* **Write a simple Banking System program by using OOPs concept where you can get account Holder name balance etc?**

**import** java.util.Scanner;

**class** BankDetails {

**private** String accno;

**private** String name;

**private** String acc\_type;

**private** **long** balance;

    Scanner sc = **new** Scanner(System.in);

    //method to open new account

**public** **void** openAccount() {

        System.out.print("Enter Account No: ");

        accno = sc.next();

        System.out.print("Enter Account type: ");

        acc\_type = sc.next();

        System.out.print("Enter Name: ");

        name = sc.next();

        System.out.print("Enter Balance: ");

        balance = sc.nextLong();

    }

    //method to display account details

**public** **void** showAccount() {

        System.out.println("Name of account holder: " + name);

        System.out.println("Account no.: " + accno);

        System.out.println("Account type: " + acc\_type);

        System.out.println("Balance: " + balance);

    }

    //method to deposit money

**public** **void** deposit() {

**long** amt;

        System.out.println("Enter the amount you want to deposit: ");

        amt = sc.nextLong();

        balance = balance + amt;

    }

    //method to withdraw money

**public** **void** withdrawal() {

**long** amt;

        System.out.println("Enter the amount you want to withdraw: ");

        amt = sc.nextLong();

**if** (balance >= amt) {

            balance = balance - amt;

           System.out.println("Balance after withdrawal: " + balance);

        } **else** {

            System.out.println("Your balance is less than " + amt + "\tTransaction failed...!!" );

        }

    }

    //method to search an account number

**public** **boolean** search(String ac\_no) {

**if** (accno.equals(ac\_no)) {

            showAccount();

**return** (**true**);

        }

**return** (**false**);

    }

}

**public** **class** BankingApp {

**public** **static** **void** main(String arg[]) {

        Scanner sc = **new** Scanner(System.in);

        //create initial accounts

        System.out.print("How many number of customers do you want to input? ");

**int** n = sc.nextInt();

        BankDetails C[] = **new** BankDetails[n];

**for** (**int** i = 0; i < C.length; i++) {

            C[i] = **new** BankDetails();

            C[i].openAccount();

        }

        // loop runs until number 5 is not pressed to exit

**int** ch;

**do** {

            System.out.println("\n \*\*\*Banking System Application\*\*\*");

            System.out.println("1. Display all account details \n 2. Search by Account number\n 3. Deposit the amount \n 4. Withdraw the amount \n 5.Exit ");

            System.out.println("Enter your choice: ");

            ch = sc.nextInt();

**switch** (ch) {

**case** 1:

**for** (**int** i = 0; i < C.length; i++) {

                            C[i].showAccount();

                        }

**break**;

**case** 2:

                        System.out.print("Enter account no. you want to search: ");

                        String ac\_no = sc.next();

**boolean** found = **false**;

**for** (**int** i = 0; i < C.length; i++) {

                            found = C[i].search(ac\_no);

**if** (found) {

**break**;

                            }

                        }

**if** (!found) {

                            System.out.println("Search failed! Account doesn't exist..!!");

                        }

**break**;

**case** 3:

                        System.out.print("Enter Account no. : ");

                        ac\_no = sc.next();

                        found = **false**;

**for** (**int** i = 0; i < C.length; i++) {

                            found = C[i].search(ac\_no);

**if** (found) {

                                C[i].deposit();

**break**;

                            }

                        }

**if** (!found) {

                            System.out.println("Search failed! Account doesn't exist..!!");

                        }

**break**;

**case** 4:

                        System.out.print("Enter Account No : ");

                        ac\_no = sc.next();

                        found = **false**;

**for** (**int** i = 0; i < C.length; i++) {

                            found = C[i].search(ac\_no);

**if** (found) {

                                C[i].withdrawal();

**break**;

                            }

                        }

**if** (!found) {

                            System.out.println("Search failed! Account doesn't exist..!!");

                        }

**break**;

**case** 5:

                        System.out.println("See you soon...");

**break**;

                }

            }

**while** (ch != 5);

        }

    }

* **Write a Program where you inherit method from parent class and show method Overridden Concept?**

**class** Vehicle{

**void** run(){System.out.println("Vehicle is running");}

}

//Creating a child class

**class** Bike **extends** Vehicle{

**public** **static** **void** main(String args[]){

  //creating an instance of child class

  Bike obj = **new** Bike();

  //calling the method with child class instance

  obj.run();

  }

}

* **Write a program to show run time polymorphism in java?**

**class** Bike{

**void** run(){System.out.println("running");}

}

**class** Splendor **extends** Bike{

**void** run(){System.out.println("running safely with 60km");}

**public** **static** **void** main(String args[]){

    Bike b = **new** Splendor();//upcasting

    b.run();

  }

}

* **Write a program to show Compile time polymorphism in java?**

**//Java program to demonstrate the working of method**

**// overloading by changing the number of parameters**

**public class MethodOverloading {**

**// 1 parameter**

**void show(int num1)**

**{**

**System.out.println("number 1 : " + num1);**

**}**

**// 2 parameter**

**void show(int num1, int num2)**

**{**

**System.out.println("number 1 : " + num1**

**+ " number 2 : " + num2);**

**}**

**public static void main(String[] args)**

**{**

**MethodOverloading obj = new MethodOverloading();**

**// 1st show function**

**obj.show(3);**

**// 2nd show function**

**obj.show(4, 5);**

**}**

**}**

* **Achieve loose coupling in java by using OOPs  concept?**

**class** A

{

**void** foo()

{

  System.out.println("Inside the foo method of base class.");

}

}

// child or derived class

**class** B **extends** A

{

**void** foo()

{

  System.out.println("Inside the foo method of derived class.");

}

}

**public** **class** CouplingExample

{

// main method

**public** **static** **void** main(String argvs[])

{

// creating an object of class B

B obj = **new** B();

obj.foo();

}

}

* **What is the benefit of encapsulation in java?**

With encapsulation, it's simple and easy to debug an application. It is also possible to alter and make edits to your codebase without disrupting the regular functioning of your program. It enables the programmer to monitor the data accessibility of a class

* **Is java a t 100% Object oriented Programming language? If no why ?**

Java is not fully object oriented because it supports primitive data type like it,byte,long etc.,which are not objects. Because in JAVA we use data types like int, float, double etc which are not object oriented, and of course is what opposite of OOP is. That is why JAVA is not 100% objected oriented.

* **What are the advantages of abstraction in java?**

Adding new features to a system is easier because the interface remains the same, and the implementation details can be hidden behind it. Maintainability: Abstraction helps to reduce code duplication by providing a way to define common behavior in a single location.

* **What is an abstraction explained with an Example?**

Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation. Consider a real-life example of a man driving a car.

* **What is the final class in Java?**

The final class is a class that is declared with the final keyword. We can restrict class inheritance by making use of the final class. Final classes cannot be extended or inherited. If we try to inherit a final class, then the compiler throws an error during compilation.