Name of Experiment :- Write a test application named Account Test that demostrate the account capabilities creating 5 objects with constructor.

Introduction:-

We have to design a class representating a bank account and we also have to write a test application to demostrate the capabilities of the bank account class using constructors.

Objectives:-

- experiencing the object oriented approach of solving a problem
- demonstrate different aspects of object oriented programming.
- how to construct and use object.
- how to define methods and call them.

Analysis :-

After analysing our problem we have to found following components of our problem to solve it.

'Class Account : this is the class that will represent a bank account

☐ Method :-

- deposit : to deposit an amount
- withdraw : to withdraw an amount
- showBalance : to display the name and balance

☐ Data Members :-

- holderName : a string representing the account holder name
- accNum : a string representing the account number.
- accType : an integer representing the account type
- balance : a real number representing the balance of the account

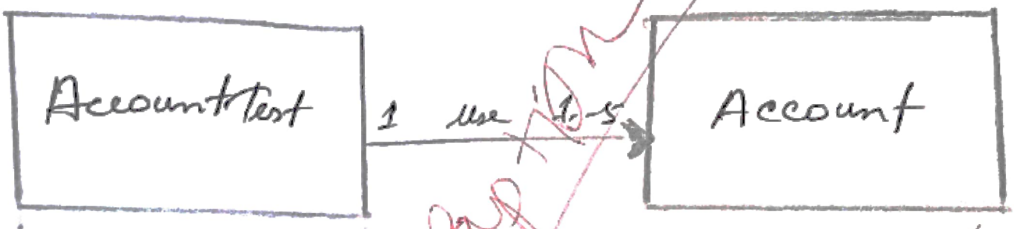
☐ Methods :-

- init : to assign intial values.
- deposit : to deposit an amount
- withdraw : to withdraw an amount
- showBalance : to display the name and balance

Class AccountTest :- this class will contain the main method

☐ Methods :

- main : main method will use the Account class to create objects and demonstrate its capabilities .



Design :-

Account
- holderName : string - accNum : string - accType : int - balance : double
+ init (n : string, num : string, type : int, amount : double) : void + deposit (amount : double) : void + withdraw (amount : double) : void + showBalance () : void

AccountTest
+ main (args : string[1]) : void

```
Account():
  // default initialize every data member

Account(n: string, num: string, type: int, amount: double);
  // parameterized constructors
    holderName = n
    accNum  = num
    accType = type
    balance = amount

deposit(amount: double):
  // deposit amount and add that to the existing balance
    balance += amount

withdraw(amount: double):
  //withdraw specific amount if available
  if amount greater than balance:
    print error message
  else:
    balance = balance - amount
    print new balance after withdrawl

ShowBalance():
  // show the available balance
```

print account holder name and available balance

main (args : string[]) :

// create 5 Account class object and show capabilities

create 5 Account class object

show method calls

Implementation :-

```
Class Account {
    private string holderName;
    private string accNum;
    private int accType;
    private double balance;

    Account () {
    }
    Account (string name, string num, int type, double amount) {
    holderName = name;
    accNum     = num;
    accType    = type;
    balance    = amount;
    }
```

```
class AccountTest {
 public static void main (String[] args) {
 Account [] accounts = new Account[5];
 accounts [0] = new Account( "Garry Kasparov", 123abc", 1,500);
 accounts [1] = new Account ("Magnus Carlsen", 123abd", 1, 4500);
accounts [2] = new Account( "Bobby Fischer", 12cbe, 5, 4399);
accounts [3] = new Account ("Hikaru Nakamura", 123kbf", 0, 4000);
accounts [4] = new Account ("Viswi Anand" 456def", 0, 4570);

accounts [1]. deposit (5000);
accounts [1]. withdraw (55555);
accounts [1]. showBalance ();
  }
}
```

Conclusion :-

From a test program we get the below output :-

Your new balance after withdraw : 8944.45

Name : Magnus Carlsen

Balance : 8944.45