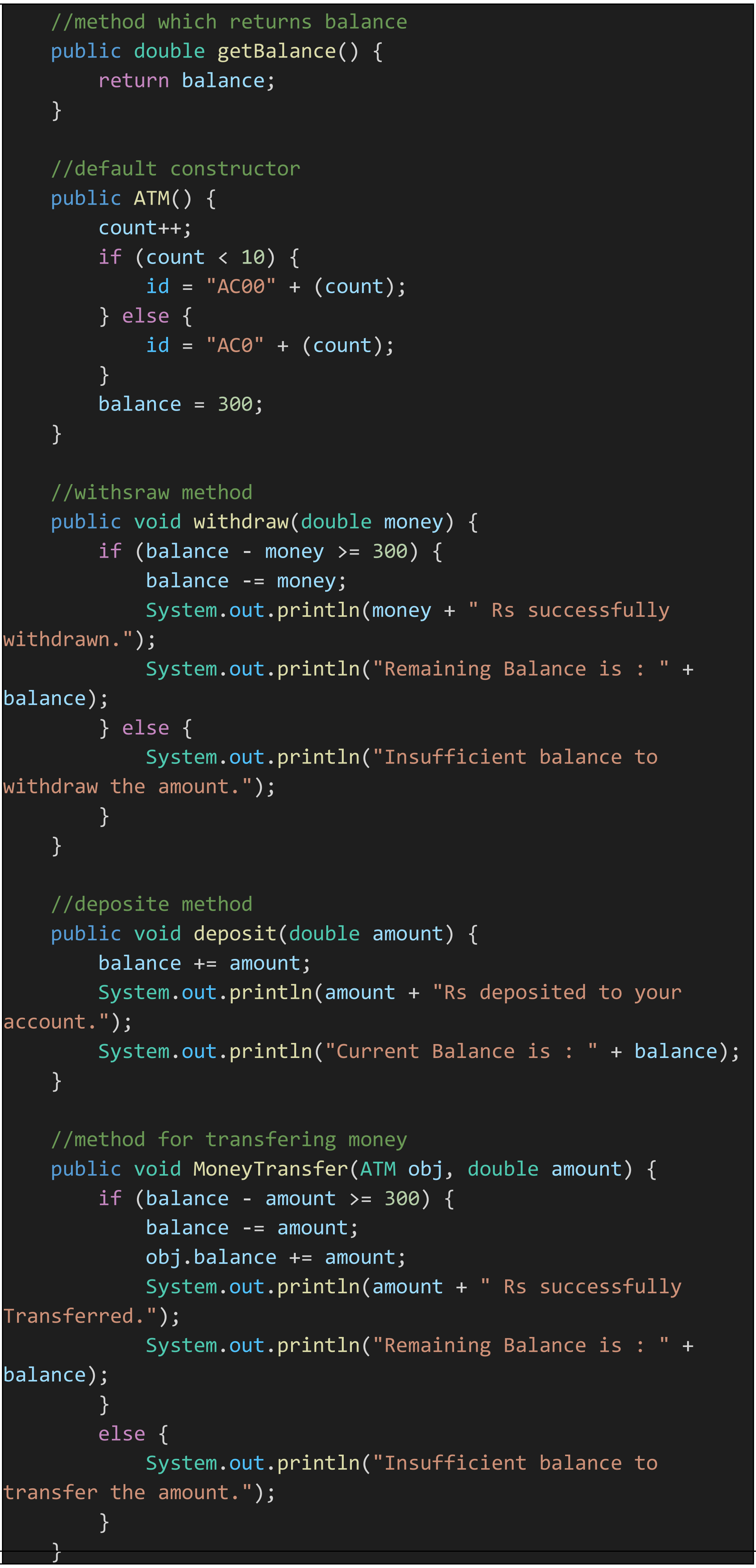
```
// this program is prepared by patelvraj_21ce105
// Use the Account class created as above to simulate an ATM machine.
// Create 10 accounts with id AC001…..AC010 with initial balance 300₹.
// The system prompts the users to enter an id.
// If the id is entered incorrectly, ask the user to enter 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
//  GITHUB LINK:https://github.com/PatelVraj10/java-practical-file-1
import java.util.*;

public class PR_2_3main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        //declare variable as given
        String id = "";
        String id2 = "";
        boolean flag = true;
        int choice;
        double amt;
        //sreate arraylist for 10 ID
        ArrayList<ATM> people = new ArrayList<ATM>();
        for (int i = 1; i <= 10; i++) {
```



```
            people.add(new ATM());
        }
        System.out.print("Enter Your Account Number : ");
        id = sc.next();
        int userNumber = userID(id, people);

        //choice for switch case
        while (flag) {
            System.out.println();
            System.out.println("Make a choice......");
            System.out.println("1.Balance inquiry ");
            System.out.println("2.Withdraw money ");
            System.out.println("3.Deposit money");
            System.out.println("4.Money Transfer ");
            System.out.println("5.Create Account ");
            System.out.println("6.Deactivate Account");
            System.out.println("7.Exit ");
            choice = sc.nextInt();

            //switch case for above condition
            switch (choice) {
                case 1:{
                    System.out.println("Account Number : " +
id);
                    System.out.println("Current Balance : " +
people.get(userNumber).getBalance());
                }
                case 2:{
                    System.out.print("Enter Amount To Withdraw
: ");
                    amt = sc.nextDouble();
                    people.get(userNumber).withdraw(amt);
                }
                case 3:{
                    System.out.print("Enter Amount To Deposit
: ");
                    amt = sc.nextInt();
                    people.get(userNumber).deposit(amt);
                }
                case 4:{
                    System.out.print("Enter Account Number To
Transfer Money :");
                    id2 = sc.next();
                    int u2 = userID(id2, people);
                    System.out.print("Enter Amount To Transfer
: ");
                    amt = sc.nextInt();
                    people.get(userNumber).MoneyTransfer(peopl
e.get(u2), amt);
                }
                case 5:{
                    people.add(new ATM());
                    System.out.println("Account Created 
Successfully.");
                    System.out.println("The New Account Number 
Is :" + people.get(people.size() - 1).getId());
                }
                case 6:{
                    people.remove(userNumber);
                    System.out.println("Account Deleted 
Successfully.");
                    flag = false;
                }
                case 7:flag = false;
                default:System.out.println("Make a valid 
choice..");
            }
        }

    }

    //method for show user data
    public static int userID(String id, ArrayList<ATM>people) 
{
        Scanner s = new Scanner(System.in);
        int user = 10000;
        int i;
        for (i = 0; i < people.size(); i++) {
            if (id.equals(people.get(i).getId())) {
                user = i;
                break;
            }
        }
        if (i == people.size()) {
            System.out.println("No Such Account Exists.\nTry 
Again..");
            System.out.print("Enter your account id :");
            id = s.next();
            return userID(id, people);
        }
        else
        return user;
    }
}
```

MAIN PROGRAM

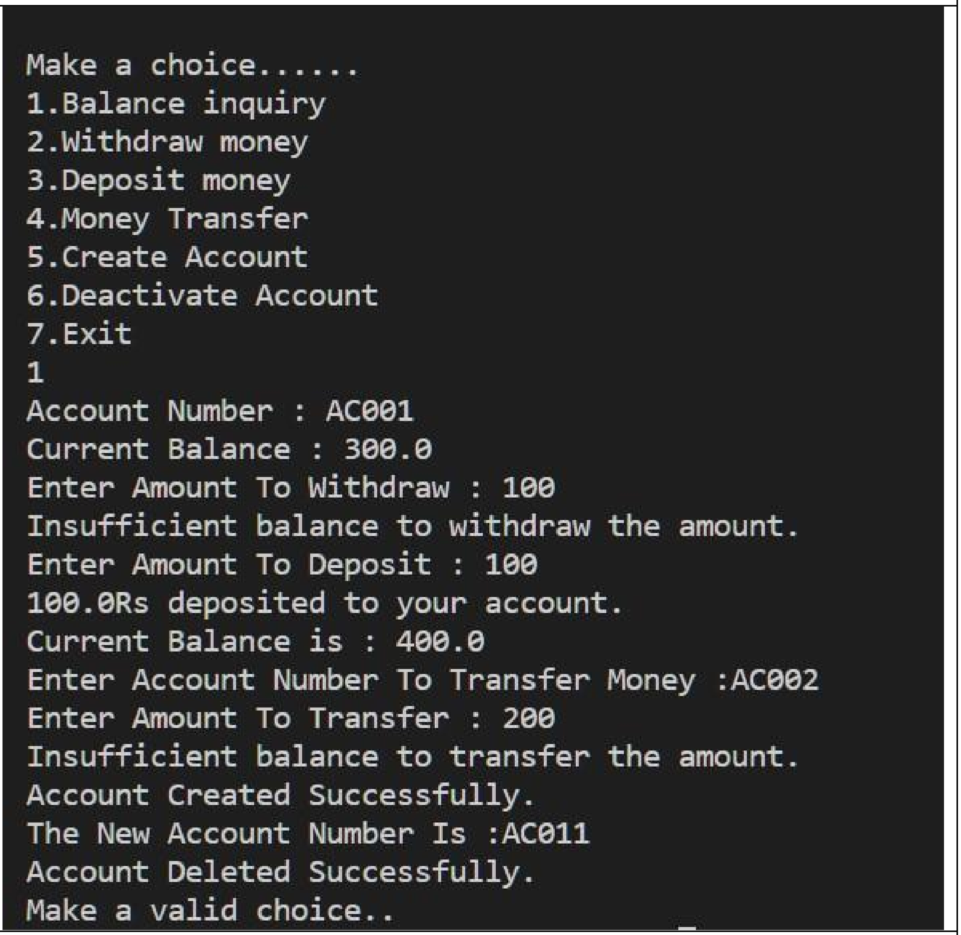
```
public class ATM {
    private static int count;
    private final String id;
    private double balance;

    //method which returns ID
    public String getId() {
        return id;
    }
```



```
    //method which returns balance
    public double getBalance() {
        return balance;
    }

    //default constructor
    public ATM() {
        count++;
        if (count < 10) {
            id = "AC00" + (count);
        } else {
            id = "AC0" + (count);
        }
        balance = 300;
    }

    //withsraw method
    public void withdraw(double money) {
        if (balance - money >= 300) {
            balance -= money;
            System.out.println(money + " Rs successfully
withdrawn.");
            System.out.println("Remaining Balance is : " +
balance);
        } else {
            System.out.println("Insufficient balance to
withdraw the amount.");
        }
    }

    //deposite method
    public void deposit(double amount) {
        balance += amount;
        System.out.println(amount + "Rs deposited to your
account.");
        System.out.println("Current Balance is : " + balance);
    }

    //method for transfering money
    public void MoneyTransfer(ATM obj, double amount) {
        if (balance - amount >= 300) {
            balance -= amount;
            obj.balance += amount;
            System.out.println(amount + " Rs successfully
Transferred.");
            System.out.println("Remaining Balance is : " +
balance);
        }
        else {
            System.out.println("Insufficient balance to
transfer the amount.");
        }
    }
```

```
}
```

sOUTPUT



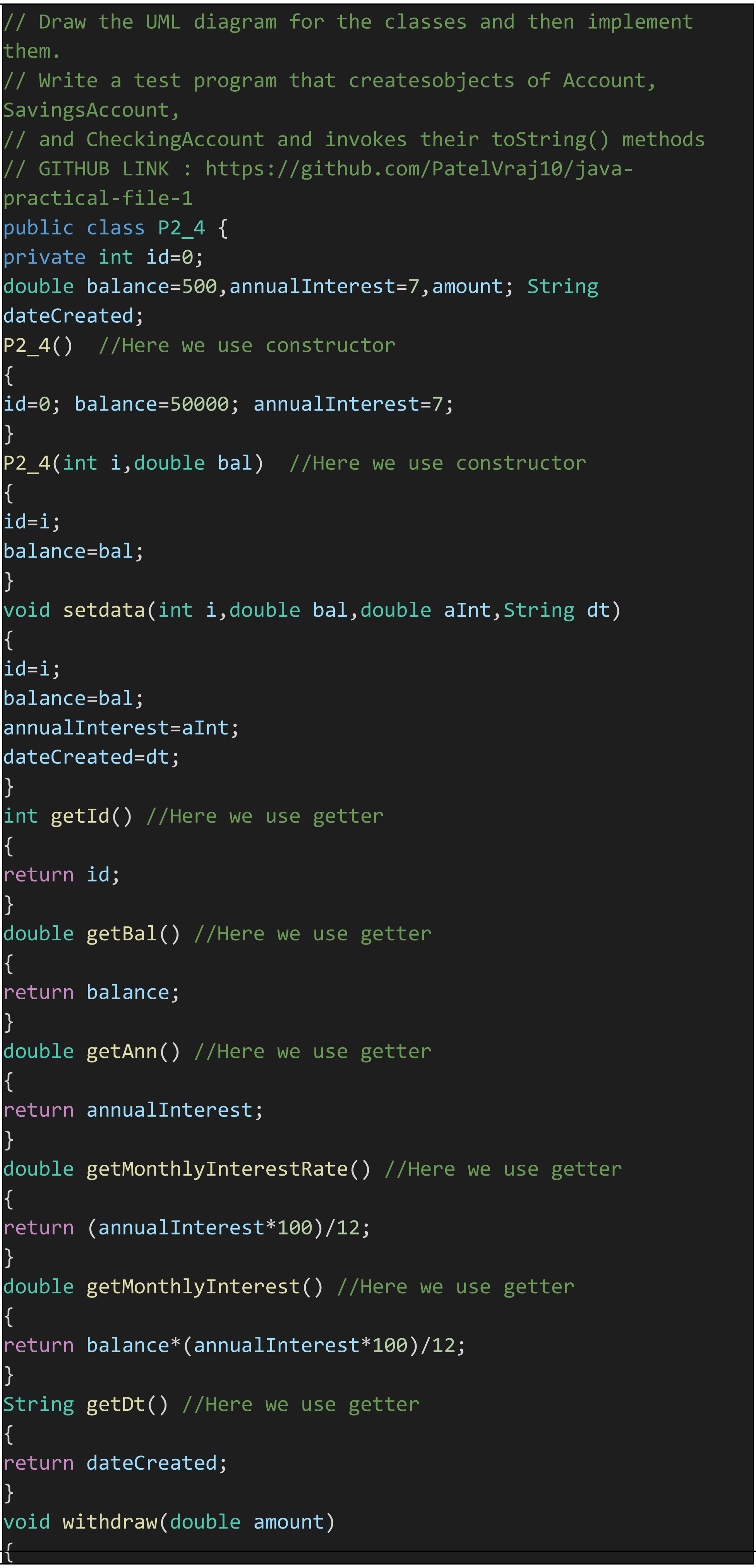
```
Make a choice......
1.Balance inquiry
2.Withdraw money
3.Deposit money
4.Money Transfer
5.Create Account
6.Deactivate Account
7.Exit
1
Account Number : AC001
Current Balance : 300.0
Enter Amount To Withdraw : 100
Insufficient balance to withdraw the amount.
Enter Amount To Deposit : 100
100.0Rs deposited to your account.
Current Balance is : 400.0
Enter Account Number To Transfer Money :AC002
Enter Amount To Transfer : 200
Insufficient balance to transfer the amount.
Account Created Successfully.
The New Account Number Is :AC011
Account Deleted Successfully.
Make a valid choice..
```

Practical 2.4

(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 invokestheir toString() methods

CODE

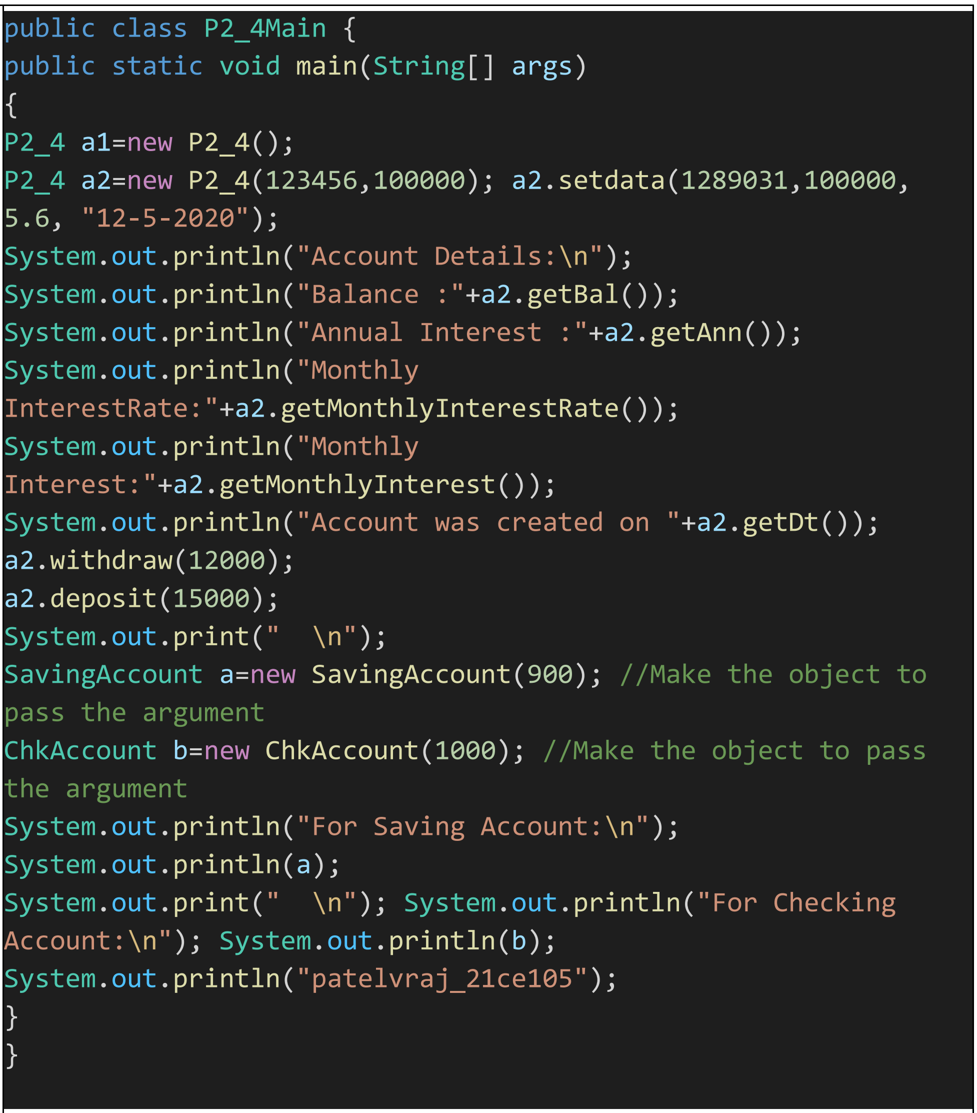
```
// this program is prepared by patelvraj_21ce105
// (Subclasses of Account) In Programming Exercise 2, the
Account class was defined to model a bank account.
// An account has the properties account number, balance,
annual interest rate,
// and date created, and methods to deposit and withdraw
funds.
// 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 createsobjects of Account,
SavingsAccount,
// and CheckingAccount and invokes their toString() methods
// GITHUB LINK : https://github.com/PatelVraj10/java-
practical-file-1
public class P2_4 {
private int id=0;
double balance=500,annualInterest=7,amount; String
dateCreated;
P2_4()  //Here we use constructor
{
id=0; balance=50000; annualInterest=7;
}
P2_4(int i,double bal)  //Here we use constructor
{
id=i;
balance=bal;
}
void setdata(int i,double bal,double aInt,String dt)
{
id=i;
balance=bal;
annualInterest=aInt;
dateCreated=dt;
}
int getId() //Here we use getter
{
return id;
}
double getBal() //Here we use getter
{
return balance;
}
double getAnn() //Here we use getter
{
return annualInterest;
}
double getMonthlyInterestRate() //Here we use getter
{
return (annualInterest*100)/12;
}
double getMonthlyInterest() //Here we use getter
{
return balance*(annualInterest*100)/12;
}
String getDt() //Here we use getter
{
return dateCreated;
}
void withdraw(double amount)
{
```

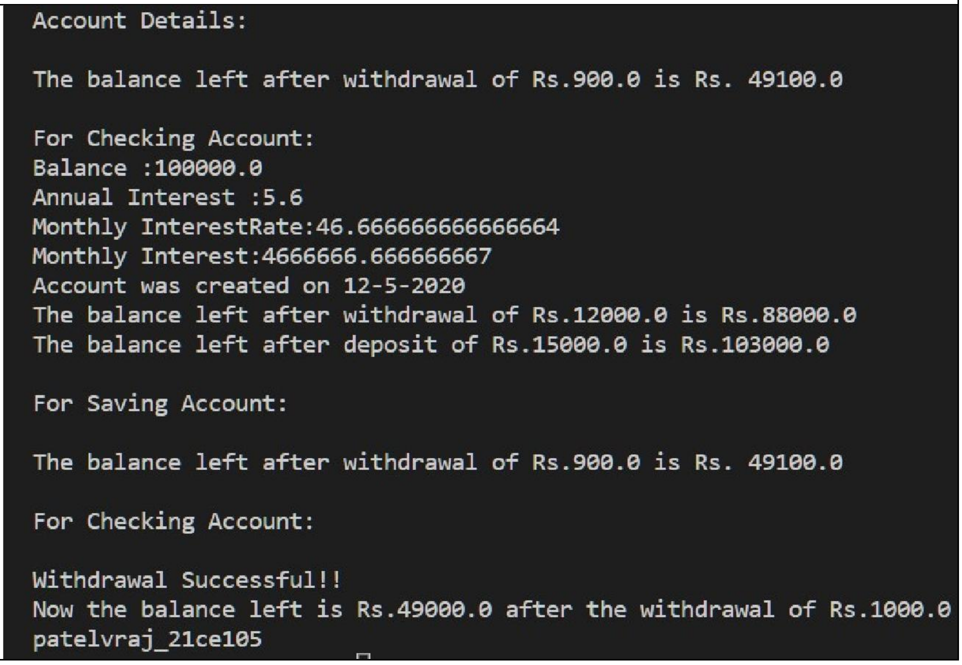
```
balance-=amount; if(balance>0)
System.out.println("The balance left after withdrawal of
Rs."+amount+" is Rs."+balance);
else
System.out.println("Withdrawal of Rs."+amount+" is not
possible!!");
}
void deposit(double amount)
{
balance+=amount;
System.out.println("The balance left after deposit of
Rs."+amount+" is Rs."+balance);
}
}
class SavingAccount extends P2_4    //Here we make a new class
for more bank details.
{
SavingAccount(double a)
{
amount=a; balance-=amount;
}
public String toString()
{
if(balance>=3000)   //Here we use if else to check balance
left after widrawal and for minimum balance required
{
return "The balance left after withdrawal of Rs."+amount+" is
Rs. "+balance;
}
else
{
return "Beyond1 Over Draft Limit Not Possible!!\nMinimum
balance of Rs. 3000 is required.";
}
}
}
class ChkAccount extends P2_4   //Here we make a class for
check account details
{
ChkAccount(double am)
{
amount = am; balance-=amount;
}
public String toString()
{
System.out.println("Withdrawal Successful!!");
return "Now the balance left is Rs."+balance+" after the
withdrawal of Rs."+amount;
}
}
```

MAIN PROGRAM



```
public class P2_4Main {
public static void main(String[] args)
{
P2_4 a1=new P2_4();
P2_4 a2=new P2_4(123456,100000); a2.setdata(1289031,100000,
5.6, "12-5-2020");
System.out.println("Account Details:\n");
System.out.println("Balance :"+a2.getBal());
System.out.println("Annual Interest :"+a2.getAnn());
System.out.println("Monthly
InterestRate:"+a2.getMonthlyInterestRate());
System.out.println("Monthly
Interest:"+a2.getMonthlyInterest());
System.out.println("Account was created on "+a2.getDt());
a2.withdraw(12000);
a2.deposit(15000);
System.out.print("  \n");
SavingAccount a=new SavingAccount(900); //Make the object to
pass the argument
ChkAccount b=new ChkAccount(1000); //Make the object to pass
the argument
System.out.println("For Saving Account:\n");
System.out.println(a);
System.out.print("  \n"); System.out.println("For Checking
Account:\n"); System.out.println(b);
System.out.println("patelvraj_21ce105");
}
}
```

OUTPUT



```
Account Details:

The balance left after withdrawal of Rs.900.0 is Rs. 49100.0

For Checking Account:
Balance :100000.0
Annual Interest :5.6
Monthly InterestRate:46.666666666666664
Monthly Interest:4666666.666666667
Account was created on 12-5-2020
The balance left after withdrawal of Rs.12000.0 is Rs.88000.0
The balance left after deposit of Rs.15000.0 is Rs.103000.0

For Saving Account:

The balance left after withdrawal of Rs.900.0 is Rs. 49100.0

For Checking Account:

Withdrawal Successful!!
Now the balance left is Rs.49000.0 after the withdrawal of Rs.1000.0
patelvraj_21ce105
```

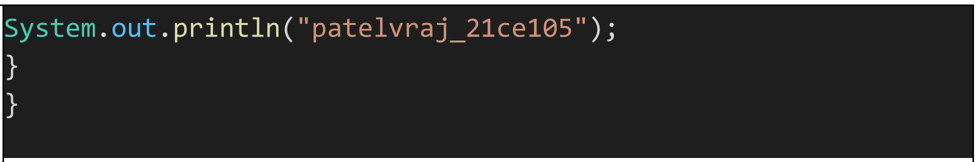
Practical 2.5	**Develop a Program that illustrate method overloading concept.**

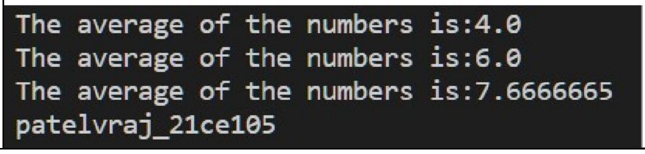
CODE

```java
// this program is prepared by patelvraj_21ce105
// Develop a Program that illustrate method overloading concept.
// GITHUB LINK : https://github.com/PatelVraj10/java-practical-file-1
public class P2_5 {
float getAverage(float a)
{
    //Here we use getter
    float avg;
    avg=a; return avg;
}
float getAverage(float a,float b)
{
    //Here we use getter
    float avg;
    avg=(a+b)/2; return avg;
}
float getAverage(float a,float b,float c)
{
    //Here we use getter
    float avg;
    avg=(a+b+c)/3;
    return avg;
}
}
```

MAIN PROGRAM

```java
public class P2_5Main
{
public static void main(String[] args)
{
    P2_5 p1=new P2_5();
    P2_5 sc1= new P2_5();
    P2_5 sc2=new P2_5();
    P2_5 sc3 =new P2_5();

System.out.println("The average of the numbers is:"+sc1.getAverage(4));
//call the function getAverage
System.out.println("The average of the numbers is:"+sc2.getAverage(4,8));
//call an overloaded function getAverage
System.out.println("The average of the numbers is:"+sc3.getAverage(4,7,12));
//call an overloaded function getAverage
```

	
	System.out.println("patelvraj_21ce105");
	}
	}

OUTPUT	 The average of the numbers is:4.0 The average of the numbers is:6.0 The average of the numbers is:7.6666665 patelvraj_21ce105